Forschungsberichte aus dem
wbk Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Sebastian Tim Behrendt

Service-oriented Production Planning and Control of Reconfigurable Manufacturing Systems

Band 313

wbk
Institut für Produktionstechnik

Forschungsberichte aus dem
wbk Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Hrsg.: Prof. Dr.-Ing. Jürgen Fleischer
Prof. Dr.-Ing. Gisela Lanza
Prof. Dr.-Ing. habil. Volker Schulze
Prof. Dr.-Ing. Frederik Zanger

Sebastian Tim Behrendt

Service-oriented Production Planning and Control of Reconfigurable Manufacturing Systems

Band 313

Karlsruher Institut für Technologie

wbk
Institut für Produktionstechnik

Service-oriented Production Planning and Control of Reconfigurable Manufacturing Systems

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des

Karlsruher Instituts für Technologie (KIT)

angenommene

Dissertation

von

Sebastian Tim Behrendt, M.Sc.

Tag der mündlichen Prüfung:	27.04.2026
Hauptreferentin:	Prof. Dr.-Ing. Gisela Lanza
Korreferent:	Univ.-Prof. Dr.-Ing. Oliver Riedel

KIT
Karlsruher Institut für Technologie

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: Karlsruhe, Karlsruher Institut für Technologie, Diss., 2026

Printed in Germany.

Print-ISBN	978-3-8191-0795-5
PDF-ISBN	978-3-8191-0833-4
ISSN	2944-6430
eISSN	2944-6449

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Telefon: 02421 / 99 0 11 - 0 • Telefax: 02421 / 99 0 11 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

wbk
Institut für Produktionstechnik

Vorwort der Herausgeber

Die schnelle und effiziente Umsetzung innovativer Technologien wird vor dem Hintergrund der Globalisierung der Wirtschaft der entscheidende Wirtschaftsfaktor für produzierende Unternehmen. Universitäten können als “Wertschöpfungspartner” einen wesentlichen Beitrag zur Wettbewerbsfähigkeit der Industrie leisten, indem sie wissenschaftliche Grundlagen sowie neue Methoden und Technologien erarbeiten und aktiv den Umsetzungsprozess in die praktische Anwendung unterstützen.

Vor diesem Hintergrund wird im Rahmen dieser Schriftenreihe über aktuelle Forschungsergebnisse des Instituts für Produktionstechnik (wbk) am Karlsruher Institut für Technologie (KIT) berichtet. Unsere Forschungsarbeiten beschäftigen sich sowohl mit der Leistungssteigerung von additiven und subtraktiven Fertigungsverfahren, den Produktionsanlagen und der Prozessautomatisierung sowie mit der ganzheitlichen Betrachtung und Optimierung der Produktionssysteme und -netzwerke. Hierbei werden jeweils technologische wie auch organisatorische Aspekte betrachtet.

Prof. Dr.-Ing. Jürgen Fleischer
Prof. Dr.-Ing. Gisela Lanza
Prof. Dr.-Ing. habil. Volker Schulze
Prof. Dr.-Ing. Frederik Zanger

KIT
Karlsruher Institut für Technologie

Vorwort des Verfassers

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am wbk Institut für Produktionstechnik des Karlsruher Instituts für Technologie (KIT). Mein besonderer Dank gilt Frau Prof. Dr.-Ing. Gisela Lanza als Hauptreferentin für ihr Vertrauen und ihre kontinuierliche fachliche und persönliche Unterstützung während der gesamten Zeit. Weiterhin bedanke ich mich bei Univ.-Prof. Dr.-Ing. Oliver Riedel für die Übernahme des Korreferats sowie Prof. Dr.-Ing. Rania Rayyes für den Prüfungsvorsitz.

Dem Karlsruhe House of Young Scientists (KHYS) des KIT danke ich für die Förderung meines Forschungsaufenthaltes am Massachusetts Institute of Technology (MIT) in Boston, USA. Ebenso gilt mein Dank Dr. Brian Anthony vom Department of Mechanical Engineering für die herzliche Gastfreundschaft und die wissenschaftlichen Diskussionen.

Meinen Projektpartnern aus dem Forschungsprojekt SDM4FZI spreche ich ebenfalls besonderen Dank aus. Ohne die Unterstützung sowie die intensiven fachlichen Diskussionen mit Hansjörg Tutsch, Wolfgang Artschwager, Alexander Dilg, Sven Spieckermann, Michel Bussmann und Constantin Enke wäre diese Dissertation in dieser Form nicht möglich gewesen.

Bei allen Kolleginnen und Kollegen am wbk bedanke ich mich für die außergewöhnliche Zusammenarbeit und das offene, kollegiale Miteinander. Besonderer Dank gilt hierbei Patrizia Gartner, Magnus Kandler, Julia Dvorak, Marco Wurster und Jork Groenewold. Darüber hinaus danke ich Ali Bilen und Finn Bail herzlich für das sorgfältige Korrekturlesen sowie die vielen wertvollen fachlichen Diskussionen. Mein besonderer Dank und Respekt gilt zudem allen Studierenden, die durch ihr Engagement und ihre Kreativität einen wesentlichen Beitrag zum Gelingen dieser Arbeit geleistet haben.

Meinen engsten Freunden danke ich für ihren Rückhalt während der Promotionszeit, insbesondere meinem besten Freund Axel, der mir stets Kraft und Unterstützung gegeben hat. Abschließend möchte ich mich bei meinem Bruder Alexander, meiner Schwägerin Alina sowie bei meinen Eltern Ute und Armin für ihre fortwährende Unterstützung und ihren bedingungslosen Rückhalt während meiner Promotion bedanken.

Karlsruhe, 02.02.2026

Sebastian Tim Behrendt

Abstract

This thesis presents a service-oriented approach to Production Planning and Control (PPC) tailored for reconfigurable manufacturing systems (RMS). It addresses the limitations of monolithic, tightly coupled PPC solutions in dynamic industrial environments, where heterogeneous data sources, evolving system capabilities, and volatile demand require adaptive integration and orchestration. The proposed architecture combines a Middleware for schema-aware data integration, a Service Registry for dynamic discovery, and a Petri net–based task-agnostic service orchestration engine. Large Language Models (LLMs) are leveraged for automated schema mapping, reducing effort and enabling rapid integration of new services. The system was validated across multiple benchmarks and case studies, demonstrating its capacity to compose, execute, and reconfigure complex PPC control loops with minimal human intervention. Quantitative results show improved automation in data integration and flexibility in service orchestration, while qualitative feedback confirms practical applicability in brownfield contexts. The findings contribute a formalized, scalable, and adaptable framework that advances PPC toward software-defined, fully reconfigurable manufacturing ecosystems.

Kurzfassung

Diese Dissertation präsentiert einen serviceorientierten Ansatz für die Produktionsplanung und -steuerung (PPS), der speziell auf rekonfigurierbare Produktionssysteme zugeschnitten ist. Sie adressiert die Einschränkungen monolithischer, eng gekoppelter PPS-Lösungen in dynamischen Industrieumgebungen, in denen heterogene Datenquellen, sich verändernde Systemfähigkeiten und volatile Nachfrage eine adaptive Integration und Orchestrierung erfordern. Die vorgeschlagene Architektur kombiniert eine Middleware für schemabasierte Datenintegration, ein Serviceregister für dynamische Auffindbarkeit sowie eine auf Petrinetzen basierende aufgabenagnostische Serviceorchestrierungen. Large Language Models (LLMs) werden für das automatisierte Schemamapping eingesetzt, um den Integrationsaufwand zu reduzieren und die schnelle Einbindung neuer Services zu ermöglichen. Das System wurde in verschiedenen industriellen Szenarien validiert und zeigte die Fähigkeit, komplexe PPS-Regelkreise mit minimalem menschlichem Eingriff zu komponieren, auszuführen und zu rekonfigurieren. Quantitative Ergebnisse belegen eine gesteigerte Automatisierung der Datenintegration und Flexibilität in der Service-Orchestrierung, während qualitative Rückmeldungen die praktische Anwendbarkeit in Brownfield-Kontexten bestätigen. Die Arbeit liefert damit einen formalisierten, skalierbaren und anpassungsfähigen Rahmen, der PPS in Richtung softwaredefinierter, vollständig rekonfigurierbarer Produktionsökosysteme weiterentwickelt.

Contents

Abbreviations

Abbreviation	Description
AAS	Asset Administration Shell
AGV	Automated Guided Vehicle
AML	AutomationML
API	Application Programming Interface
BPMN	Business Process Model and Notation
CD	Continuous Deployment
CI/CD	Continuous Integration and Continuous Deployment
CIM	Computer Integrated Manufacturing
CoT	Chain-of-Thought
CapEx	Capital Expenditure
CPN	Colored Petri net
CPS	Cyber-Physical Systems
CPPS	Cyber-Physical Production Systems
CRUD	Create, Read, Update, Delete
DAG	Directed Acyclic Graph
DCF	Discounted Cash Flow
DSR	Design Science Research Methodology
DMS	Dedicated Manufacturing System
DPP	Digital Product Passport
DT	Digital Twin
EAI	Enterprise Application Integration
EDA	Event-Driven Architecture
EDC	Eclipse Data Space Connector
EII	Enterprise Information Integration
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
ETL	Extract, Transform, Load
FMS	Flexible Manufacturing System
fn	False Negative
fp	False Positive
GAV	Global-as-View
GS	Global Schema
HTTP	Hypertext Transfer Protocol
HS	Homogeneous Schema

Abbreviation	Description
I4.0	Industry 4.0
IDTA	Industrial Digital Twin Association
IIoT	Industrial Internet of Things
IS	Information System
IT	Information Technology
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LAV	Local-as-View
LFGP	Learning Factory Global Production
LLM	Large Language Model
MDA	Model-driven Architecture
MES	Manufacturing Execution System
ML	Machine Learning
MRP	Material Requirements Planning
MSA	Microservices Architecture
MQTT	Message Queuing Telemetry Transport
MW	Middleware
NLP	Natural Language Processing
OE	Orchestration Engine
OPC UA	OPC Unified Architecture
OAS	OpenAPI Specification
OS	Operating System
OT	Operation Technology
PAL	Physical Abstraction Layer
PDDL	Planning Domain Definition Language
PLC	Programmable Logic Controller
PPC	Production Planning and Control
PPS	Produktionsplanung und -steuerung
PTP	Point-to-Point Integration
RAMI 4.0	Reference Architecture for Manufacturing Information Technology 4.0
REST	Representational State Transfer
RMS	Reconfigurable Manufacturing Systems
RPS	Robot Planning Service
RTDE	Real-Time Data Exchange Format
S	LLM prompt with only Schema

Abbreviation	Description
SAPN	Schema-aware Petri Net
SOA	Service-oriented Architecture
SDCM	Software-Defined Cloud Manufacturing
SDM	Software-defined Manufacturing
SDN	Software-defined Networking
SED	LLM prompt with Schema, Examples, and Description
SEDR	LLM prompt with Schema, Examples, Description, and Rules
SLADTA	Six-Layer Architecture for Digital Twins with Aggregation
SIS	Schema Integration Service
SOAP	Simple Object Access Protocol
SR	Service Registry
SSE	Server-Sent Events
SysML	Systems Modeling Language
tp	True Positive
TP	Throughput
TSN	Time-Sensitive Networking
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VUCA	Volatile, Uncertain, Complex, and Ambiguous
XML	Extensible Markup Language

Symbol	Description	Unit	Value range
$\mathcal{I}$	Data Integration System tuple $(\mathcal{G}, \mathcal{S}, \mathcal{M})$	-	-
$\mathcal{G}$	Global Schema	-	-
$\mathcal{S}$	Source Schema	-	-
$\mathcal{M}$	Mapping between global and source schemas	-	-
$\mathcal{L}_\mathcal{G}$	Language used to express the global schema	-	-
$\mathcal{A}_\mathcal{G}$	Alphabet of the global schema	-	-
$\mathcal{L}_\mathcal{S}$	Language used to express the source schema	-	-
$\mathcal{A}_\mathcal{S}$	Alphabet of the source schema	-	-
Svc	PPC Service tuple $(\mathcal{X}_{in}, \mathcal{Y}_{out}, f)$	-	-
$\mathcal{X}_{in}$	Formal input schema of a service	-	-
$\mathcal{Y}_{out}$	Formal output schema of a service	-	-
f	Service function mapping input to output	-	-
$I(S)$	Set of all valid data instances conforming to schema S	-	-
$\mathcal{W}$	Integration workflow	-	-
c	Connector function (resolves technical heterogeneity)	-	-
h	Formatter function (resolves syntactic heterogeneity)	-	-
m	Mapper function (resolves semantic heterogeneity)	-	-
Σ	Finite set of color sets (data types) in SAPN	-	-
P	Finite set of places in a Petri net	-	-
T	Finite set of transitions in a Petri net	-	-
A	Finite set of arcs in a Petri net	-	-
M_0	Initial marking of a Petri net	-	$\mathbb{N}^P$
F	Flow relation $F \subseteq (P \times T) \cup (T \times P)$	-	-
N	Node function $N : A \to (P \times T) \cup (T \times P)$	-	-
C	Color function $C : P \to \Sigma$	-	-
G	Guard function assigning expressions to transitions	-	-
E	Arc expression function	-	-
I	Initial marking function of SAPN	-	-
σ	Service signature function $\sigma : T \to (\Sigma_{in} \times \Sigma_{out})$	-	-
M	Marking function $M : P \to \mathbb{N}$ (tokens per place)	-	$\mathbb{N}_{\geq 0}$
W	Arc-weight function	-	$\mathbb{N}_{>0}$
$\bullet t$	Preset (input places) of transition t	-	-
$t^\bullet$	Postset (output places) of transition t	-	-
$\Sigma_{in}(t)$	Union of schemas of input tokens for transition t	-	-

Symbol	Description	Unit	Value range
$\Sigma_{out}(t)$	Schema of data produced by transition t	-	-
TP	True Positive count	-	$\mathbb{N}_{\geq 0}$
FP	False Positive count	-	$\mathbb{N}_{\geq 0}$
FN	False Negative count	-	$\mathbb{N}_{\geq 0}$

1 Introduction

1.1 Context and Relevance

Production Planning and Control (PPC) is a central function in manufacturing, coordinating material flows, capacity utilization, and order fulfillment to achieve delivery reliability, cost efficiency, and high asset productivity (Kellner & Lienland et al. 2020, p. 161; Schuh 2006, p. 43). In volatile, uncertain, complex, and ambiguous (VUCA) markets, PPC systems must cope with unpredictable demand fluctuations, shorter product life cycles, and increasing product variety (Gutenschwager & Rabe et al. 2017, p. 38). These dynamics place new challenges on both the organizational and technical design of PPC systems (Mönch & Fowler et al. 2013, p. 245; Zelewski & Hohmann et al. 2010, p. 19).

In recent years, the paradigm of Industry 4.0 (I4.0) has emerged as a key enabler for addressing these challenges (Wiendahl & Reichardt et al. 2023, p. 108). Through the integration of cyber-physical production systems (CPPS), ubiquitous connectivity, and real-time analytics, I4.0 fosters horizontally and vertically integrated production systems (Monostori 2014). In such systems, information flows seamlessly, enabling early detection of disruptions and rapid adaptation of processes (Stark & Fresemann et al. 2019). Smart factories, as envisioned by Kagermann & Wahlster et al. (2013), are characterized by products that are uniquely identifiable, aware of their state, and capable of autonomously interacting with production resources (Bauer 2017, p. 9).

Reconfigurable Manufacturing Systems (RMS) extend this vision by providing modular, integrable, and scalable production architectures that can rapidly adapt their functionality and capacity to changing requirements (ElMaraghy 2005; Wiendahl & ElMaraghy et al. 2007). By enabling the reconfiguration of hardware, control systems, and workflows, RMS provide the means to react to the dynamics of VUCA markets and assure reduced lifecycle costs (Wiendahl & Reichardt et al. 2023, p. 101; Koren & Gu et al. 2018). However, the potential of RMS can only be fully realized if the underlying PPC systems are equally adaptable (ElMaraghy & ElMaraghy et al. 2012).

1.2 Problem Statement and Motivation

Despite these technological advances in I4.0, a significant gap persists between the physical flexibility of modern production systems and the architectural rigidity of the software that governs them.

The software architectures that underpin many PPC systems remain monolithic, static, and poorly interoperable (Mönch & Fowler et al. 2013, p. 246; Wiendahl & Reichardt et al. 2023,

p. 155). Traditional solutions often rely on bespoke, point-to-point integrations between enterprise systems, such as ERP (Enterprise Resource Planning), MES (Manufacturing Execution System), and specialized planning tools, resulting in a brittle and costly interface landscape (Zelewski & Hohmann et al. 2010, p. 19; Babel 2024, p. 167).

Integration efforts are frequently limited to file formats and technical interfaces while the deeper semantic interoperability remains unresolved (Bueno & Godinho Filho et al. 2020). Consequently, significant manual effort is still required to interpret and transform data between systems, as evidenced by the widespread reliance on tools like Excel for critical planning tasks (Stark & Fresemann et al. 2019; Luo & Thevenin et al. 2023).

The consequences of these architectural limitations are substantial. Administrative overhead and manual data handling inflate operational costs, while delays in adapting PPC workflows hinder responsiveness to market changes (Kellner & Lienland et al. 2020, p. 170). Advanced planning algorithms, which are computationally intensive and essential for optimizing complex production systems, remain difficult to implement and maintain within these rigid environments (Luo & Thevenin et al. 2023). Unlike technology-agnostic or modular architectures, current PPC systems are tightly coupled and cannot easily accommodate new or previously unknown software components (Mönch & Fowler et al. 2013, p. 248).

The complexity and cost of integrating heterogeneous systems further limit the implementation of adaptive PPC, which is central to realizing the I4.0 vision (Bauernhansl & Krüger et al. 2016, p. 32; Elnadi & Abdallah 2024). Beyond technical integration efforts, recent research emphasizes that the growing volume and heterogeneity of data in smart manufacturing systems significantly increase the complexity of decision-making processes, requiring appropriate abstraction and human-centred support mechanisms (Cimini & Lagorio et al. 2022).

The limited flexibility of PPC systems is becoming more acute with the move toward CPPS and RMS. The rigid hierarchies of the traditional automation pyramid, where IT (Information Technology) and OT (Operational Technology) are structured in separated layers, dissolve to create dynamic networks of distributed, service-oriented components (Monostori 2014). This fundamental shift from hierarchical to networked control calls for a corresponding evolution in software architecture. Monolithic PPC systems must give way to modular, service-oriented systems, where planning and control functionalities are encapsulated in loosely coupled services that can be discovered, composed, and orchestrated at runtime to meet changing operational needs (Mönch & Fowler et al. 2013, p. 28; Zelewski & Hohmann et al. 2010, p. 866; Tao & Qi et al. 2018).

To address the long-standing challenge of integrating IT and OT systems, recent advances in *Large Language Models (LLMs)* offer a promising new frontier. LLMs have shown potential to

bridge syntactic and semantic gaps by generating mappings based on interface descriptions and data examples (Zhao & Zhou et al. 2025). This shows the potential to target one of the most persistent interoperability challenges in companies: the manual creation of mappings (Bauernhansl & Krüger et al. 2016, p. 18; Doan & Halevy et al. 2012, p. 121). This new capability, combined with Service-oriented Architecture (SOA), provides a pathway to creating truly adaptive and intelligent PPC systems for the future of manufacturing.

1.3 Objectives and Research Questions

The overarching objective of this thesis is to design, implement, and evaluate a service-oriented PPC framework that enables flexible, automated integration and orchestration in RMS.

From this objective, the following research questions are derived:

1. **Architectural Design:** How can a technologically consistent and holistic framework for service-oriented PPC be designed and realized for an industrial context?

2. **Semantic Integration:** How can the integration of heterogeneous information systems be automated to the greatest extent possible, enabling seamless, end-to-end information flows?

3. **Task-Agnostic Orchestration:** What principles and methods are required to define and orchestrate a reconfigurable and task-agnostic PPC system?

This thesis follows a design science research (DSR) methodology (Hevner & March et al. 2004) that combines conceptual development, technical implementation, and empirical validation. The work begins with the derivation of a reference architecture for service-oriented PPC, informed by the requirements of RMS, I4.0, and interoperability standards such as the Asset Administration Shell (AAS). Building on this conceptual foundation, an integration layer is developed that enables interoperability, service registration, dynamic discovery, and data integration workflows.

A key methodological element is the integration of automated mapping capabilities. An LLM-based pipeline is designed to generate executable mappings directly from interface descriptions and examples, reducing manual integration effort and addressing both syntactic and semantic levels of interoperability. For process model execution, an extended formalism building on Colored Petri nets (CPN) is developed for defining, validating, and executing task-agnostic PPC processes, ensuring that orchestration logic remains decoupled from specific application domains.

The proposed framework is evaluated through a combination of performance benchmarking and case studies. Benchmarks measure component-level characteristics such as mapping accuracy, orchestration latency, and scalability, while case studies demonstrate applicability in real-world production environments. This dual approach ensures that both the technical and practical relevance of the framework are assessed.

This thesis makes the following contributions:

1. **Architectural Blueprint:** A formalized, service-oriented PPC architecture tailored to the requirements of RMS and I4.0 integration.
2. **Semantic Integration Method:** A novel LLM-based approach for prompt-to-executable semantic data mapping, reducing manual integration effort.
3. **Orchestration Model:** A Petri net formalism for task-agnostic and schema-aware orchestration of PPC process models.
4. **Empirical Validation:** Evaluation of the framework in diverse industrial contexts, demonstrating reduced integration effort, improved flexibility, and enhanced responsiveness.

1.4 Structure of the Thesis

The thesis is organized into nine chapters, each building upon the previous to develop and validate the proposed framework. Chapter 1 introduces the research context, motivation, objectives, and contributions. Chapter 2 presents the theoretical foundations, defining the key concepts of PPC, RMS, I4.0, data integration, and SOA. Chapter 3 reviews the state of the art in industrial PPC architectures, data integration methods, and smart manufacturing approaches, identifying the research gaps that motivate this work. Chapter 4 then introduces the research methodology, detailing the DSR paradigm and the validation strategy applied throughout this thesis. Chapter 5 describes the design of the proposed service-oriented PPC framework, including the LLM-based integration method and the Petri net orchestration model. Chapter 6 details the technical implementation of the architecture, the Schema Integration Service (SIS), and the Orchestration Engine (OE). Chapter 7 presents the validation and evaluation of the framework through performance tests and case studies in real manufacturing environments. Chapter 8 discusses the results, limitations, and future research opportunities, while Chapter 9 concludes the thesis by summarizing the main findings and their implications.

2 Theoretical Foundations

This chapter establishes the conceptual basis for the research conducted in this thesis. It consolidates key theoretical principles and definitions from the domains of PPC, manufacturing paradigms, and digital manufacturing technologies. Beginning with the fundamentals of PPC (Section 2.1), it outlines the scope, objectives, and evolution of planning and control models, highlighting the implications for modern industrial contexts. Building on this, the concept of changeability and the paradigm of RMS are examined in Section 2.2 to frame the operational and economic rationale for adaptable manufacturing systems.

The chapter then extends to the enablers of digital manufacturing in Section 2.3, with a focus on the shift from hierarchical automation architectures towards distributed, interoperable information networks. Particular emphasis is placed on data integration challenges (Section 2.4) and the role of SOA in bridging OT and IT in Section 2.5.

2.1 Production Planning and Control: Concepts and Systems

This section introduces the fundamental concepts of PPC as the operational backbone of manufacturing enterprises. It outlines the scope, objectives, and core task structure of PPC in Section 2.1.1, and discusses in Section 2.1.2 how different PPC models structure decision-making across planning and execution. Building on this conceptual foundation, the divergence between PPC models and their realization in software-based PPC systems is elaborated in Section 2.1.3, highlighting the evolution of PPC systems and the resulting implications for flexibility and reconfigurability in I4.0.

2.1.1 Scope, Goals and Core Concepts of Production Planning and Control

PPC constitutes the core of any industrial enterprise, encompassing the entire domain of operational production management (Schuh 2006, p. 26). As a central task of operations management, it coordinates all production processes within an existing manufacturing system (Kellner & Lienland et al. 2020, p. 161). In its modern interpretation, the scope of PPC covers the entire technical order-fulfillment process, from sales and design through purchasing, manufacturing, and dispatch (Schuh 2006, p. 21).

The fundamental task of PPC is the quantitative, timely, and capacity-oriented planning and control of all manufacturing processes (Eversheim 2002, p. 43). This involves efficiently planning, steering, and, where necessary, correcting the production flow in consideration of customer demand, available capacities, and strategic objectives (Kellner & Lienland et al. 2020, p. 161). PPC inherently entails conflicting objectives, such as high capacity utilization, on-time delivery, low capital commitment in inventory, and low procurement costs (Kellner

& Lienland et al. 2020, p. 162). Resolving the trade-offs between these objectives in the presence of variable lead times and operational disturbances represents the central challenge of production management (Lödding 2016, p. 36; Zelewski & Hohmann et al. 2010, p. 16).

To manage this complexity, PPC is typically structured into hierarchical models with clearly separated tasks (Acker 2011, p. 10). Although solving PPC holistically would in principle yield better results, task decomposition into sequentially solved subproblems is considered essential in practice (Mönch & Fowler et al. 2013, pp. 245-246). Due to the computational complexity of these subproblems, manual processing is rarely feasible, and PPC models must be embedded in suitable information systems (IS), called *PPC systems* (Mönch & Fowler et al. 2013, pp. 245-246).

The main tasks of PPC, according to the model of Acker (2011), are visualized in Figure 2.1. The model distinguishes *Production Planning (PP)*, which creates the plans necessary for production through tasks such as production program planning, material requirements planning (MRP), rough-cut scheduling, and capacity planning, from *Production Control (PC)*, which targets their execution through order release, detailed scheduling, and progress control. The results of each task are passed in a top-down fashion to the subsequent task. It should be noted that no consensus exists in the literature on the scope, objectives, and sequence of PPC tasks (Zelewski & Hohmann et al. 2010, p. 225); the model of Acker (2011) therefore serves as a reference for later discussions rather than a normative scheme.

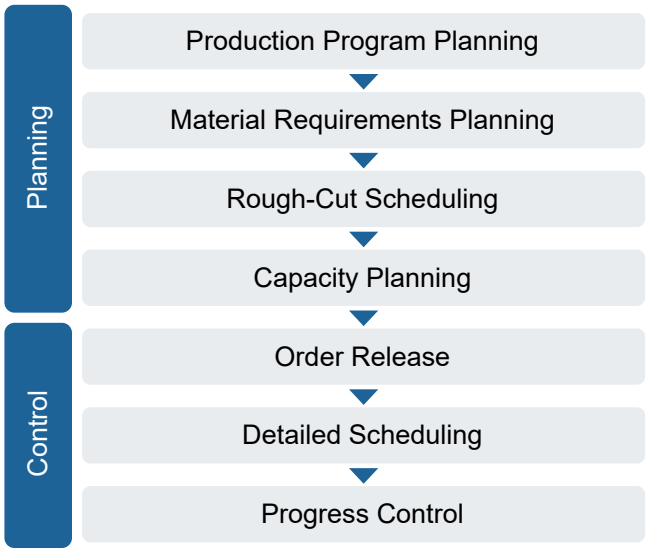

Figure 2.1: Main tasks of PPC based on Acker 2011 with reference of (Hackstein 1984)

2.1.2 Classification and Delimitation of Production Planning and Control Models

A common denominator across most PPC models is the conceptual separation of PP and PC: PP defines future-oriented target decisions, whereas PC ensures their operational realization by guiding and monitoring shop-floor execution (Dyckhoff & Spengler 2007, p. 36). The *order release* function typically constitutes the key interface between both domains, linking the abstract planning world with the realities of execution (Meudt & Wonnemann et al. 2017, p. 10). Beyond this shared principle, however, PPC models differ considerably in scope, granularity, and sequencing, reflecting heterogeneous manufacturing contexts and planning philosophies (Zelewski & Hohmann et al. 2010, p. 225). PPC models are consequently best understood as generic, abstract frameworks that deliberately omit case-specific details and must be tailored to the specific context of an organization (Zelewski & Hohmann et al. 2010, p. 16).

The evolution of PPC models is closely tied to the development of supporting IS, as illustrated in Figure 2.2. Early MRP approaches centered on material need calculation evolved into MRP II, which integrated personnel and machine capacities, and further into ERP systems that cover all enterprise resources beyond direct production (Bauer 2017, pp. 5, 318). To address the planning shortfalls of ERP systems (Mönch & Fowler et al. 2013, p. 28), specialized Advanced Planning and Scheduling (APS) systems employ Operations Research methods to optimize plans against multiple finite resource constraints (Kellner & Lienland et al. 2020, p. 318). Across these generations, communication and IT technologies typically serve as enabling technologies adopted from broader IT developments with some delay (Bauernhansl & Krüger et al. 2016).

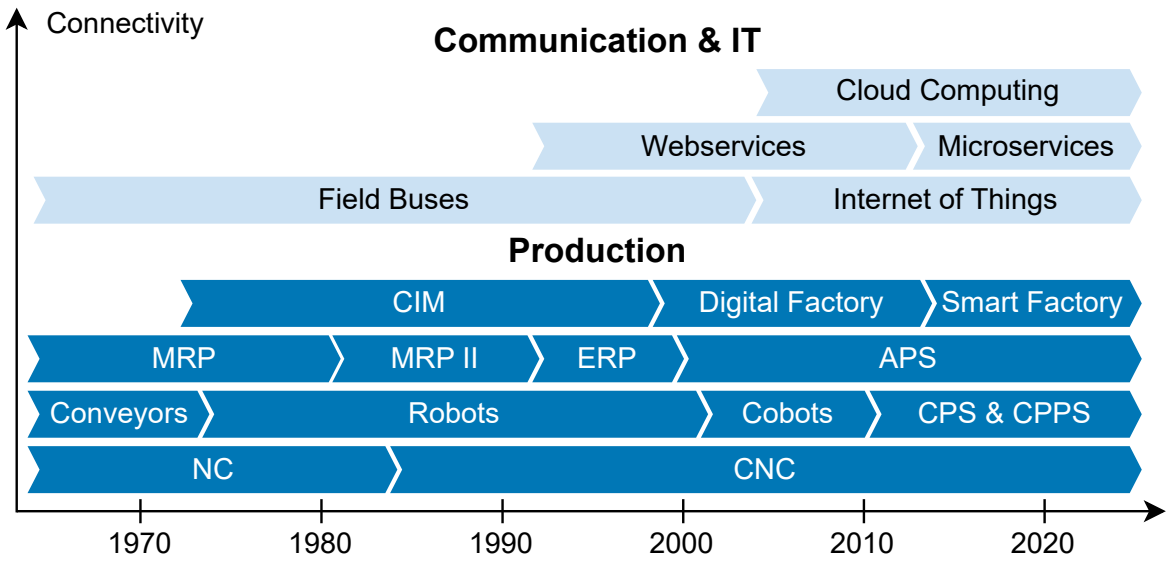

Figure 2.2: Historical development of production, PPC and IT technologies based on Bauernhansl & Krüger et al. (2016) and Babel (2024)

To systematically compare PPC models, Zelewski & Hohmann et al. (2010, p. 230) propose a multi-dimensional typology covering, among others, the degree of centralization, the hierarchical structure, the coordination focus (inventory- vs. capacity-oriented), and the scope (*complete* vs. *focusing* models, with the latter specializing in specific sub-tasks of PPC). Traditional MRP-II-based models are archetypal examples of centralized, top-down, hierarchical-sequential approaches, in which the output of one module becomes a fixed input for the next (Acker 2011; Zelewski & Hohmann et al. 2010). A meta-study of twenty PPC models developed between 1970 and 2014 confirms that most can be characterized as hierarchical-sequential frameworks, with a strong emphasis on the planning phase while control is often reduced to order release and monitoring (Meudt & Wonnemann et al. 2017).

A particularly relevant delimitation among PPC models concerns their handling of *feedback*. The strictly sequential nature of many hierarchical models makes the integration of feedback loops difficult (Schuh 2006, p. 57). While operational control inherently includes monitoring and corrective actions, this feedback often does not automatically propagate back to alter higher-level production plans. This is a critical weakness, as real-world production is fraught with frequent and unpredictable disruptions such as machine breakdowns or material delays (Zelewski & Hohmann et al. 2010, p. 229). In contrast, approaches, such as the one proposed by Lödding (2016), explicitly model the control process as a closed loop with defined control variables and target values, aiming to react more dynamically to shop floor events. The ability to systematically incorporate feedback across planning and control layers therefore emerges as a key requirement for modern PPC systems.

2.1.3 From Concept to Software: Models vs. Systems

The development and introduction of PPC systems is one of the most complex tasks in PPC (Schuh 2006, p. 32). The primary challenge is the difficulty of translating abstract, generic PPC models into concrete, operational PPC systems (Zelewski & Hohmann et al. 2010, p. 18). A *PPC system* thus bridges the “implementation gap” between an abstract PPC model and its practical application, creating a mutual dependency between the two: every system rests on one or more underlying models, and conversely, models rely on systems for their implementation (Zelewski & Hohmann et al. 2010, pp. 17–18).

This translation is hindered by several factors. First, the conceptual vagueness and inconsistent terminology in the literature complicate the comparison, selection, and application of a suitable model, which is why PPC models are rarely implemented in their pure form (Zelewski & Hohmann et al. 2010; Meudt & Wonnemann et al. 2017, p. 18). Instead, components are often omitted or blended with elements from other, potentially incompatible, concepts, which can lead to conceptual inconsistencies and suboptimal performance. Second, the sheer

complexity of the production environment, with its numerous interdependencies and high degree of uncertainty, requires planning and control logic to be efficiently adapted to changing strategies (Acker 2011, p. 9). Current implementations, however, are predominantly monolithic applications with rigid structures that hinder this flexibility, motivating a trend towards more open, modular, and service-oriented architectures for PPC functionality (Mönch & Fowler et al. 2013, p. 248).

The trend to focus on flexibility and reconfigurability of production systems amplifies the need for flexible and reconfigurable PPC systems. In the following section, the requirements and design principles of flexible and reconfigurable manufacturing will be explored to identify further requirements for PPC systems.

2.2 Changeability and Reconfigurable Manufacturing Systems

This section introduces the concept of *changeability* as a key paradigm for coping with volatile markets and increasing system complexity in modern manufacturing. First, Section 2.2.1 establishes the conceptual foundations by distinguishing types of changeability and their role in structuring adaptation capabilities across different manufacturing levels. Building on this terminology, Section 2.2.2 focuses on RMS, introducing their defining characteristics, design principles, and the resulting implications for PPC on both physical and logical levels. Finally, Section 2.2.3 discusses the economic motivation for RMS, highlighting why the long-term value of reconfigurability depends on its effective utilization and on suitable PPC systems.

2.2.1 Conceptual Foundations: Changeability, Flexibility, Reconfigurability

Modern manufacturing is shaped by frequent changes in requirements (ElMaraghy 2009), making the ability to adapt a critical factor for competitive success (Wiendahl & ElMaraghy et al. 2007; Bauernhansl 2023). This necessity has given rise to the paradigm of changeable manufacturing. The term *changeability* serves in this paradigm as an umbrella concept, embodying a range of capabilities such as flexibility, reconfigurability, transformability, and agility, each corresponding to different levels within a manufacturing organization, as visualized in Figure 2.3 (ElMaraghy & Monostori et al. 2021). At its core, Wiendahl & ElMaraghy et al. (2007) define changeability as the characteristic “[...] to accomplish early and foresighted adjustments of the factory’s structures and processes on all levels to change impulses economically”.

The drivers for changeability arise from the interplay of volatile market requirements and the structural limitations of traditional manufacturing systems. Increasing demand volatility and a growing variety of product variants require production systems to react faster and more frequently to changing product mixes and volumes (Wiendahl & ElMaraghy et al. 2007),

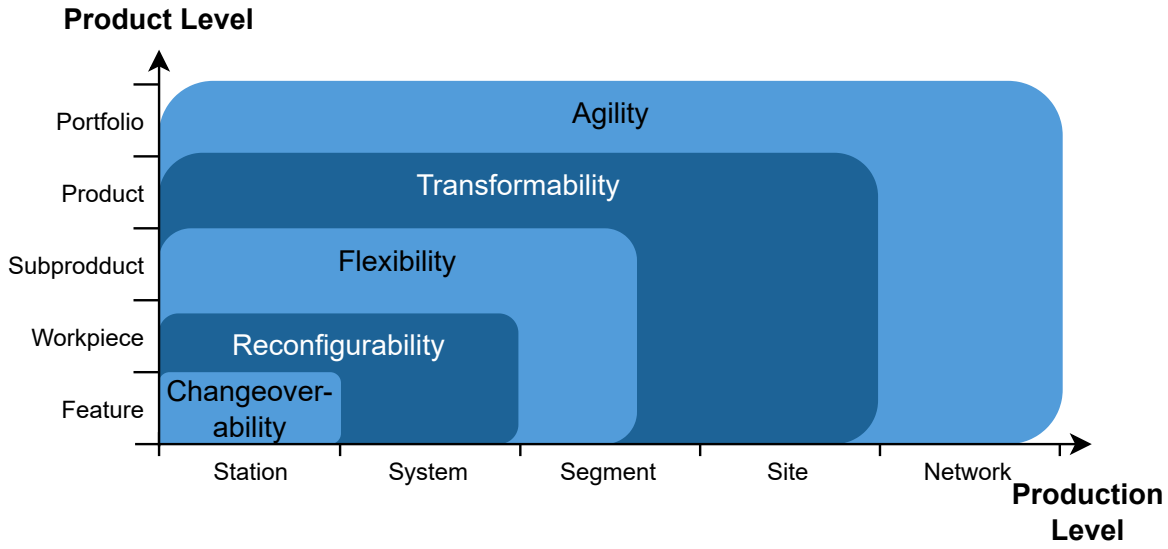

Figure 2.3: Overview of the classes of Changeability with their influence on product and production level based on Wiendahl & ElMaraghy et al. (2007)

while legacy factories with their rigid hardware configurations and tightly coupled software landscapes make adjustments costly and slow (Bauernhansl 2023, p. 46). Modular and compatible components reduce the effort of adaptations and thereby lower planning and reconfiguration costs (ElMaraghy & ElMaraghy et al. 2012; Bortolini & Galizia et al. 2018).

In the context of PPC, changeability becomes operational through the ability to adapt plans and decisions under changing conditions. From a planning and control perspective, this capability is primarily realized through two constituents: **flexibility** and **reconfigurability** (ElMaraghy 2005, p. 265).

Flexibility refers to the “tactical ability of an entire production and logistics area to switch with reasonably little time and effort to new – although similar – families of components by changing manufacturing processes, material flows and logistical functions” (Wiendahl & ElMaraghy et al. 2007) and is built into the system *a priori* to anticipate a range of potential variations, as realized in Flexible Manufacturing Systems (FMS). Reconfigurability, in contrast, denotes the “operative ability of a manufacturing system to switch with minimal effort and delay to a particular family of workpieces or subassemblies through the addition or removal of functional elements” (Wiendahl & ElMaraghy et al. 2007). The latter notion reflects the evolution from dedicated, high-volume lines (Dedicated Manufacturing Systems (DMS)) through FMS towards RMS (Wiendahl & ElMaraghy et al. 2007), each paradigm addressing different needs for variety and responsiveness.

2.2.2 Reconfigurable Manufacturing Systems

RMS were introduced to enable rapid and cost-efficient adjustment of both production capacity and functionality in response to unforeseen or uncertain market changes (Mehrabi & Ulsoy et al. 2000; Koren & Gu et al. 2018). In contrast to FMS, which provide a predefined flexibility range *a priori*, RMS deliver exactly the required capacity and functionality for a targeted product family and allow modifying this operating point through reconfiguration when requirements evolve (ElMaraghy 2005). This enables a “live” factory architecture in which machines and material handling units can be added, removed, or modified with limited effort, extending the useful life of the manufacturing system (Koren & Gu et al. 2018). As illustrated in Figure 2.4, reconfigurability allows an RMS to drastically shift its operating point in both functionality and capacity, making it particularly suitable for volatile and uncertain market conditions.

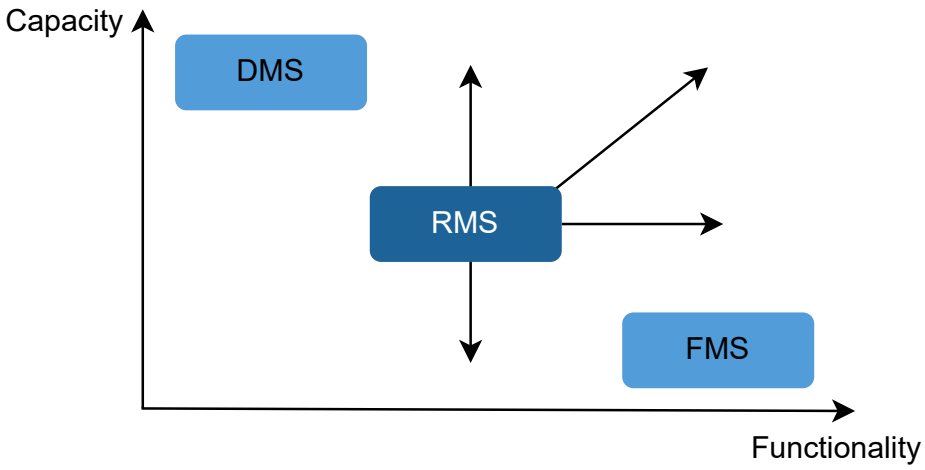

Figure 2.4: Mapping of functionality and capacity of different types of manufacturing systems based on Mehrabi & Ulsoy et al. (2000)

To achieve reconfigurability, an RMS must realize six design principles – *modularity*, *integrability*, *convertibility*, *diagnosability*, *customization*, and *scalability* – as identified by ElMaraghy (2005), Koren & Gu et al. (2018), and Mehrabi & Ulsoy et al. (2000) and visualized in Figure 2.5. *Modularity* ensures that system components are designed as independent and interchangeable modules, while *integrability* guarantees that components can be readily integrated and extended with new technologies. *Convertibility* enables quick changeovers between existing products, *diagnosability* supports the rapid identification of quality and reliability problems, and *customization* matches the system’s capability to the specific application to avoid the cost of unnecessary flexibility. Finally, *scalability* enables incremental and economical changes of system capacity.

Consideration of these design principles for both software and hardware is a prerequisite to realize efficient reconfigurations. In general, reconfigurations of an RMS can be categorized

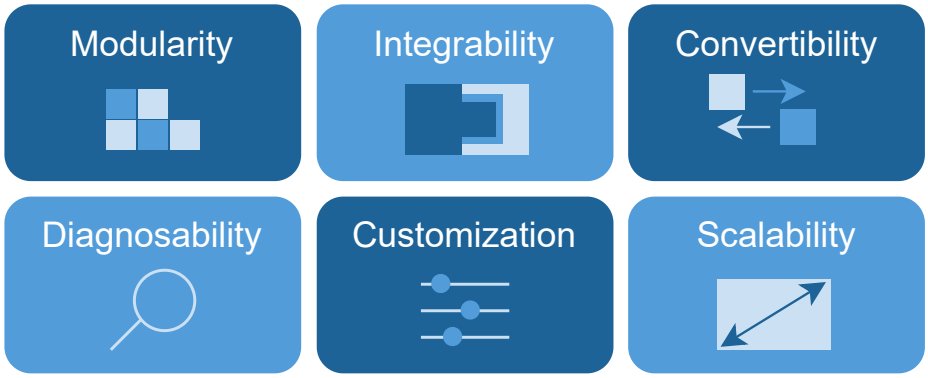

Figure 2.5: Overview of the design principles of RMS based on ElMaraghy (2005)

as *physical* (hard) changes – adding, modifying, removing, or interchanging machine components, machines, cells, or material handling units – and *logical* (soft) changes that adapt the controlling software systems (Koren & Heisel et al. 1999; ElMaraghy 2005).

Crucially, a physical reconfiguration is ineffective without a corresponding reconfigurability on the logical level. Since fixed capacities and rigid routings are overhauled with every reconfiguration, classical PPC tasks are fundamentally altered. This gives rise to *Adaptive PPC*, which covers the capability to dynamically adjust plans and control policies in response to changes in product volume, mix, or the production system configuration (Wiendahl & ElMaraghy et al. 2007). Consequently, the functional coverage required of a PPC system in an RMS is enlarged: capacity becomes a planning variable rather than a fixed constraint, layouts and routings must be adapted dynamically, and schedules require rapid re-computation to reflect new system states (ElMaraghy 2005). These *replanning* and *reprogramming* activities at the logical level require the PPC system itself to be designed according to the same RMS design principles, enabling it to co-evolve with the physical manufacturing system it controls (Wiendahl & ElMaraghy et al. 2007).

2.2.3 Economic Rationale for Reconfigurable Manufacturing Systems

The fundamental economic driver for RMS is the need for manufacturing enterprises to respond cost-effectively to market volatility and unforeseen changes in product demand (Koren & Gu et al. 2018). Unlike traditional manufacturing paradigms, the economic justification for RMS is not based on minimizing initial capital expenditure, but on optimizing the total cost of ownership over the system's entire life-cycle (Koren & Heisel et al. 1999). While FMS often require high initial capital investment to embed flexibility for a wide range of anticipated product variations – flexibility that is sometimes underutilized (Wiendahl & ElMaraghy et al. 2007; Koren & Gu et al. 2018) – RMS follow the economic principle of *economies of scope*, in which the factory adapts cost-effectively to changing requirements and supplies products at a

competitive price for many years beyond its initial design (Wiendahl & ElMaraghy et al. 2007; Koren & Gu et al. 2018; ElMaraghy 2005; Nyhuis & Hübner et al. 2017).

The economic viability of RMS is directly tied to its technical enablers: *scalability* allows capacity to be changed incrementally in response to market shifts, while *convertibility* and *integrability* reduce the time and cost required to launch new products or reconfigure the system (Koren & Gu et al. 2018; ElMaraghy 2005). Logical adjustments are particularly attractive economically, provided that the underlying PPC systems offer the required reconfigurability (ElMaraghy 2009). Bortolini & Galizia et al. (2018) quantitatively confirm this rationale through a Net Present Value analysis: RMS outperform FMS and DMS by requiring a lower level of investment to manage equivalent production complexity, but only if the reconfigurability is actively utilized. Sensitivity analyses show that without sufficiently frequent reconfiguration, the financial performance of RMS becomes comparable to that of DMS or FMS (Bortolini & Galizia et al. 2018). The economic potential of RMS therefore materializes only when its technical enablers are fulfilled and effectively leveraged through the controlling PPC systems.

While the necessary technological building blocks for RMS, i.e. modular manufacturing equipment on the hardware side and intelligent digitization powered by I4.0 on the software side, are nowadays available (Bortolini & Galizia et al. 2018), many PPC systems in industrial practice still do not sufficiently support fast and cost-efficient logical reconfigurations (Zelewski & Hohmann et al. 2010). Crucial planning functions such as integrated capacity and layout adaptation are often only partially covered or require substantial manual effort, which limits the practical applicability and economic potential of RMS due to unmet RMS design principles at the software level (Koren & Gu et al. 2018; Zelewski & Hohmann et al. 2010). The following sections therefore explore the digital enablers that provide the necessary foundations for realizing a PPC system that suits the vision of RMS.

2.3 Digital Manufacturing

This section introduces the technological foundations for realizing interconnected, data-driven, and adaptive production systems. Section 2.3.1 first summarizes the core concepts of I4.0 and its key components, highlighting the role of CPPS and interoperability. Building on this, Section 2.3.2 discusses the transition from the hierarchical automation pyramid towards distributed information networks and service-oriented interactions. Section 2.3.3 then outlines relevant communication standards and paradigms that enable vertical and horizontal integration across OT, IIoT, and IT domains. Finally, Section 2.3.4 reviews industrial data model standards and modeling languages as prerequisites for semantic interoperability and scalable data integration in manufacturing.

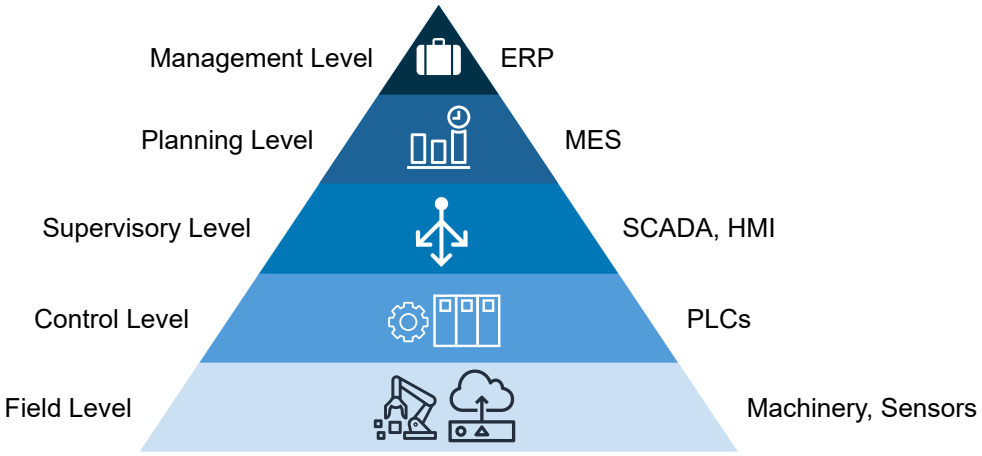

Figure 2.6: Overview of the automation pyramid with its levels and technologies based on Babel (2024)

2.3.1 Industry 4.0: Concepts and Components

I4.0 represents the current, fourth industrial revolution and constitutes a new paradigm for organizing and controlling manufacturing operations across the entire value chain to meet increasingly individualized customer demands (Bauernhansl & Krüger et al. 2016; Heidel & Hoffmeister et al. 2017, p. 12). In contrast to earlier initiatives such as Computer Integrated Manufacturing (CIM), which were constrained by rigid centralized control (Bauernhansl & Krüger et al. 2016, p. 7), more recent paradigms related to I4.0 – including Smart Manufacturing (Kusiak 2018), Cloud Manufacturing (Ren & Zhang et al. 2017), the Digital Twin (DT) (Kritzinger & Karner et al. 2018), and the Industrial Internet of Things (IIoT) (Boyes & Hallaq et al. 2018) – emphasize a deeper and more automated IT integration and a virtual representation of physical assets in manufacturing (Monostori 2014, pp. 11–12).

The technological foundation of I4.0 is the convergence of the physical and digital worlds through Cyber-Physical Systems (CPS) and their application in manufacturing as CPPS (Mönch & Fowler et al. 2013, p. 9; Monostori 2014). Unlike their mechatronic predecessors, CPPS are network-connected, enabling them to form intelligent networks that monitor physical processes, create virtual representations with DTs, and, crucially, make decentralized decisions (Bauernhansl & Krüger et al. 2016, p. 10). Realizing this vision requires comprehensive *interoperability*, defined as the ability of heterogeneous systems to connect, communicate, and utilize information seamlessly across the value chain (Bauernhansl 2023, p. 297).

2.3.2 From the Automation Pyramid to a Distributed Information Network

The traditional, hierarchical automation pyramid with its layers from the field level to the business level, as depicted in Figure 2.6, is ill-suited for the demands of I4.0 for flexibility and

decentralized intelligence (Bauernhansl 2023, p. 46). In the automation pyramid communication is typically rigid, flowing vertically between adjacent levels and horizontally within a single level (for more information about the automation pyramid, see Appendix A1) (Boyes & Hallaq et al. 2018, p. 9, p.216). This structure limits the necessary horizontal and vertical integration of I4.0, leading to its dissolution in favor of decentralized, service-oriented networks of intelligent components (Bauernhansl & Krüger et al. 2016). This is particularly important in the context of PPC, as fast and continuous information flows between shop floor and PPC systems are essential for effective planning and control.

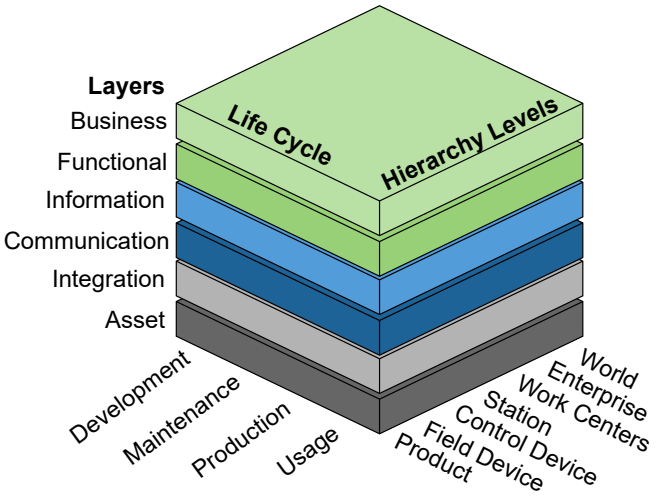

Figure 2.7: RAMI4.0 visualization based on Babel (2024) with viewpoints of different tasks in manufacturing.

This paradigm shift is formalized by reference architectures such as RAMI 4.0 (depicted in Figure 2.7), which replaces the one-dimensional hierarchy with a three-dimensional map (Hierarchy Levels, Life Cycle & Value Stream, and Layers), allowing complex functions and components such as PPC or CPPS to be modeled across multiple levels (Babel 2024, p. 468). PPC in CPPS therefore cannot reside on a single hierarchy layer; instead, its sub-tasks such as PP or PC span multiple Layers, Life Cycle Steps, and Hierarchy Levels (Monostori 2014). Realizing such cross-cutting functionality requires a decentralized, service-oriented approach that stands in conflict with the monolithic designs of classical PPC systems, that lack support for dynamic reconfiguration and distributed decision-making (Mönch & Fowler et al. 2013; Monostori 2014). SOA and IIoT are therefore key enablers for this transition, offering the modularity and flexibility needed to realize the vision of I4.0 (Boyes & Hallaq et al. 2018).

2.3.3 Communication Standards in Industry 4.0

The I4.0 paradigm is driven by the convergence of OT and IT, which requires seamless communication from the shop floor to the enterprise cloud (Stark & Fresemann et al. 2019). To address the diverse requirements across these domains, communication standards can be broadly categorized based on their primary application area: OT, IIoT, and IT, as visualized in Figure 2.8.

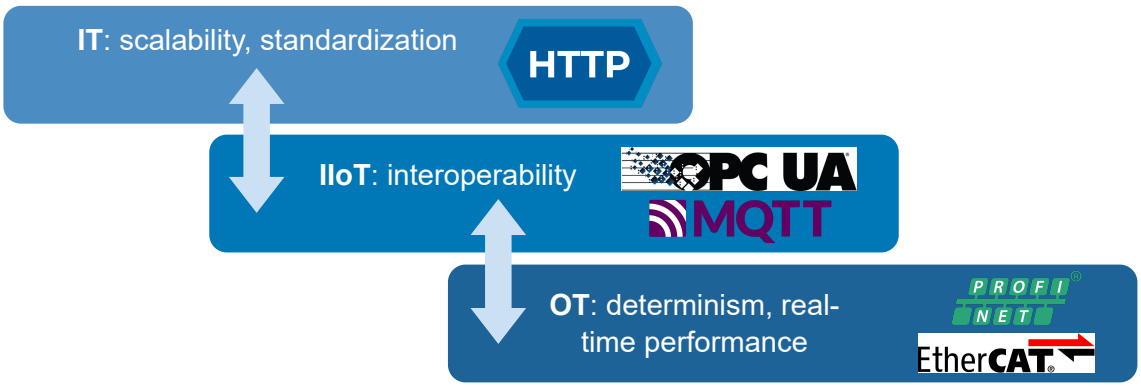

Figure 2.8: OT, IIoT, and IT communication standards with their priorities and exemplary technologies.

In the OT domain, communication is dominated by real-time fieldbus systems and Industrial Ethernet protocols such as PROFINET or EtherCAT, which are purpose-built for deterministic, low-latency, and highly reliable data exchange between sensors, actuators, and Programmable Logic Controllers (PLCs) (Babel 2024, p. 128). While essential for operational tasks, their specialized nature poses challenges for direct vertical integration with higher-level systems (Lechler & Schlechtendahl 2023). To bridge this OT–IT divide, IIoT protocols provide more flexible, platform-independent, and scalable data exchange. Two de facto standards are central here: *MQTT*, a lightweight publish/subscribe protocol especially suited for telemetry data from a large number of sensors and constrained devices (Hunkeler & Truong et al. 2008), and *OPC UA*, a platform-independent, service-oriented architecture with powerful information modeling capabilities that supports both a client-server binary protocol and an MQTT-based publish-subscribe profile (Mahnke & Leitner et al. 2009; Babel 2024, pp. 37, 191, 467). At the enterprise IT level, communication is typically realized through APIs based on standardized web protocols such as HTTP (Wang & Keivanloo et al. 2014), which are highly scalable, platform-independent, and serve as a key enabler for loosely coupled, service-oriented architectures (see Section 2.5).

The OT–IIoT–IT layering is essential for achieving vertical integration, as IIoT standards act as a crucial intermediary that translates the high-frequency, low-context data of the OT world

into a standardized, information-rich format consumable by IT systems (Boyes & Hallaq et al. 2018, p. 9).

2.3.3.1 Communication Paradigms: Publish/Subscribe vs. Request/Response

The technologies discussed above are built on two fundamentally different communication paradigms. The **request-response** model is a tightly coupled, synchronous pattern where a client sends a request to a specific server and waits for a response, as characteristic of web services and the client-server style of OPC UA (Wang & Keivanloo et al. 2014). In contrast, the **publish/subscribe (pub/sub)** model is a loosely coupled, data-centric communication style in which publishers send messages to logical channels and a broker distributes them to subscribers (Hunkeler & Truong et al. 2008). This approach, exemplified by MQTT, decouples participants in both space and time and is especially suited for streaming applications with low latency requirements. The choice between these paradigms fundamentally influences the architecture's degree of coupling, scalability, network load, and resilience (Papazoglou & Van Den Heuvel 2006). Modern industrial systems consequently leverage both patterns to meet diverse communication requirements, from direct machine control to large-scale data aggregation.

2.3.4 Standards for Industrial Interoperability

The integration of digital technologies across the manufacturing landscape has led to a proliferation of heterogeneous data environments. These environments are characterized by a wide variety of data formats, including text files (e.g., CSV, XML, JSON) or CAD formats (e.g., STEP) (Gutenschwager & Rabe et al. 2017, pp. 229–230). Linking them remains a significant challenge (Stark & Fresemann et al. 2019), often requiring considerable manual effort to develop specialized modules for translating and transforming data between different formats (Malik & Sharma et al. 2021, p. 125). To overcome these challenges, standardized frameworks address not only syntactical but also semantic aspects of data exchange (Bauernhansl 2023, pp. 297–298). These standards can be broadly categorized into two groups: modeling languages that define structure and syntax, and domain models and dictionaries that provide semantic meaning.

Modeling Languages. In industrial software and engineering contexts, formal modeling languages provide the syntactical and structural foundation for describing assets and processes, enabling the creation of consistent and machine-readable models that can be exchanged between different engineering tools and software platforms. Three modeling concepts are particularly relevant in the I4.0 context.

OPC UA supports information modeling by defining an address space that represents assets, their properties, and relationships as a structured graph of nodes – ranging from simple variables and objects to externally callable methods – interconnected through references (Mahnke & Leitner et al. 2009). *AutomationML (AML)* is a standard designed as an integration format for engineering data that supports the storage of plant topology, geometry, kinematics, and logic by building on existing standards such as CAEX, COLLADA, and PLCopen (Drath & Luder et al. 2008). The more recent *AAS* creates a digital counterpart of physical assets based on a metamodel, serialization, and an API (Industrial Digital Twin Association 2023a; Industrial Digital Twin Association 2023b). As summarized in Figure 2.9, an AAS is always linked to an Asset and contains multiple *Submodels*, which in turn are composed of *SubmodelElements*. Depending on the required expressiveness, submodel elements can represent primitive values (*Property*), structured objects (*SubmodelElementCollection*), lists (*SubmodelElementList*), or references to other model entities (*ReferenceElement*). This composition principle forms the basis for serializing the data and exchanging it via a standardized API, providing generic access to asset data independent of vendor-specific data formats.

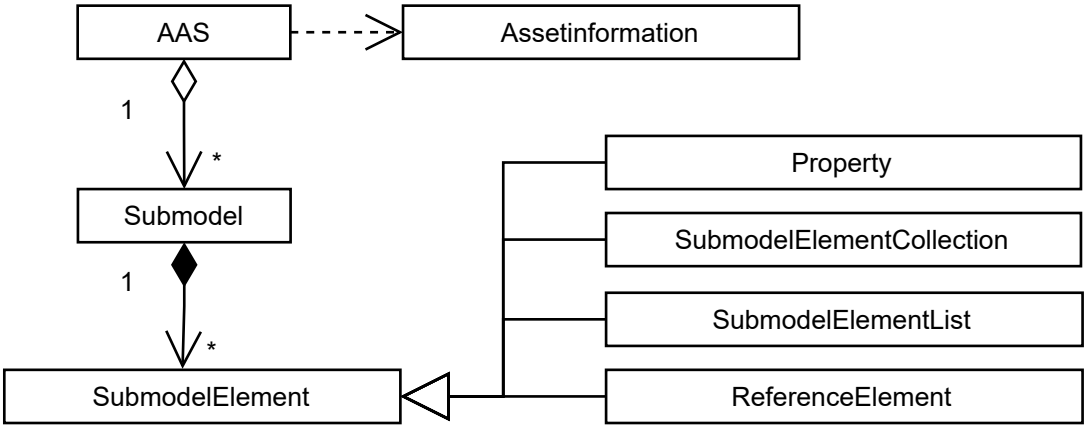

Figure 2.9: Overview of the most important AAS metamodel building blocks based on a UML definition (Industrial Digital Twin Association 2023a)

Although AML, OPC UA information models, and AAS overlap in their modeling scope, the three governing standardization organizations have delineated complementary roles: AML focuses on engineering data exchange with static and descriptive data, OPC UA on operational communication with dynamic real-time data, and AAS on digital lifecycle records that act as a self-describing information hub (Drath & Mosch et al. 2023). However, inherent differences in their modeling paradigms limit seamless integration and transformation between them (Alexopoulos & Kalogeras et al. 2025).

Domain Models and Dictionaries. The most common approach to achieve interoperable representation of data is by using domain models and dictionaries that establish a common

vocabulary and semantics within a particular field of application (Bauernhansl 2023, p. 297; Stark & Fresemann et al. 2019). They operationalize the modeling languages described above and define the precise meaning of data, ensuring that different systems interpret the data consistently and integration efforts are reduced (Abiteboul & Hull et al. 1996, p. 22; Bernstein & Haas 2008, p. 77). A large number of such standards exists in the I4.0 domain (Bastos & Sguario Coelho De Andrade et al. 2021); representative examples are summarized in Table 2.1. These standards can be grouped into *dictionaries* – such as ECLASS or the IEC Common Data Dictionary – which precisely define technical terms in a standardized way to align semantics, and *domain models* – such as IO-Link, MTConnect, or ISA-95 (IEC 62264) – which standardize how data should be structured to describe specific domain-related information.

Table 2.1: Overview of selected standards for manufacturing data integration and interoperability.

Standard	**Core Functionality**
IEC 62264 / ISA-95 (2013)	Defines hierarchical models and standardized information structures for integrating business (ERP) and production (MES, control) systems.
ECLASS (2024)	Provides standardized classes and property lists for describing products and materials across industries.
IEC 61360 (IEC CDD) (2024)	Common Data Dictionary that offers globally unique identifiers and standardized definitions for component characteristics and attributes.
MTConnect (2023)	Defines an open XML/JSON-based protocol for exchanging machine and equipment status and performance data.
IO-Link (IEC 61131-9) (2023)	Specifies a standardized digital communication interface for smart sensors and actuators, supporting parameterization and diagnostics.

Modern interoperability frameworks aim to leverage these domain data models. For instance, OPC UA Companion Specifications (Mahnke & Leitner et al. 2009) standardize information models for specific domains or devices, and AAS Submodel Templates, standardized by the Industrial Digital Twin Association (IDTA), provide pre-defined, domain-specific structures for AAS data. Since these standards are generated independently across organizations, they are not natively interoperable and require additional integration efforts between them (Alexopoulos & Kalogeras et al. 2025). The long-term objective is that manufacturing equipment is delivered with standardized digital descriptions, enabling plug-and-produce integration as a crucial aspect of RMS (Baheti & Gill 2011; Bortolini & Galizia et al. 2018). While this vision is gradually being realized – some industrial sensors already provide standardized IO-Link

or AAS interface descriptions – AAS adoption among equipment vendors remains at an early stage, and most devices still require manual or semi-automatic generation of digital representations (Alexopoulos & Kalogeras et al. 2025).

Despite this landscape of standards, a single de facto standard for PPC data has yet to emerge (Stark & Fresemann et al. 2019). In the area of MES and ERP integration, ISA-95 is the most established standard (Scholten 2007), yet it is often considered insufficient for real-time, data-intensive applications and is based on the rigid hierarchical structure of the automation pyramid. In practice, the data sources of manufacturing companies remain highly heterogeneous and customized, and the fragmentation of data formats and models is a primary barrier to creating interoperable information networks. This highlights the critical need for robust data integration strategies, which will be explored in the following section.

2.4 Data Integration and Interoperability Challenges

Achieving interoperability is a prerequisite for effective PPC in I4.0 environments, as production systems increasingly rely on digital models and integrated systems (Dumitrescu & Albers et al. 2021). The manufacturing landscape, however, is fragmented by a multitude of communication standards, data formats, and domain models, which presents a formidable integration challenge (Monostori 2014). This section confronts this challenge, beginning with a systematic categorization of data heterogeneity. It then formalizes the integration problem, outlines the data integration process, and explores the architectural patterns designed to create a unified and coherent view of enterprise data.

2.4.1 Categorization of Data Heterogeneity

While modern production environments generate a vast amount of data from diverse sources, including MES, ERP systems, and machinery on the shop floor, this data is often trapped in isolated silos, limiting vertical and horizontal integration (Stark & Fresemann et al. 2019; Kritzinger & Karner et al. 2018). Overcoming these silos requires complex transformation processes, known as *data integration*. It is defined as the process of combining data from disparate sources to provide users with a unified view (Hull 1997), with the primary goal of achieving interoperability (Bauernhansl 2023, p. 297). Failing to manage this challenge results in convoluted, expensive, and brittle point-to-point architectures that hinder building business processes economically with IS (Erl 2016, pp. 56, 106).

The fundamental obstacle to interoperability is *data heterogeneity* (Hull 1997, p. 51), which spans the full range of differences among systems used to model and physically represent data, from underlying hardware and software platforms to the data models and schemas used to structure information (Batini & Scannapieco 2006, p. 26). Heterogeneity is therefore

not a problem to be eliminated but a fundamental reality of modern IT landscapes that must be managed (Krafzig & Banke et al. 2004, p. 75). The literature distinguishes between three main categories of heterogeneity – technical, syntactic, and semantic – as visualized in Figure 2.10 (Leser & Naumann 2007, p. 61). Although these categories provide a useful analytical framework, in practice they are not always clearly separable and often occur in complex mixtures.

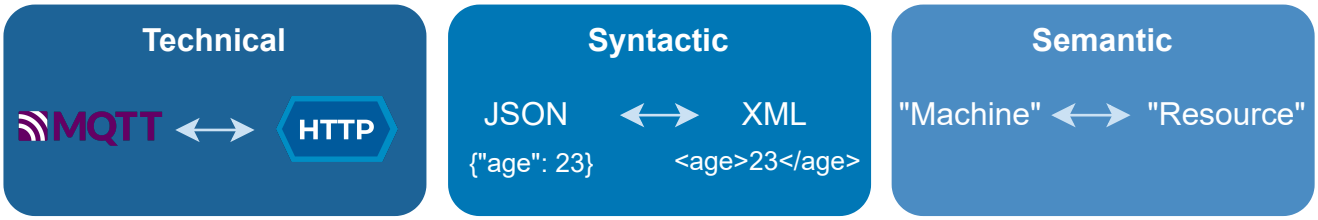

Figure 2.10: Overview of the data heterogeneity classes based on Leser & Naumann (2007)

Technical heterogeneity encompasses all issues related to the technical implementation of data access, including differences in hardware, operating systems, database management systems, and network protocols (Leser & Naumann 2007, p. 61; Hull 1997, p. 51).

Syntactic heterogeneity concerns differences in the format used to represent information, including conflicts arising from different data models (e.g., relational vs. object-oriented) or specific formats (e.g., XML vs. JSON) (Leser & Naumann 2007, pp. 6, 61).

The most difficult integration challenge lies in overcoming **semantic heterogeneity**, which relates to disagreements in the meaning and interpretation of data (Leser & Naumann 2007, pp. 6, 61): even when systems use the same syntax, the intended meaning can vary significantly, leading to costly integration failures (Erl 2016, p. 106). Semantic conflicts arise because disparate systems are designed for different purposes (Doan & Halevy et al. 2012, pp. 7, 92, 125; Hansen & Madnick et al. 2003, p. 177). Common examples include differences in scale and representation (e.g., Celsius vs. Fahrenheit), naming conflicts (e.g., “Machine” vs. “Resource”), and structural heterogeneity when schemas are differently structured.

True interoperability depends on achieving semantic agreement, a task that requires a deep “understanding” of the data that cannot be solved by technical and syntactic standardization alone (Bianco & Kotermanski et al. 2007, p. 21; Leser & Naumann 2007, p. 6). A successful integration strategy for I4.0 must therefore address heterogeneity on all levels, with a particular focus on resolving semantic conflicts to unlock the full value of data in smart manufacturing environments (Alonso & Casati et al. 2004, p. 134).

2.4.2 Formalization of the Data Integration Problem

To provide a rigorous foundation, we adopt the established theoretical framework for data integration proposed by Lenzerini (2002). This model formalizes a system designed to provide a unified view over a set of heterogeneous data sources.

Definition 1 (Data Integration System (Lenzerini 2002))**.** *A **Data Integration System**, denoted $\mathcal{I}$, is a triple $(\mathcal{G}, \mathcal{S}, \mathcal{M})$ where:*

- *$\mathcal{G}$ is the **Global Schema**, expressed in a language $\mathcal{L}_\mathcal{G}$ over an alphabet $\mathcal{A}_\mathcal{G}$. It provides a reconciled and unified view of all data.*
- *$\mathcal{S}$ is the **Source Schema**, expressed in a language $\mathcal{L}_\mathcal{S}$ over an alphabet $\mathcal{A}_\mathcal{S}$. It describes the structure of the underlying data sources.*
- *$\mathcal{M}$ is the **Mapping** between the global schema $\mathcal{G}$ and the source schema $\mathcal{S}$. It is constituted by a set of assertions that logically relate the concepts in $\mathcal{G}$ to the concepts in $\mathcal{S}$. Each assertion establishes a correspondence between a query over $\mathcal{G}$ and a query over $\mathcal{S}$.*

The global schema $\mathcal{G}$ and source schema $\mathcal{S}$ are formal descriptions of data structures. The definition is not restricted to a single data model paradigm, the schemas can be relational, object-oriented, graph-based, or follow other formalisms. The *alphabet* $(\mathcal{A}_\mathcal{G}, \mathcal{A}_\mathcal{S})$ of a schema contains the symbols of its model, such as attribute or class names. The *language* $(\mathcal{L}_\mathcal{G}, \mathcal{L}_\mathcal{S})$ defines the constructs available to express the schema, ranging from simple structural definitions to rich languages that can enforce integrity constraints. Examples for languages to express schema definitions are SQL's Data Definition Language for relational data models or the Unified Modeling Language (UML) for object-oriented models. In web service communication, a widely used object-oriented schema definition language is JSON Schema (see Section 2.5.5).

The mapping $\mathcal{M}$ is the declarative specification that bridges the semantic gap between the global schema and the sources. The assertions in the mapping are typically specified using one of two fundamental approaches:

- **Global-as-View (GAV):** Each concept in the global schema $\mathcal{G}$ is defined directly by a formula or recipe that combines concepts from the source schema $\mathcal{S}$. This approach establishes a direct, bottom-up correspondence from the sources to the global model, specifying *how* to construct each unified concept. For example, the global concept of a `User` could be defined as the combination of all records from the source tables `Customer` and `Client`.

- **Local-as-View (LAV):** Each source concept in $\mathcal{S}$ is described as a specific, constrained portion of the concepts in the global schema $\mathcal{G}$. This approach provides a top-down description, where the global schema acts as a stable, overarching model to which sources are mapped. It describes *what* information a source contains in terms of the unified model. For example, a source table named `GermanCustomers` would be described as corresponding to those `User` objects in the global schema where the 'country' attribute has the value 'Germany'.

This presented formalism establishes the core components needed to reconcile data heterogeneity: a target unified model ($\mathcal{G}$), the models of the individual sources ($\mathcal{S}$), and the explicit rules for transformation between them ($\mathcal{M}$). In the following, we will explore in more detail which steps are necessary to define these mappings for data integration.

2.4.3 Schema Integration Process

At the heart of any data integration effort lies the schema integration process, a structured procedure for reconciling the schematic differences between software systems (Rahm & Bernstein 2001, p. 334; Doan & Halevy et al. 2012, p. 162). As visualized in Figure 2.11, the process can be broken down into a sequence of three core tasks: schema matching, schema mapping, and data transformation.

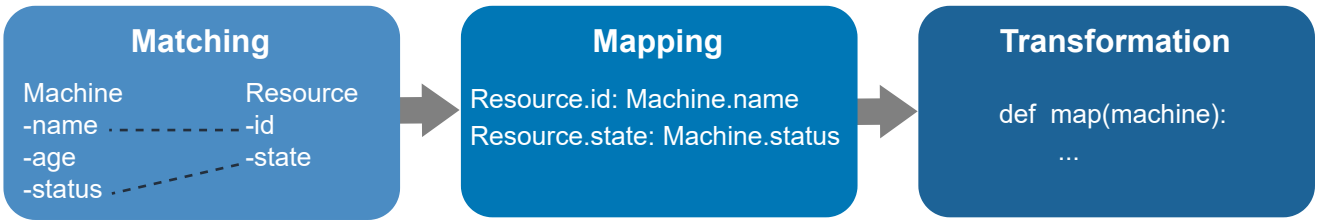

Figure 2.11: Overview of the schema integration process based on Doan & Halevy et al. (2012)

The process begins with **schema matching**, which identifies elements across different schemas (database tables, attributes, or classes) that correspond to the same real-world concept (Christen 2012, p. 3; Rahm & Bernstein 2001, p. 334). For example, the `name` attribute in a `Machine` schema might be semantically equivalent to an `id` field in a `Resource` schema. Schema matching is fundamental to nearly all data-centric applications handling multiple sources and is widely considered one of the most tedious and time-consuming parts of integration, often requiring substantial human involvement even when supported by semi-automatic techniques (Rahm & Bernstein 2001, p. 334; Lemos & Daniel et al. 2015, p. 347).

Once semantic correspondences have been identified, the next step is to create a **schema mapping**: a *declarative* formal specification (center of Figure 2.11) that defines how the data structured under a source schema can be translated into the structure of a target schema (Bernstein & Haas 2008, p. 77; Leser & Naumann 2007, p. 116). In practice, mappings often go beyond simple one-to-one matches and define complex transformations required to bridge structural and semantic gaps, such as combining `FirstName` and `LastName` fields from a source into a single `FullName` field in the target (Doan & Halevy et al. 2012, p. 129).

The mappings serve as the blueprint for the final task: **data transformation**. In this step, the declarative mapping specifications are operationalized as *imperative* transformation routines, which define – typically in executable code – the procedural, step-by-step conversion of source data into the target representation (Doan & Halevy et al. 2012, p. 271). Transformation routines, often implemented within Extract-Transform-Load (ETL) pipelines, can involve operations such as selecting subsets of data, joining tables, aggregating values, and applying data cleaning or normalization procedures (Doan & Halevy et al. 2012, p. 271; Rahm & Bernstein 2001, p. 334).

The schema integration process is not performed in isolation, instead it is implemented within broader system designs. The following architectural patterns determine how data is physically and logically managed during the integration process.

2.4.4 Architectural Approaches to Integration

Patterns for data integration dictate how data is accessed, processed, and presented, fundamentally shaping the integration solution. Their core components typically include the data sources, a global or mediated schema, and the mappings that bridge the heterogeneity gap (Doan & Halevy et al. 2012, p. 125). While early architectural patterns such as Enterprise Application Integration (EAI) and the Enterprise Service Bus (ESB) established standardized methods for integrating business processes through message exchange (Bernstein & Haas 2008, p. 75; Kaib 2002, p. 79), the rise of modern web services and IIoT standards has shifted the focus: by promoting common protocols (HTTP, OPC UA) and data formats (JSON, XML), these technologies have largely resolved technical and syntactic heterogeneity, allowing integration efforts to concentrate on the more complex semantic level (Erl 2016, p. 24; Boyes & Hallaq et al. 2018, p. 9).

At a higher architectural level, a key distinction is made between *materialized* and *virtual* integration. The materialized approach, best exemplified by the Data Warehouse, involves physically consolidating data from multiple sources into a new, central database (Bernstein & Haas 2008, p. 75; Leser & Naumann 2007, p. 403). Data is periodically copied from sources, cleansed, and reshaped into a common schema, which improves query performance and

data quality but reduces data freshness (Bernstein & Haas 2008, p. 75; Hull 1997, p. 54). This approach inherently aligns with the GAV paradigm, as it requires a prescriptive recipe for constructing each element of the global schema from the sources.

In contrast, *virtual data integration* provides the illusion of an integrated database without physically materializing the data (Bernstein & Haas 2008, p. 75). In this model, often called Enterprise Information Integration (EII), users pose queries against a virtual mediated schema; the system translates these queries into sub-queries on the underlying sources, executes them, and integrates the results on the fly (Bernstein & Haas 2008, p. 75; Leser & Naumann 2007, p. 5). This ensures access to the most up-to-date data and represents the LAV integration approach, as it describes what information each source contains in terms of the global model (Doan & Halevy et al. 2012, p. 271).

2.4.5 Challenges of Data Integration

Despite the availability of sophisticated architectural patterns, the schema integration process faces significant and persistent challenges. As different system designers inevitably create different schemas even for the same domain, real-world entities are frequently represented using conflicting values, formats, and structures (Doan & Halevy et al. 2012, pp. 7, 92). This is compounded by organizational challenges, where the deep semantic knowledge required for accurate matching and mapping is often distributed among multiple domain experts, lost over time, or hindered by data owners reluctant to cooperate (Doan & Halevy et al. 2012, pp. 7, 121; Bernstein & Haas 2008, p. 79). Consequently, the process becomes manually intensive, brittle, and prone to breaking when source schemas evolve (Lemos & Daniel et al. 2015, p. 351).

A prominent approach for semantic integration is the use of *ontologies*, which provide formal, explicit specifications of a shared conceptualization by defining a common vocabulary and the relationships between concepts (Leser & Naumann 2007, p. 267). Rather than constituting a global schema in a strict structural sense, ontologies operate at a conceptual level and can serve as a semantic foundation for mapping disparate data sources, enabling formal reasoning and logical inference to explicitly resolve semantic heterogeneities (Bernstein & Haas 2008, p. 77).

However, the adoption of ontologies in industrial practice faces significant hurdles: reaching consensus on a single, standardized ontology is a major organizational and technical challenge (Doan & Halevy et al. 2012, p. 126); semantic technologies are often perceived as complex and exhibit performance and scalability issues critical in the data-intensive, low-latency environments of I4.0 (Tzavaras & Mainas et al. 2023; Kang & Krishnaswamy et al. 2020); and the resource-intensive process of creating and maintaining an ontology often

fails to capture the pragmatic, purpose-driven context of information exchange, positioning ontologies more as a research field than a widespread industrial solution (Leser & Naumann 2007, p. 288; Alizadeh & Shahrezaei et al. 2019).

Semantic integration thus remains the central, persistent challenge in data integration (Leser & Naumann 2007, p. 122). Particularly in PPC, most integration efforts achieve technical and syntactic interoperability through standardized communication, while the crucial semantic integration is left to manual, error-prone, and costly human interpretation (Zelewski & Hohmann et al. 2010, p. 776). This restricts the reconfigurability of PPC systems in the context of I4.0 and RMS, making software systems limiting factors for change. Realizing the vision of I4.0 therefore requires a shift from manual data integration toward flexible, automated approaches capable of inferring semantic correspondences in real-time – a move toward context-aware technologies that motivates the application of LLMs.

2.4.6 Large Language Models for Semantic Integration

LLMs represent a significant departure from traditional, rule-based integration algorithms. This section outlines their architectural principles and analyzes the capabilities that render them effective for resolving semantic heterogeneity in manufacturing environments.

Transformer Architecture and Contextual Semantics. LLMs are neural networks distinguished by their immense scale, often containing hundreds of billions of parameters, and their training on diverse corpora of text and code. While their fundamental training objective is probabilistic next-token prediction, their utility in data integration stems from the underlying Transformer architecture introduced by Vaswani & Shazeer et al. (2017). In contrast to traditional schema matching, which relies on linguistic similarity (e.g., string distance metrics) or static dictionaries and often fails when facing ambiguity, the Transformer's *self-attention* mechanism processes an entire input sequence in parallel and dynamically weights the relevance of surrounding tokens to determine meaning (Vaswani & Shazeer et al. 2017). This is paramount for interpreting complex data schemas: the semantic meaning of a schema element such as a field named `id` is rarely intrinsic but defined by its hierarchical context. By attending to parent entities (e.g., `Resource` or `Machine`) and adjacent attributes simultaneously, an LLM constructs a context-aware representation of the data structure, effectively simulating a domain expert's ability to infer meaning from context and resolving semantic ambiguities that derail rigid, rule-based matching algorithms.

In-Context Learning and Generalization. A paradigm-shifting capability of LLMs is *In-Context Learning*: LLMs can perform new tasks based solely on instructions and examples provided within the input prompt, without requiring updates to the model's internal parameters (Brown & Mann et al. 2020). This is categorized into *Zero-Shot Learning*, where the model

executes a task based on instruction alone, and *Few-Shot Learning*, where the instruction is augmented with a small number of example demonstrations. Traditional machine learning approaches for data integration instead rely on supervised learning with large, domain-specific datasets of labeled mappings, which is impractical in brownfield manufacturing environments where such data is scarce. In contrast, in-context learning allows integration logic to be defined declaratively at runtime by providing source and target schemas along with a few transformation examples (Buss & Safari et al. 2025), eliminating costly training phases and enabling dynamic adaptation to new or evolving schemas simply by modifying the prompt context.

Instruction Following and Code Synthesis. While the Transformer architecture enables understanding and in-context learning enables agility, the ability to generate executable artifacts is driven by the LLM's proficiency in code synthesis. Modern LLMs are trained on vast repositories of source code in addition to natural language, allowing users to specify complex transformation rules (e.g., "combine first and last name", "filter null values", or "convert units") in natural language and have them translated into executable code in high-level languages such as Python or JavaScript (Zhao & Zhou et al. 2025). This moves the integration process beyond simple schema matching – which produces static correspondence tables – toward full automation, as the system can generate imperative mapping functions that handle data type conversion, structural reshaping, and error handling. Reliability is further enhanced through techniques such as *Chain-of-Thought (CoT)* reasoning, where the model articulates its logical steps before generating code (Zhao & Zhou et al. 2025), which increases transparency and tends to outperform direct inference on complex tasks with large contexts.

This section has established the theoretical foundations of data integration and highlighted semantic heterogeneity as the central obstacle to interoperability in I4.0 environments. While recent advances – most notably LLMs – significantly reduce the manual effort required for schema matching, mapping, and transformation, these capabilities alone do not constitute an operational system. Their value can only be realized when integration is embedded within a coherent software architecture that governs how functionality is exposed, composed, and evolved across organizational and system boundaries (Papazoglou & Van Den Heuvel 2006, p. 412). Effective interoperability and reconfigurability in PPC consequently require not only intelligent integration mechanisms, but also an architectural style that systematically exploits these capabilities at the system level.

2.5 Service-oriented Architectures in Industry 4.0

This section introduces service-oriented architectures (SOA) as a key architectural paradigm for implementing modular, interoperable, and evolvable software systems in I4.0. Section 2.5.1

first provides the historical context and business motivation for SOA, linking its principles to the need for adaptability in manufacturing and PPC. Building on this, Section 2.5.2 summarizes core SOA principles and architectural building blocks, and Section 2.5.3 introduces microservices architecture (MSA) as an evolution emphasizing service autonomy. Section 2.5.4 then introduces event-driven architecture (EDA) as a complementary paradigm for asynchronous, push-based integration. The technological realization of SOA via web APIs and interface specifications is discussed in Section 2.5.5. Finally, Section 2.5.6 addresses how business processes are coordinated through orchestration and choreography, and Section 2.5.7 outlines deployment and scalability practices that operationalize SOA.

2.5.1 Historical Context and Business Motivation

SOA emerged from 1990s-era distributed messaging middleware systems and was driven by the need to integrate increasingly complex and heterogeneous IT landscapes while overcoming the limitations of tightly coupled integrations seen in earlier EAI systems (Erl 2016, p. 24; Valipour & Amirzafari et al. 2009; Papazoglou & Van Den Heuvel 2006, p. 412). Its business motivation stems from the core principles of modularity, loose coupling, and reusability (Niknejad & Ismail et al. 2020): by decomposing monolithic applications into discrete, interoperable services, enterprises gain the flexibility to dynamically compose and reuse business logic.

This agility is critical for adapting to market changes, especially within the I4.0 context (Mönch & Fowler et al. 2013, p. 248). SOA supports manufacturing paradigms such as RMS by making complexity manageable and reducing the *total cost of change*, as selective updates replace complete system overhauls and prolong the lifecycle of IT investments (Zhong & Xu et al. 2017; Koren & Heisel et al. 1999; ElMaraghy & Monostori et al. 2021). With regard to the requirements for more adaptive PPC systems, SOA represents a suitable architectural style (Zelewski & Hohmann et al. 2010, p. 866). Nevertheless, achieving fully integrated, service-oriented enterprises in the manufacturing domain remains challenging: while sectors such as MES-driven automotive production show advanced implementation, others still struggle with legacy system integration and persistent data silos between platforms such as ERP and MES (Babel 2024, p. 164).

2.5.2 Principles and Building Blocks of Service-oriented Architecture

SOA is not a specific technology but an architectural paradigm based on a set of design principles for building distributed systems (Papazoglou & Van Den Heuvel 2006, p. 413). It structures an application as a collection of distinct, modular services that can be flexibly combined and reused to implement new or enhance existing business processes (Valipour & Amirzafari et al. 2009; Niknejad & Ismail et al. 2020).

The fundamental building block of an SOA is the *service*, a self-contained software component that exposes its functionality through a well-defined interface while encapsulating its technical realization in an underlying implementation (Valipour & Amirzafari et al. 2009). A service consumer interacts exclusively via this interface, abstracting away implementation details. In an SOA landscape, services rarely operate in isolation but are composed and consumed by higher-level *applications* that realize a business process or user function; applications may themselves expose parts of their functionality as new services, allowing SOA to scale from individual services to enterprise-wide ecosystems (Krafzig & Banke et al. 2004).

According to Krafzig & Banke et al. (2004), a canonical SOA landscape consists of several key architectural components, as visualized in Figure 2.12, that fulfill distinct roles. The *service provider* implements a service and makes it available on the network. In practice, and as illustrated in the figure, each deployed service instance (Service 1–3) assumes the role of a service provider. The *service consumer* is an application or another service that requests the execution of a service from a provider. A *service registry* acts as a central directory where providers publish their service descriptions, and consumers can discover them. Communication is often mediated by a *service bus*, which provides an underlying backbone for message routing and interaction between consumers and providers. While central to the SOA vision, some implementations may operate without a formal registry (Bianco & Kotermanski et al. 2007).

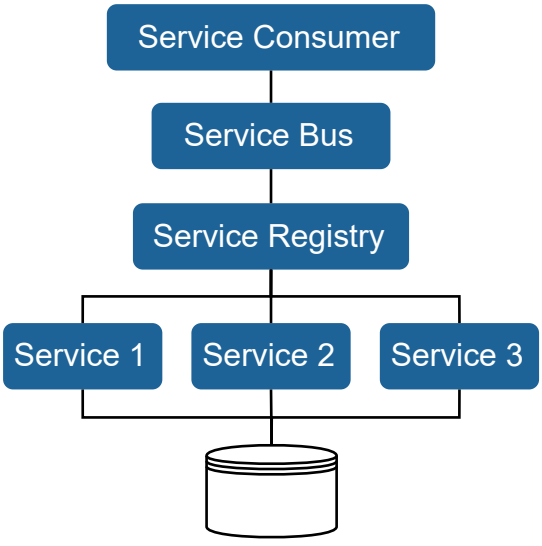

Figure 2.12: Overview of the components in SOA according to Krafzig & Banke et al. (2004).

At its core, SOA is governed by a set of fundamental principles designed to promote agility, reusability, and interoperability (Niknejad & Ismail et al. 2020; Valipour & Amirzafari et al. 2009; Krafzig & Banke et al. 2004; Bianco & Kotermanski et al. 2007):

- **Modularity:** Services are *self-contained* and *modular*, each encapsulating a specific, cohesive set of business functions.
- **Loose Coupling:** Minimizes the dependencies between service consumers and providers, ensuring that changes in one service have minimal impact on others.
- **Service Contract:** Services adhere to a formal, self-describing contract that defines their functionality, data formats, and usage constraints, which is a prerequisite for discovery and interaction.
- **Interoperability:** Systems built on different platforms and in different programming languages can communicate effectively, typically by leveraging standardized protocols and data formats.
- **Discoverability:** Services can be dynamically found by querying a service registry.
- **Location Transparency:** a service's physical location can change without affecting its consumers. This is based on service-discovery and *dynamic binding*, where connection to a service's network is done at runtime.
- **Composability:** Services can be combined to realize business processes.

A primary goal of applying these principles is to maximize reusability. This leads to a distinction between *agnostic services*, which are designed to be multipurpose by containing generic logic that is not specific to any single business process, and services that contain task-specific logic and are therefore limited in their reusability (Erl 2016, pp. 42, 78). A mature SOA strategically leverages agnostic services to build a flexible and efficient enterprise-wide application portfolio – a principle that becomes particularly relevant for PPC, where task-agnostic service designs enable the dynamic composition and reconfiguration of planning processes from interchangeable building blocks.

2.5.3 Microservices Architecture

A closely related architecture to SOA is the MSA, considered an evolution and a more opinionated implementation style of SOA (De Lauretis 2019). While MSA and SOA share many ideas, the primary architectural distinction, as shown in Figure 2.13, lies in MSA's emphasis on maximizing service autonomy to achieve extremely low coupling (Richards 2015, p. 22). This autonomy is accomplished by designing services to be fully self-contained, including each service managing its own independent database (De Lauretis 2019). MSA also departs from the traditional SOA pattern of a centralized service bus, instead favoring a share-as-little-as-possible philosophy where individual services contain the business logic and

communicate directly through lightweight mechanisms; an *API Gateway* typically provides unified access and routing to all services (Richards 2015, p. 24).

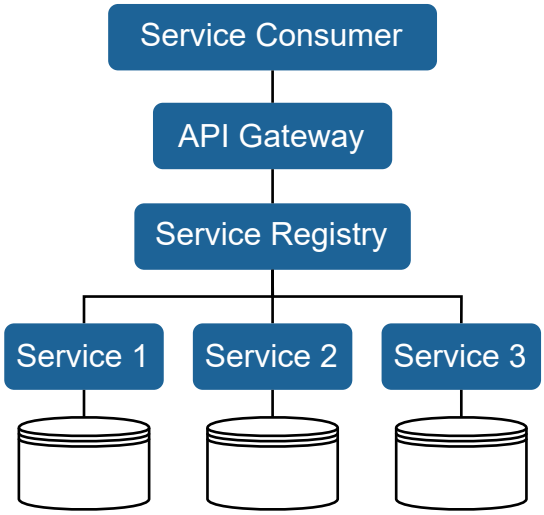

Figure 2.13: Overview of the components in MSA according to Richards (2015).

Although MSA improves low coupling, it comes at the cost of lower reusability of services with the risk of inconsistent schemas, requiring potentially higher integration efforts, and departing from the service bus removes the protocol-agnostic nature of SOA (Richards 2015, pp. 22, 43; Dmitry & Manfred 2014). Many organizations therefore employ a hybrid approach, where truly independent microservices coexist with more traditional services that may share a database or rely on a bus for integration. Especially in IIoT, an asynchronous shared service bus can make information distribution more manageable (Dmitry & Manfred 2014).

2.5.4 Event-driven Architecture

While SOA and MSA organize systems around explicit service calls, EDA structures systems around the production, detection, and reaction to *events* – significant changes in state, such as the completion of a manufacturing operation or the arrival of a new sensor value (Erl 2016, p. 336). In contrast to the synchronous request-response model, EDA employs asynchronous, message-based communication where components emit and react to events via publish-subscribe mechanisms (Hunkeler & Truong et al. 2008; Papazoglou & Van Den Heuvel 2006). The decoupling achieved through event streams and message brokers (e.g., MQTT) enables high scalability and resilience, as producers and consumers of information do not need to be aware of each other's identity or timing (Erl 2016, p. 336). This communication style aligns with *push-based* information exchange (see Section 2.3.3.1), supporting real-time responsiveness and temporal decoupling between distributed manufacturing components.

In the context of PPC, EDA complements service-oriented approaches by providing the infrastructure for responsive, data-driven decision-making. For instance, a scheduling service

can automatically react to machine state changes or order completions emitted as events, thereby enabling closed-loop control. Many modern architectures consequently adopt hybrid patterns combining pull-based communication for process orchestration with EDA for event propagation and monitoring (Valipour & Amirzafari et al. 2009; Richards 2015).

2.5.5 Web Service Technologies

While SOA itself is technology-neutral, its principles are most commonly realized through Web Services (Valipour & Amirzafari et al. 2009). The earlier “Big Web Services” (WS-*) stack, which relied on SOAP and XML for data transfer (Alonso & Casati et al. 2004, p. 131), has been largely superseded by more lightweight, web-aligned technologies (Richardson & Ruby 2007). Today, two architectural styles dominate the implementation of services: REST and GraphQL, both depicted in Figure 2.14.

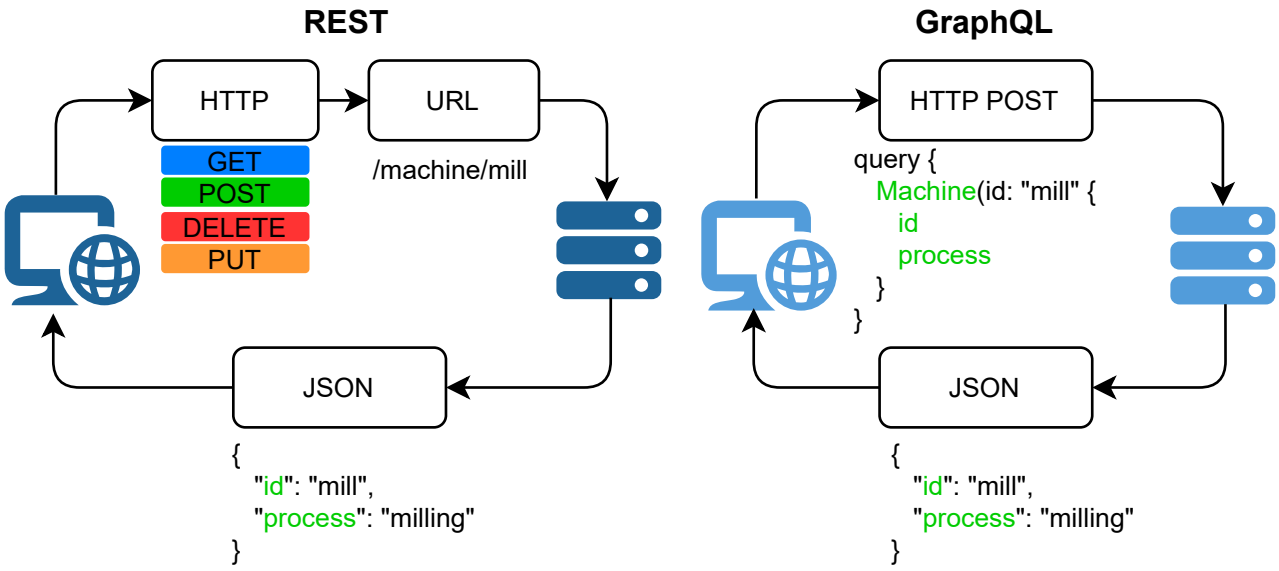

Figure 2.14: Visualization of the client-server interactions in REST and GraphQL APIs, based on Richardson & Ruby (2007) and Hartig & Pérez (2018).

REST is an architectural style, not a protocol, that leverages the standard features of HTTP (Fielding 2000). It treats entities as *resources*, each identified by a unique URI (e.g., `/machine/mill`), with which clients interact using standard HTTP methods such as `GET` or `POST`. REST is stateless: each request must contain all the information necessary for the server to fulfill it, which enhances scalability and reliability. **GraphQL** is a query language for APIs and a runtime for fulfilling those queries with existing data (Hartig & Pérez 2018), developed to address REST’s limitations such as over-fetching or under-fetching. GraphQL exposes a single endpoint to which the client sends a structured query specifying exactly which data fields it requires; the server returns a JSON object that precisely matches the query structure, providing greater flexibility and efficiency for clients with varying data needs.

For data exchange in both styles, JSON is the dominant data format due to its simplicity, lightweight nature, and parsing performance (Breje & Gyorödi et al. 2018; Cánovas Izquierdo & Cabot 2013). To ensure interoperability and correctness, the structure of exchanged messages must be formally defined: in REST, the **OpenAPI Specification** has become the industry standard for describing endpoints, operations, parameters, and request/response bodies in a language-agnostic manner (Tzavaras & Mainas et al. 2023), while **JSON Schema** is a widely used standard for defining the structure of JSON data itself (Hartig & Pérez 2018); OpenAPI incorporates a data format that is fully compatible with a subset of JSON Schema. In GraphQL, this contract is inherently part of its strongly typed schema, which defines the entire data graph and serves as the authoritative documentation for the API. A more detailed discussion of these specification languages follows in Appendix A2.

Push-Based Communication in Web Services for EDA. Beyond request/response interactions, modern service-oriented systems increasingly rely on push-based communication to support EDA (Eugster & Felber et al. 2003). Two widely adopted Web-based technologies are *Webhooks*, which implement a callback mechanism in which the client exposes an endpoint that the server invokes whenever a relevant event occurs (Jin & Sahni et al. 2018), and *Server-Sent Events (SSE)*, which establish a long-lived, unidirectional HTTP connection over which the server continuously streams event updates to the client (Słodziak & Nowak 2016). Both mechanisms enable efficient real-time event propagation and are commonly used in lightweight microservice and IIoT integrations.

2.5.6 Coordination of Business Processes in Service-oriented Architecture

In SOA, end-to-end business processes are realized by *composing* multiple services into an executable whole. For PPC, this composition is a key lever to overcome the rigidity of traditional, tightly coupled PPC systems: functions such as scheduling, order release, machine status monitoring, and exception handling can be combined, replaced, or extended at runtime without touching a monolith's internals (Mönch & Fowler et al. 2013, p. 248; Zelewski & Hohmann et al. 2010, p. 866).

The coordination of these service compositions follows two primary models (see Figure 2.15): *orchestration*, where a central, executable process controls the sequence and data flow, and *choreography*, where participating services follow a shared protocol without a central controller (Lemos & Daniel et al. 2015). Orchestration is well suited to implement PPC logic as a controllable process (e.g., rescheduling workflows), while choreography fits decentralized collaboration scenarios (e.g., B2B supply processes) and aligns with microservice-style autonomy (Valipour & Amirzafari et al. 2009; Richards 2015, p. 26).

To specify service orchestration, process modeling approaches such as Business Process Model and Notation (BPMN) or Petri nets can be used to describe the sequence and logical flow of the orchestration (Safi & Murad et al. 2011). Petri nets are particularly suitable when executable process models with formally analyzable behavior are required, as is the case for automated PPC orchestration. While BPMN models can be mapped to Petri nets for analysis, BPMN itself lacks a strong mathematical formalism for rigorously defining and analyzing behavioral properties of process models.

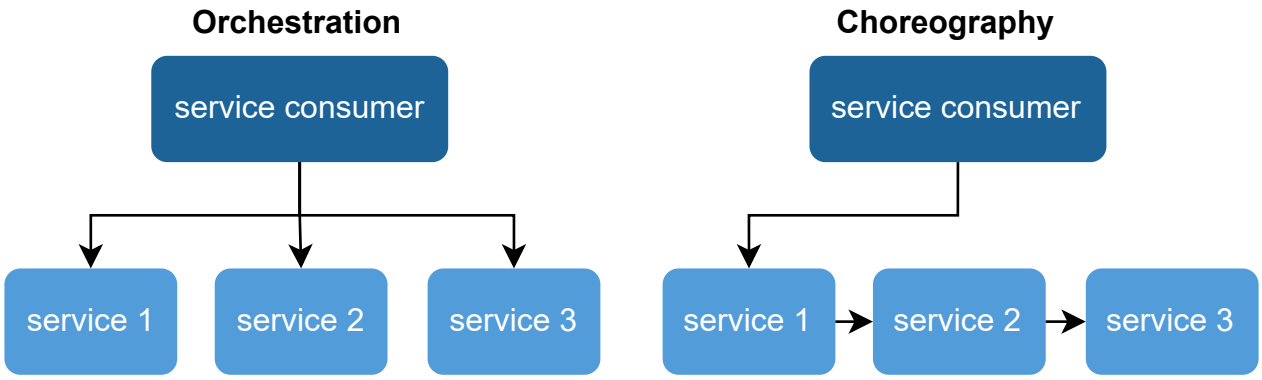

Figure 2.15: Comparison of the service composition models service orchestration and service choreography, based on Richards (2015).

2.5.6.1 Petri Nets

Petri nets provide a formal and graphical modeling language for representing concurrency, synchronization, and causality in distributed systems (Reisig 2011; Van Der Aalst 1998). They are widely used in workflow and process model management as well as in service orchestration, where explicit control-flow and dataflow modeling is required.

A Petri net is a tuple

$$N = (P, T, F, W, M_0), \tag{2.1}$$

where:

- P is a finite set of places,
- T is a finite set of transitions (with $P \cap T = \emptyset$),
- $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation,
- $W : F \to \mathbb{N}_{>0}$ is an arc-weight function, and
- $M_0 : P \to \mathbb{N}$ is the initial marking.

A marking is a function $M : P \to \mathbb{N}$ assigning a non-negative number of tokens to every place. The pair (N, M) describes the current state of the net.

For a transition $t \in T$, let the set of input places (preset) and output places (postset) be defined respectively as:

$$\bullet t = \{p \in P \mid (p, t) \in F\} \quad \text{and} \quad t^\bullet = \{p \in P \mid (t, p) \in F\}. \tag{2.2}$$

A transition t is enabled under marking M if and only if:

$$\forall p \in \bullet t : M(p) \geq W(p, t). \tag{2.3}$$

If t is enabled, firing t consumes tokens from its input places and produces tokens in its output places according to the arc weights, yielding a new marking.

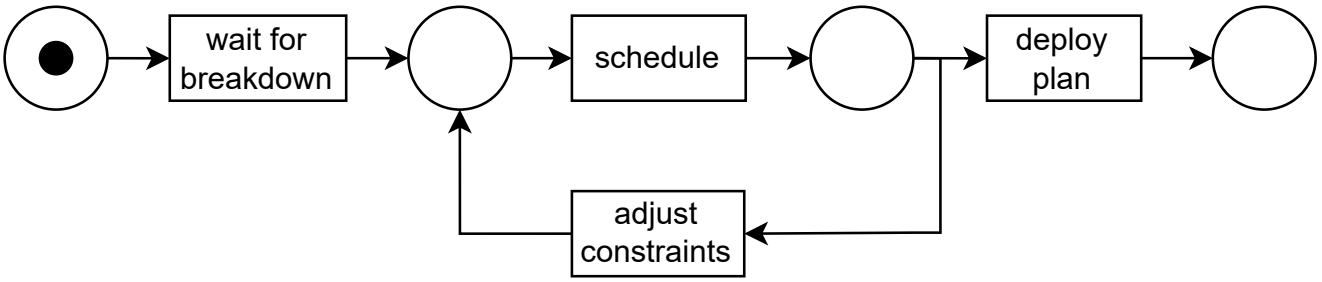

Figure 2.16: Visualization of a Petri net describing a simple PPC process model.

An example of a Petri net is visualized in Figure 2.16, which models a simple planning and execution process in PPC. Places represent discrete process states, while transitions model the execution of PPC activities that move the process from one state to another. Petri nets naturally model structural properties such as *concurrency* (two transitions are concurrently enabled if they do not share input places or if sufficient tokens exist to satisfy both), *conflict* (multiple enabled transitions competing for the same tokens), and *synchronization* (a transition requiring tokens from several places, $|\bullet t| > 1$).

Beyond structural modeling, Petri nets enable the formal analysis of behavioral properties that are critical for executable service orchestrations and PPC workflows (Van Der Aalst 1998; Reisig 2011). Among the most relevant are *reachability* (which markings can be reached from M_0 through firing sequences), *boundedness* (the number of tokens per place is bounded, ensuring finite resource usage), *liveness* (from any reachable marking, every transition can eventually fire, excluding deadlocks), and *soundness* as defined for workflow nets, which guarantees proper termination by requiring that a designated final marking is always reachable and contains no residual tokens. These properties provide a formal basis for validating that

an orchestration model is free of deadlocks, livelocks, and unintended behavior, making Petri nets particularly suitable for executable PPC workflows.

An extension of Petri nets are CPNs, which associate data values (“colors”) with tokens (Jensen & Kristensen 2009). A CPN is defined as the tuple

$$N = (P, T, A, \Sigma, C, E, G, M_0), \tag{2.4}$$

where (as defined above) P is the set of places, T the set of transitions, A the set of arcs, and M_0 the initial marking. Additionally:

- Σ is a set of color sets (data types),
- $C : P \to \Sigma$ assigns a color type to each place,
- $E : A \to$ Expr assigns expressions to arcs (where A is the set of arcs), and
- $G : T \to \text{Expr}_{\text{bool}}$ assigns guard expressions to transitions.

This extension enables the modeling of complex PPC-relevant data such as production orders, schedules, and machine states, making CPNs particularly suitable as a foundation for the extended orchestration formalism introduced in later chapters.

2.5.7 Deployment and Scalability Practices

The principles of SOA form a conceptual foundation for modern cloud-native deployment and operations. Its inherently distributed nature – particularly the principle of location transparency – makes SOA well aligned with cloud computing environments (Niknejad & Ismail et al. 2020): because services are discovered through registries and dynamically bound at runtime, they can be replicated, relocated, or scaled across infrastructure resources without affecting their consumers. This elasticity directly supports reconfigurable manufacturing environments, where computational resources and planning services must dynamically adjust to changing production configurations and workloads. Furthermore, SOA’s modular and independently deployable nature inherently supports DevOps practices (Ebert & Gallardo et al. 2016): through containerization, each service can be packaged with its dependencies into a portable unit, facilitating automated CI/CD pipelines that allow teams to build, test, and release services autonomously. These capabilities accelerate iteration cycles and enable rapid adaptation of planning and control logic – an essential prerequisite for realizing reconfigurability in PPC systems.

2.6 Conclusion of Theoretical Foundations

In summary, this chapter has established the theoretical pillars that underpin the research presented in this thesis. It began by examining the domain of PPC, highlighting its central role in manufacturing operations. A critical challenge identified is the persistent “implementation gap” between abstract, hierarchical PPC models and their realization in software, i.e. PPC systems. Current PPC systems are often monolithic, rigid applications that lack the flexibility to adapt to changing operational requirements, thus constraining the very processes they are meant to manage.

The need for greater adaptability is amplified by the paradigm of changeable manufacturing, particularly the concept of RMS. As established, the core value proposition of RMS, the ability to cost-effectively adjust production capacity and functionality, is fundamentally dependent on a corresponding reconfigurability at the software level. This imposes a parallel demand for flexible PPC systems capable of managing dynamic tasks since a physically reconfigurable factory remains ineffective if its digital control systems cannot co-evolve.

The technological landscape of I4.0 provides the enabling technologies to address these challenges. The convergence of OT and IT through IIoT technologies dissolves the rigid automation pyramid, paving the way for distributed, interconnected information networks. However, this technological shift has not completely resolved the fragmented ecosystem of communication protocols, data formats, and domain-specific data models. Despite numerous standardization efforts, the manufacturing domain is characterized by significant data heterogeneity, which presents a formidable barrier to achieving seamless interoperability.

To overcome this fragmentation, the principles of data integration are essential. This chapter has categorized the different facets of heterogeneity (technical, syntactic, and semantic) and has identified semantic integration as the most persistent and complex challenge. While technical and syntactic issues can largely be resolved through modern standards, reconciling the meaning of data from disparate sources remains a significant hurdle. LLMs have been discussed as a promising enabler for automating parts of semantic integration, yet their application remains limited in industrial PPC contexts. As a result, semantic integration still relies on costly and error-prone manual efforts, which severely limits the reconfigurability and autonomy of software systems in the context of I4.0 and RMS.

Against this backdrop, SOA was identified as the most promising architectural paradigm for engineering the next generation of PPC systems. The core principles of SOA, such as modularity, loose coupling, integrability, and composability, directly mirror the design requirements of RMS. By structuring PPC functionalities as a collection of independent, interoperable services, SOA provides a foundation for building flexible, scalable, and adaptable systems.

Furthermore, models like service orchestration offer the means to dynamically compose and reconfigure business processes, directly addressing the requirement for adaptable production logic.

These foundations clarify the functional and technical requirements that any future PPC framework must address: it must be capable of integrating heterogeneous data sources, coordinating distributed processes, and allowing for the runtime reconfiguration of its components and logic. While SOA provides the architectural blueprint, the challenges of achieving automated semantic integration and managing legacy systems persist. The next chapter will build directly upon these insights by surveying the state of the art in industrial architectures, automated data integration approaches, and smart PPC methods. Through this review, the specific research gaps that motivate the proposed framework will be identified and formalized into the research questions guiding this work.

3 State of the Art and Research

The preceding chapter (Chapter 2) outlined the theoretical background of this work and revealed a fundamental mismatch between the requirements of modern manufacturing systems and the capabilities of conventional PPC systems. While I4.0 and RMS demand modular, changeable, and data-driven approaches, existing PPC systems remain constrained by monolithic architectures, rigid process logic, and fragmented data landscapes.

Against this backdrop, this chapter reviews the state of the art in the evolution toward integrated and adaptive PPC systems, guided by the structured evaluation framework introduced in Section 3.1. The review follows a chronological progression from industrial software architectures in Section 3.2, through automated data integration approaches in Section 3.3, to smart and automated PPC solutions in Section 3.4. The synthesis of these research streams highlights persistent limitations and open challenges, which motivate the research deficits and questions formulated in Section 3.5.

3.1 Criteria for Evaluating the State of Research

To enable a structured and transparent analysis of the literature, this work employs a multidimensional evaluation framework. While existing surveys provide valuable insights into individual aspects, such as specific planning algorithms, architectural styles, or data integration techniques, they rarely capture the interdependencies between these dimensions that are critical for realizing integrated and adaptive PPC systems.

The proposed framework comprises 21 evaluation criteria derived from the challenges and requirements identified in the theoretical foundations. These criteria are organized into three categories: Software Architecture and Service Orientation (S), Data Integration and Interoperability (I), and Smart and Automated PPC (P). Together, they provide a holistic perspective for systematically assessing existing approaches, revealing both achieved progress and persistent research gaps.

The software architecture of a PPC system determines its ability to evolve, scale, and adapt to changing production requirements. Accordingly, the underlying architectural paradigm is classified (S1), and its suitability for industrial application is assessed in terms of scalability (S2) and latency (S3). Guided by the core design principles of RMS (see Section 2.2.2) and SOA (see Section 2.5.2), architectural flexibility and reconfigurability are evaluated by examining modularity (S4), discoverability (S5), location transparency (S6), and loose coupling (S7). Finally, criterion S8 assesses the capability to adapt and coordinate control logic through service orchestration. Table 3.1 summarizes the corresponding evaluation

Table 3.1: Evaluation Criteria: Software Architecture and Service Orientation

ID	Criterion	Description
S1	Architecture Paradigm	Assesses the underlying architectural style (e.g., SOA, EDA, MSA, or Model-driven Architecture (MDA)).
S2	Scalability	Evaluates the ability to handle increasing load (horizontally/vertically).
S3	Latency	Measures system responsiveness for real-time monitoring and control.
S4	Modularity	Assesses composition from independent, self-contained modules or services.
S5	Discoverability	Evaluates the ability to dynamically find services at runtime.
S6	Location Transparency	Assesses if service consumers need to know the provider's physical location.
S7	Loose Coupling	Measures the degree of independence between system components.
S8	Orchestrability	Evaluates the capability to coordinate multiple services for complex processes.

Table 3.2: Evaluation Criteria: Data Integration and Interoperability

ID	Criterion	Description
I1	Technical Integration	Evaluates the ability to bridge different communication protocols (e.g., REST, MQTT).
I2	Syntactic Integration	Assesses the capability to handle diverse data formats automatically (e.g., JSON, XML).
I3	Semantic Integration	Evaluates the ability to reconcile different data models and semantics.
I4	Integration Approach	Identifies the method for conceptual integration (homogeneous schema, global schema, or point-to-point integration).
I5	Automated Integration	Measures automation degree of the schema integration process Section 2.4.3.
I6	Standardized Schemas	Assesses the use of established data models (e.g., AAS).
I7	Openness	Evaluates whether data models are publicly available to promote adoption.

criteria. In conceptual architectures, criteria are rated as partially fulfilled when mechanisms are specified but not demonstrated in a fully operational runtime implementation.

Since effective PPC requires seamless integration across heterogeneous IT and OT systems, the capabilities of existing approaches are further evaluated in the domain of *Data Integration and Interoperability* using the criteria summarized in Table 3.2. The degree of integration is

Table 3.3: Evaluation Criteria: Smart and Automated PPC

ID	Criterion	Description
P1	Automation of PPC	Assesses the degree of performing PPC tasks without human intervention.
P2	Task Agnosticism	Evaluates the ability to integrate new, unknown PPC services without core changes.
P3	Virtualization	Evaluates the use of simulation-based DTs to validate and optimize plans.
P4	Monitoring Real System	Assesses the ability to receive and process real-time data from the physical environment.
P5	Controlling Real System	Evaluates the ability to send control commands directly to production resources.
P6	Human-in-the-Loop	Assesses interfaces for human monitoring, supervision, and intervention.

assessed by differentiating technical (I1), syntactic (I2), and semantic (I3) levels, as introduced in Section 2.4.1. Criterion I4 characterizes how semantic integration is guided, distinguishing between approaches based on a single homogeneous schema (HS), the integration of multiple schemas into a global schema (GS), or point-to-point mappings (PTP). As semantic integration is typically associated with substantial manual effort, the degree of automation is assessed using criterion I5. Finally, criteria I6 and I7 evaluate whether approaches rely on formally defined schemas and open, standardized domain models to support scalability and long-term interoperability.

Beyond architecture and data integration, advanced PPC systems must support intelligent, closed-loop operation. Therefore, the degree of automation of PPC tasks (P1) and task agnosticism (P2), i.e., the ability to integrate new PPC services without modifying core system logic, are evaluated. Furthermore, with reference to established DT design criteria proposed by Kritzinger & Karner et al. (2018), it is assessed whether virtual models are employed for validation and optimization (P3) and whether automated information flows exist for monitoring (P4) and controlling (P5) the production system. Finally, criterion P6 captures the role of human interaction by evaluating support for transparency, supervision, and intervention within automated PPC processes. Table 3.3 summarizes the criteria addressing these higher-level capabilities for *Smart PPC*.

3.2 Approaches to Industrial Architectures

Industrial IT architectures have matured from foundational service-oriented concepts to software-defined ecosystems and decentralized microservice patterns, driven by the demands

of I4.0 for adaptability, data-driven decision-making, and interoperability (Luo & Thevenin et al. 2023).

Early Foundations: Vertical Integration and the Dawn of Software-Defined Concepts. The initial steps in this evolution focused on breaking down the traditional silos between the shop floor OT and enterprise IT. An early example of this is the model-based architecture proposed by Pérez & Irisarri et al. (2015), which aimed for a scalable (S2) and modular (S4) vertical integration of production and business processes. Their approach leveraged emerging standards like OPC UA and AML to create a homogeneous schema (I4), enabling the collection and contextualization of real-time production data (S3) for remote clients. This work established a foundational pattern: using standardized protocols (I1) and model definitions (I2,I6,I7) to improve interoperability and create a more holistic view of the production system.

Building on the need for greater flexibility, the paradigm of Software-Defined Manufacturing (SDM) emerged as a transformative concept. Thames & Schaefer (2016) first introduced the idea of a Software-Defined Cloud Manufacturing (SDCM) architecture, proposing a fundamental separation of the physical hardware (Hardware Plane) from the logical control and applications (Software Plane), as visualized in Figure 3.1. This separation, based on a unified control layer, enables the dynamic composition of manufacturing services from generic hardware and software elements promoting scalability (S2), modularity (S4), loose coupling (S7) and orchestrability (S8).

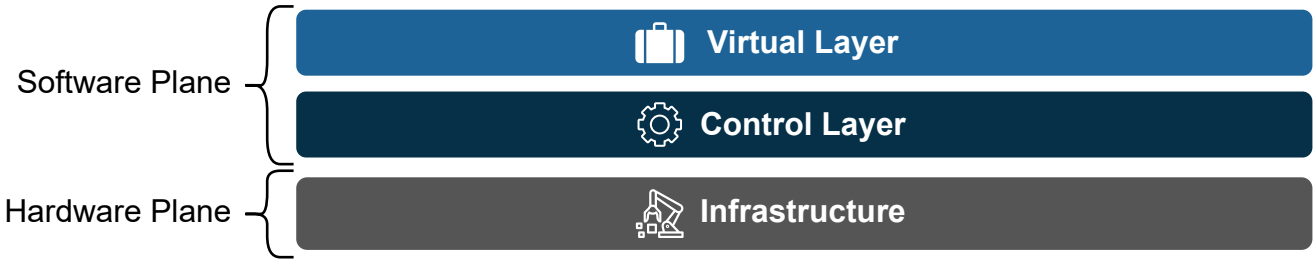

Figure 3.1: Overview of the SDM architecture adopted from Thames & Schaefer (2016) that shows a separation of the Virtual Layer and the Infrastructure based on a Control Layer.

This conceptual groundwork was significantly advanced by Lechler & Verl (2017), whose paper on SDM is a cornerstone in this domain. They refined the architecture by introducing a multi-layered model with a crucial component: the "Physical Abstraction Layer" (PAL). The PAL extends the functionality of the Control Layer, as introduced by Thames & Schaefer (2016), to encapsulate hardware-specific functionalities, creating an abstraction (S5, S6, S7) that allows the overlying application logic to be defined and reconfigured purely through software. This aligns conceptually with other approaches, such as Kusiak (2018) or Pérez & Irisarri et al.

(2015) , which also build upon a layered architecture for decoupling manufacturing equipment and IT systems.

A key insight from this line of work was the identification of OPC UA combined with Time-Sensitive Networking (TSN) as enabling technologies for achieving both flexible and deterministic communication (S3), critical for industrial control. Although primarily conceptual, these approaches laid out a comprehensive vision for complete software–hardware decoupling that promises rapid logical reconfigurations through loose coupling (S7), location transparency (S6), and modularity (S4) – a central theme that echoes through subsequent research on adaptive PPC.

Pragmatic Integration: Middleware and Brownfield Solutions. While the SDM vision matured, a parallel track of research focused on pragmatic solutions for integrating heterogeneous brownfield environments. A common theme, as explained before in Section 2.3.2, was the replacement of the rigid automation pyramid with flatter, service-oriented communication models, typically using OPC UA as the de facto standard for peer-to-peer interaction (Faller & Höftmann 2018).

To manage complexity, researchers proposed structured, multi-layer IIoT frameworks that leveraged both OPC UA and REST APIs for data handling and provided middleware with adapters for legacy system integration (Saqlain & Piao et al. 2019; Trunzer & Calà et al. 2019; Cimino & Grazia Gnoni et al. 2023). In this context, Engelsberger & Greiner (2018) advanced a process-independent approach for dynamically reconfiguring service-oriented resources across Edge and Cloud infrastructures, enhancing system reliability through continuous, criteria-based service placement (S5, S6). A different, less disruptive approach was also explored, featuring logically centralized controllers that act in an advisory capacity. By creating a global system view, these controllers could provide intelligent routings (S5, S6, S7, S8) to existing MES or ERP systems without requiring a complete architectural overhaul (Lopez & Shao et al. 2018).

The Rise of Digital Twin Architectures. Around 2020, the DT concept gained significant traction, leading to the development of dedicated architectural frameworks. A notable contribution was the Six-Layer Architecture for Digital Twins (SLADTA), which provided a structured model from the physical system up to virtual emulation and simulation layers (Redelinghuys & Kruger et al. 2020; Redelinghuys & Basson et al. 2020). A key innovation within this line of research was the idea of aggregating multiple DTs to enable higher-level, system-wide decision-making, promoting modularity (S4), loose coupling (S7) and scalability (S2). This foundational work spurred further research into implementation specifics, such as web-based platforms for collaborative DT modeling and the use of edge-cloud hierarchies (Liu & Jiang

et al. 2020). The focus gradually shifted from high-level concepts, as proposed, e.g., by Stark & Fresemann et al. (2019) and Kritzinger & Karner et al. (2018), to detailed technical implementations and design evaluations. Human & Basson et al. (2021), for instance, evaluate the efficiency of data pipeline protocols like MQTT within these DT frameworks.

The Modern Era: Microservices, Choreography, and the AAS. The most recent contributions in the field of industrial IT architectures synthesize earlier trends by adopting modern IT paradigms to create decentralized, resilient, and interoperable ecosystems. A significant shift is the move from centrally orchestrated services to choreographed microservices with EDA. In this model, loosely coupled services (S7) react to events on an asynchronous message broker, enhancing flexibility and scalability (S2) (Pedro Lopes & Ibrahim et al. 2024). This evolution culminates in a mature conceptualization of the original SDM vision, structured into distinct layers for resources, modular solutions (S4), and a mediating core for orchestration (S8) and discovery (S5, S6) (Frick & Ellwein et al. 2024). Building on this role, recent work extends such AAS-centric, event-driven architectures beyond factory boundaries to enable Digital Product Passports (DPPs), where standardized AAS models and event sourcing designed for low latency (S3) are used to aggregate and expose production data across the product lifecycle in an interoperable and decentralized manner (Ajdinović & Strljic et al. 2024).

Crucially, these modern architectures elevate the AAS from a mere data payload to a central architectural pillar. The AAS becomes the fundamental data model and universal directory for the entire ecosystem, ensuring all resources and software-based solutions are discoverable (S5, S6) and their metadata is universally understandable, promoting loose coupling (S7). By integrating the AAS with technologies like containerization (S4), virtual PLCs and DTs (P3), and event-driven communication mechanisms (S3), these frameworks represent a holistic approach to building truly open and adaptable production systems.

In summary, industrial IT architectures have converged toward decentralized, service-oriented, and event-driven ecosystems, with the AAS emerging as a central enabler - while simultaneously revealing data integration as a remaining bottleneck.

3.3 Approaches to Automated Data Integration

The evolution of proposed solutions for data integration reveals a clear trajectory from foundational algorithmic approaches to structured, standard-driven frameworks, and most recently, to the disruptive potential of LLMs.

Foundational Approaches: Algorithms and Early Architectures. Early solutions for automated data integration were primarily algorithmic. The influential Cupid algorithm, for instance, established a hybrid method for schema matching that combined linguistic and structural

heuristics without a deep understanding of real-world concepts (Madhavan & Bernstein et al. 2001). While effective for its time, its reliance on predefined thesauri highlighted the need for richer semantic interpretation (I3). Concurrently, architectural patterns like SOA emerged to integrate heterogeneous applications through standardized protocols (I1) and data formats like XML (I2) (Qiu 2007). These foundational approaches marked a shift from isolated problems to system-level integration, but they lacked mechanisms for automated semantic matching and still required significant custom development.

Standard-Driven Frameworks. A significant trend in the last decade has been to enforce interoperability by design through standard-driven frameworks (I6, I7), with the AAS emerging as a key enabler for I4.0. Rather than retrospectively matching disparate schemas, this approach seeks to provide a common language and structure (I1, I2, partially I3). A series of works demonstrates the AAS's versatility across different contexts: it has been used as a communication backbone with OPC UA for Plug-and-Produce scenarios (Ye & Hong et al. 2021; Ye & Xu et al. 2023), applied in Manufacturing as a Service (Iñigo & Legaristi et al. 2022), and adapted for the process industry's engineering lifecycle (Grüner & Hoernicke et al. 2023). Its extensibility is further highlighted by conceptual extensions to model processes and even human workers as digital assets (Ellwein & Neumann et al. 2023; Cutrona & Bonomi et al. 2024). Collectively, these studies show the mature adoption of AAS as a foundational block for creating interoperable digital representations, often within SOA or EDA (Wilhelm & Niermann et al. 2024).

However, challenges remain, particularly in the integration of legacy object-oriented systems and due to the complexity of current AAS implementation tooling (Alexopoulos & Kalogeras et al. 2025). In practice, differences in metamodels between the AAS and established standards such as AutomationML or OPC UA information models further complicate integration (Alexopoulos & Kalogeras et al. 2025). Although numerous standardization initiatives for AAS Submodel Templates exist (see Section 2.3.4), neither semantic convergence (I3) nor widespread adoption and open accessibility of these standards (I7) has been achieved. This results in a heterogeneous landscape of existing AAS domain models (Metović & Maisch et al. 2026). Moreover, while concepts for Submodel Template validation have been proposed in the literature (Behrendt & Stamer et al. 2023; Nowshin & Mohamed et al. 2025), current AAS reference implementations, most notably Eclipse BaSyx (*Eclipse BaSyx – Industry 4.0 Operating System* 2025), do not provide native support for Submodel Template compliance validation (I6), further increasing the difficulty of integrating heterogeneous schemas (I3). Consequently, AAS adoption without mature tooling can paradoxically increase integration effort, leaving semantic integration (I3) as an insufficiently addressed challenge.

Formal Semantics: Ontologies and Model-Driven Engineering. To provide a formal semantic backbone, many integration approaches have leveraged ontologies and model-driven engineering. These methods use a global schema (I4), often defined by a formal ontology (I6), to generate consistent data models and automate integration tasks, creating a unified and unambiguous representation of a domain (Petrasch & Petrasch 2022; Pereira & Cainelli et al. 2023; Göppert & Grahn et al. 2023). By defining a shared understanding, they enable robust integration (I1, I2, I3) within a predefined scope, with reference architectures like RAMI 4.0 (see Section 2.3.2) serving as analytical tools to guide this process systematically (Wilhelm & Niermann et al. 2024).

However, these structured approaches share a critical limitation: the creation of ontologies requires significant manual effort and domain expertise (Alizadeh & Shahrezaei et al. 2019). Furthermore, the schema matching is typically predefined within the model, not performed dynamically at runtime (I5), making them rigid and less applicable to the diverse legacy systems found in real-world scenarios. Moreover available technologies for ontologies can be complex to implement (I5) and too slow for low latency applications (S3), as required in PPC (Leser & Naumann 2007, p. 287; Kang & Krishnaswamy et al. 2020).

Hybrid approaches that combine formal semantic models with statistical similarity techniques attempt to overcome this rigidity – for instance, Metović & Maisch et al. (2026) combine rule-based SPARQL pre-filtering with embedding-based similarity using RDF2vec, enabling tolerance to terminological variation while remaining grounded in ontological representations.

The Paradigm Shift: LLMs and Hybrid Integration. A recent and profound paradigm shift has been introduced by the advent of LLMs. Fernandez & Elmore et al. (2023) provide a visionary outlook, arguing that LLMs can overcome the semantic hurdles that have long challenged data integration. As explained before in Section 2.4.6, LLMs derive a deep, conceptual understanding from their vast training on text and code, allowing them to grasp real-world meaning and context. This capability is applied directly to the classic problem of schema matching in several papers. Parciak & Vandevoort et al. (2024) and Buss & Safari et al. (2025) both conduct experimental studies on using LLMs for schema matching (I3, I5) based solely on schema metadata (e.g., attribute names, descriptions). They both find that LLM-based approaches significantly outperform traditional methods. Their work also highlights the nuances of using this new technology: Parciak & Vandevoort et al. (2024) demonstrate that prompt design (the "task scope") is critical for performance, while Buss & Safari et al. (2025) focus on engineering techniques like sampling, bidirectional matching, and chunking to improve the scalability, reliability, and cost-efficiency of LLM-based matching. The superiority of LLMs in this context stems from their ability to infer semantic similarity

from natural language descriptions, a task where traditional algorithmic, such as Cupid from Madhavan & Bernstein et al. (2001), and even formal ontology-based approaches can be brittle.

However, the application of LLMs faces challenges including inconsistent outputs (“hallucinations”), prompt sensitivity, computational cost, and difficulty with complex multi-relational mappings, typically requiring a human-in-the-loop for verification (Buss & Safari et al. 2025). Pivotal work addresses the limited LLM performance by injecting a domain-specific ontology as context into the LLM’s prompt. This primes the model with necessary domain knowledge, dramatically improving its performance in semantic integration (Kayali & Wenz et al. 2024). This synergy exemplifies the future of integration: structured models like ontologies and AAS provide the verifiable and semantically rich backbone, while LLMs act as a flexible reasoning engine to automate semantic integration.

Enabled by modular, service-oriented, and loosely coupled industrial IT architectures (Wilhelm & Niermann et al. 2024; Schubert & Kuehner et al. 2023), LLM-based matching promises to be a crucial building block that covers semantic alignment within integration pipelines.

3.4 Approaches to Smart Production Planning and Control

The evolution of automated and adaptive PPC systems and algorithms within the context of I4.0 reveals a distinct trajectory from foundational interoperability towards highly autonomous, AI-driven control architectures. The development of these Smart Manufacturing approaches can be traced through several chronological and logical stages, highlighting a progressive maturation in both the scope of PPC tasks addressed and the degree of automation achieved.

Foundational Interoperability through SOA. Approaches for interoperability in the manufacturing domain focused at first on establishing integration at the machine level through SOA (I1, I2, P4). Early "Plug & Produce" approaches modeled manufacturing capabilities as discrete, discoverable "skills" exposed as services and orchestrated via OPC UA, automating low-level reconfiguration and execution tasks (P1) (Pfrommer & Stogl et al. 2015). This was complemented by model-integrated methods using the systems modeling language (SysML) (I6) to formally structure the logic of flexible production cells, promoting loose coupling (S7) and agnosticism (P2) (Fallah & Wolny et al. 2016). These works established that modularity (S4), service-orientation, and standardized communication mechanisms (I1, I2), such as OPC UA, are prerequisites for breaking down monolithic systems. However, their focus was primarily on enabling flexible execution rather than data-driven planning, addressing the foundational interoperability requirements of I4.0 but leaving significant gaps toward fully automated PPC, as highlighted by the review of Qin & Liu et al. (2016).

Real-Time Visibility via the DT. The next evolutionary step focused on creating a coherent, real-time virtual representation of the factory: a DT. A pivotal development was the automated creation of a DT directly from live CPS data (S3), using a service bus and protocols like MQTT for real-time streaming (Biesinger & Meike et al. 2019). This marked a shift from abstract services to a concrete, data-rich model of the production. In this context, the DT served as an advanced information aggregation (P3) and monitoring (P4) tool to improve planning accuracy.

More behavior-oriented DTs were scoped by the approaches of Overbeck & Graves et al. (2024) and Behrendt & Altenmüller et al. (2025), where data analysis pipelines are used to automatically generate material flow simulation models from shop floor event data. These works advanced the real-time data-capturing aspect of PPC and demonstrated the feasibility of automated creation of simulation-based DTs (P3, P4). However, automation was concentrated on creating the DTs themselves rather than enabling autonomous decision-making (P1) or controlling the real production system (P5).

Holistic Architectures and AI-Driven Decision Support. More recent efforts address the PPC process holistically, combining sophisticated control architectures with AI (Artificial Intelligence). This includes semi-heterarchical architectures that balance centralized planning with decentralized execution to manage complex material flows (Grassi & Guizzi et al. 2020). A key methodological shift was the systematic integration of Machine Learning (ML) to automate solving traditionally difficult PPC tasks (P1), such as automating dispatching in production control with Reinforcement Learning (Kuhnle & Kaiser et al. 2021), optimizing scheduling based on Monte Carlo Tree Search (Stricker & Lanza 2014) or simulation-based optimization (Krenczyk & Paprocka 2023), or predicting production yield under uncertainty (Oluyisola & Bhalla et al. 2022). By fusing heterogeneous data and applying simulations and ML models, these systems function as intelligent decision-support tools, augmenting human planners rather than replacing them (P6). Moreover, these approaches can be categorized as focusing PPC systems (see Section 2.1.2), solving a distinct PPC task instead of a complete PPC model. Such modular systems serve as a cornerstone for service-oriented PPC systems, where individual planning approaches can be flexibly orchestrated to realize reconfigurable PPC models (P3, S8).

Towards Autonomy: AAS, Multi-Agent Systems, and Real-Time Replanning. The most recent works demonstrate a maturation of these concepts, often leveraging the AAS as a standard for creating interoperable DTs. This foundation enables more advanced automation (P1), for instance by combining AAS with Knowledge Graphs for semantic modeling of production logistics (Bozkurt & Schulz 2023). A significant step towards autonomy and task-agnosticism (P1, P2) is the use of AAS to model agents within Multi-Agent Systems that

can autonomously select from a portfolio of optimization methods, including heuristics and Deep Reinforcement Learning, to solve scheduling problems (Siatras & Bakopoulos et al. 2023). An important precursor to such autonomous approaches is the formalization of skills and assembly processes to enable automated skill-based resource selection. Brovkina & Riedel (2021) propose a graph-based, AAS-aligned skill and process model that enables task-agnostic (P2) matching of abstract assembly processes to heterogeneous machine skills, supporting automated assembly line design and resource allocation. The culmination of these trends is seen in distributed MES architectures that integrate DTs (P3, P4) with AI for on-the-fly replanning based on the Planning Domain Definition Language (PDDL) (Vyskočil & Douda et al. 2023). By using AI for autonomous plan generation in response to real-time events like machine failures, these systems achieve a high degree of automation (P1), tackling the ultimate PPC challenge of dynamic adaptation.

In summary, automated PPC has evolved from low-level execution support to predictive planning, optimization, and dynamic replanning under real-time disturbances. The reviewed approaches consistently demonstrate that advanced PPC algorithms require modular, loosely coupled architectures and standardized, interoperable data to be effective. Yet, the orchestration of heterogeneous PPC services into coherent, reconfigurable control loops remains an open challenge.

3.5 Research Deficit and Research Questions

The preceding analysis of the state of the art in industrial architectures, automated data integration, and smart PPC reveals a clear and consistent trajectory. The evolution from rigid, monolithic systems towards modular, interconnected, and intelligent manufacturing environments, as envisioned by I4.0 and required by RMS, is well-documented. Significant progress has been made in leveraging service-orientation, standardizing data models with concepts like the AAS, and exploring advanced algorithms, including nascent applications of LLMs for data integration, to tackle specific challenges.

Figure 3.2 summarizes the evaluation of the presented approaches concerning the defined evaluation criteria. The analysis reveals that despite these advances, critical deficits and unresolved gaps persist, preventing the realization of a truly integrated, automated, and adaptive PPC system. The current research landscape is characterized by solutions that address isolated challenges, while a comprehensive, end-to-end framework remains elusive, as Figure 3.2 visualizes. An approach may score highly in one dimension (e.g., architecture) while remaining limited in others (e.g., PPC automation), reflecting the modular and incremental nature of current research. The primary deficits can be summarized across three interconnected dimensions.

Evaluation Criteria: ○ not fulfilled, ◐ partially fulfilled, ● fulfilled		(1) Software Architecture & Service Orientation								(2) Data Integration & Interoperability							(3) Automated PPC					
		S1	S2	S3	S4	S5	S6	S7	S8	I1	I2	I3	I4	I5	I6	I7	P1	P2	P3	P4	P5	P6
	Approaches	Architecture Paradigm	Scalability	Latency	Modularity	Discoverability	Location Transparency	Loose Coupling	Orchestrability	Technical	Syntactic	Semantic	Integration Approach	Automated Integration	Standardized Schemas	Openness	Automation of PPC	Task Agnosticism	Virtualization	Monitoring Real System	Controlling Real System	Human-in-the-Loop
Industrial Architectures	Ajdinović et al.(2024)	SOA, EDA	◐	◐	●	◐	◐	●	○	●	●	◐	HS	○	●	○	○	○	○	●	○	◐
	Cimino et al. (2023)	SOA, EDA	◐	◐	●	○	○	◐	○	◐	◐	◐	HS	○	◐	○	○	○	●	○	○	●
	Engelsberger & Greiner (2018)	SOA	◐	◐	●	○	●	●	●	●	○	○	HS	○	◐	○	●	◐	○	○	●	○
	Faller & Höftmann (2018)	SOA	◐	○	●	◐	●	●	◐	●	◐	◐	HS	○	●	◐	○	◐	○	●	◐	◐
	Frick et al. (2024)	SOA, SDM	◐	●	●	●	●	●	●	●	◐	○	GS	○	●	●	○	◐	●	○	○	◐
	Human et al. (2021)	SOA, EDA	●	◐	●	○	○	◐	○	●	○	○	HS	○	○	○	○	○	○	◐	○	○
	Lechler & Verl (2017)	SOA, SDM	◐	●	●	○	●	●	●	●	◐	○	GS	◐	●	◐	○	○	●	○	○	◐
	Liu et al. (2022)	SOA, EDA	○	○	◐	◐	◐	◐	◐	◐	◐	○		○	◐	○	○	○	◐	●	●	●
	Lopes et al. (2024)	MSA	●	◐	●	◐	●	●	◐	●	◐	○	GS	○	◐	◐	○		○	●	○	●
	Lopez et al. (2018)	SOA	◐	◐	●	●	●	●	◐	●	◐	○	GS	○	●	◐	◐	◐	●	●	○	●
	Pérez et al. (2015)	SOA, MDA	◐	◐	●	◐	◐	◐	◐	●	●	◐	HS	○	●	◐	◐	◐	○	●	◐	◐
	Redelinghuys et al. (2020a)	SOA, EDA	◐	◐	●	○	◐	●	◐	●	◐	◐	HS	○	●	◐	○	○	●	●	◐	●
	Redelinghuys et al. (2020b)	SOA, EDA	◐	○	●	◐	◐	●	◐	●	◐	◐	HS	○	●	◐	○	○	●	●	◐	●
	Saqlain et al. (2019)	SOA	◐	●	●	◐	●	●	◐	●	◐	◐	GS	○	●	◐	○	○	○	●	◐	●
	Thames & Schaefer (2016)	SOA, SDM	●	○	●	○	◐	●	◐	○	○	○	GS	○	○	◐			○	◐	●	●
	Trunzer et al. (2019)	SOA	●	●	●	◐	◐	●	●	●	◐	◐	GS	○	●	◐	○	○	○	◐	◐	◐
Automated Data Integration	Buss et al. (2025)									○	●	●	PTP	●								◐
	Cutrona et al. (2024)	SOA	◐	◐	●	◐	◐	●	◐	●	●	◐	HS	○	●	●	○	○	◐	●	◐	●
	Ellwein et al. (2023)				●	◐	◐	●		◐	●	●	GS	○	●	●						
	Göppert et al. (2023)	SOA	◐	◐	●	◐	●	●	◐	○	●	◐	GS	◐	◐	●	◐	◐	◐	◐	◐	●
	Grüner et al. (2023)	SOA	◐	◐	●	◐	●	●	●	○	◐	◐	GS	○	●	●	◐	◐	◐	◐	○	●
	Iñigo et al. (2022)	SOA, EDA	●		●	●	◐	●	●	●	●	◐	GS	○	●	●	◐	○	○	●	●	◐
	Kayali et al. (2024)									○	◐	●	PTP	●								
	Madhavan et al. (2001)									◐	◐	●	PTP	●	◐							◐
	Metović et al. (2026)		◐		◐	◐	●	◐	○	◐	●	●	GS	◐	●	●	○	○	○	○	○	○
	Parciak et al. (2024)									○	◐	●	PTP	●	○							○
	Pereira et al. (2023)	SOA, EDA	○	○	●	○	◐	●	○	●	●	◐	GS	○	●	◐	○		●	●	●	●
	Petrasch & Petrasch (2022)	MDA	◐	◐	●	●	●	●	◐	●	○	◐	HS	○	●	○	○	○	○	○	○	●
	Qiu (2007)	SOA	◐	◐	●	◐	◐	●	◐	●	●	◐	HS	○	◐	○	○	○	○	◐	◐	○
	Schubert et al. (2023)	SOA	◐	○	●	◐	●	●	◐	●	◐	◐	HS	○	◐	○	○	○	○	◐	○	◐
	Wilhelm et al. (2024)	SOA, EDA	◐	◐	●	◐	●	●	●	●	◐	◐	HS	○	●	●	◐	◐	●	●	●	●
	Ye et al. (2021)	SOA	◐	◐	●	●	●	●	●	●	●	◐	HS	○	●	●	○	○	○	●	●	●
	Ye et al. (2023)	SOA	◐	◐	●	●	●	●	●	●	●	◐	HS	○	●	●	○	○	●	○	○	●
Smart PPC	Biesinger et al. (2019)	EDA	◐	●	◐	○	○	◐	○	●	●	◐	PTP	○	◐	◐	●	○	○	●	○	○
	Brovkina & Riedel (2021)	MDA			●	◐	◐	●	○	●	◐	◐	HS	○	●	○	◐	●	○	○	◐	○
	Bozkurt & Schulz (2023)	SOA	◐	○	●	◐	◐	●	◐	●	●	●	GS	○	●	○	◐	◐	○	◐	○	●
	Fallah et al. (2016)	SOA	◐	○	●	◐	●	●	○	●	◐	◐	PTP	○	●	○	◐	◐	○	●	●	●
	Grassi et al. (2020)	Hybrid	◐	◐	●	◐	●	●	○	●	◐	●	GS	○	●	○	●	●	◐	●	●	●
	Kuhnle et al. (2021)									○	○	○	HS	○	◐	○	●	◐	●	○	○	○
	Krenczyk & Paprocka (2023)	SOA, EDA	◐	◐	◐	○	○	◐	◐	◐	○	○	GS	○	○	○	◐	○	●	●	◐	○
	Oluyisola et al. (2022)	MSA	●	◐	●	○	◐	●	◐	◐	○	○	GS	○	○	◐	◐	◐	○	●	◐	●
	Overbeck et al. (2023)	Monolith								◐	○	○	GS	○	○	○	○	○	●	●	○	○
	Behrendt et al. (2025)	Monolith								◐	○	○	GS	○	○	◐	○	○	●	●	○	○
	Pfrommer et al. (2015)	SOA	◐	◐	●	●	●	●	●	◐	◐	○	HS	○	●	◐	●	●	○	●	●	●
	Siatras et al. (2023)	SOA	◐	●	●	◐	◐	●	◐	●	◐	◐	GS	○	●	◐	●	●	○	◐	◐	◐
	Stricker et al. (2014)									○	○	○	HS	○	◐	○	●	◐	●	○	○	○
	Vyskočil et al. (2023)	SOA	◐	◐	●	◐	●	●	●	●	●	○	HS	○	●	●	●	●	●	●	◐	○

Figure 3.2: Overview of the evaluation of reviewed literature with regard to the defined evaluation criteria from Section 3.1.

1. Architectural Deficiencies in Service-oriented Implementations. While many approaches have embraced SOA and MSA as architectural paradigms (S1) to enhance interoperability, their implementation in the industrial context is often incomplete.

Key tenets of a mature SOA, such as dynamic service discovery (S5), location transparency (S6), and true loose coupling (S7), are frequently overlooked in runtime-ready implementations or only conceptually considered. This results in architectures that are 'service-oriented' in name but remain brittle, requiring manual reconfiguration and point-to-point integration when new services are introduced or existing ones are modified. This severely limits the scalability (S2) and flexibility (S4-S7) required to manage the dynamic nature of an RMS.

2. The Persistent Challenge of Semantic Data Integration. Progress in industrial data integration is evident, but it is largely confined to the technical (I1) and syntactic (I2) levels, achieved through standardized protocols and data formats. The central challenge of semantic integration (I3) is either not addressed or only partially addressed by many approaches. Many of the reviewed approaches rely on HS and thus shift semantic integration into an upfront engineering phase, assuming fixed and homogeneous semantics. Although theoretically sound, the HS approach is often infeasible or impractical in industrial environments, which are characterized by a multitude of legacy systems and proprietary, customized schemas, making such upfront alignment difficult to realize in practice (I4). Consequently, semantic integration remains a resource-intensive and predominantly manual task in most reviewed approaches (I5).

The automation of this semantic mapping process therefore holds immense potential to reduce integration effort and increase flexibility. Recent developments that utilize LLMs for semantic integration have been identified as promising foundations for automating semantic integration tasks (I5). In the reviewed literature, most approaches employ standardized schema definition languages (I6) for expressing and integrating data. However, tooling for AAS schema definition and validation is not yet mature, limiting the practical applicability of AAS-based models. Moreover, approaches for integrating legacy systems with AAS-based formats at low effort (I1, I2) are largely missing, resulting in substantial adoption barriers. Finally, many of the utilized domain models are not publicly available (I7), which further restricts the transferability and reuse of the proposed approaches.

3. Lack of Task-Agnosticism in Smart PPC Systems. In the development of smart PPC systems, research has successfully automated specific tasks, such as scheduling or real-time monitoring (P1). However, a holistic approach that automates the entire PPC process chain and, crucially, manages the orchestration of diverse services (P3-P5, S8), is rare. The inherent complexity of coordinating multiple, distributed services to fulfill a higher-level business objective is often underestimated. Furthermore, existing systems are typically task-specific: designed to orchestrate a known set of services to perform a predefined function. They often lack a task-agnostic design (P2) that would allow them to dynamically integrate and orchestrate new, previously unknown PPC services without requiring changes. This deficit is

a fundamental barrier to achieving the plug-and-play reconfigurability required for RMS and I4.0. Also, the interplay of a fully automated PPC system (P1) with a Human-in-the-Loop (P6) that orchestrates high-level tasks is not examined in detail.

Beyond individual shortcomings in specific criteria, a more fundamental insight emerges from the combined evaluation of the reviewed approaches. The primary deficits do not arise from the absence of isolated capabilities, but from the lack of *co-occurrence* of critical capabilities within a single, coherent framework. As visualized in Figure 3.2, many approaches achieve high maturity in one dimension while remaining weak in others, leading to structurally fragmented solutions.

A recurring pattern is the combination of advanced, service-oriented architectures (S4-S7) with limited semantic integration capabilities (I3, I5). While such architectures are theoretically scalable and composable, the lack of automated semantic alignment results in high integration effort in practical, heterogeneous environments, significantly constraining their real-world applicability. Similarly, several approaches demonstrate strong virtualization and digital twin capabilities (P3, P4), yet lack orchestration mechanisms (S8) and task-agnostic control logic (P2), preventing the realization of closed-loop PPC systems in which virtual models can be systematically linked to planning, decision-making, and execution.

Conversely, some smart PPC approaches achieve a high degree of automation for specific tasks (P1), but are built upon closed or non-standardized data models (I7) and tightly coupled integrations. This limits their transferability, reuse, and generalization beyond narrowly defined application scenarios. In such cases, automation is achieved at the expense of openness and interoperability, prohibiting the scalable deployment of these solutions across different factories, domains, or system landscapes.

These missing combinations highlight that the core challenge is not the development of individual architectural patterns, integration techniques, or optimization algorithms in isolation, but their systematic integration into a unified, flexible, and extensible PPC framework.

In synthesis, the current research landscape is fragmented not because of a lack of individual solutions, but because essential architectural, integration, and control capabilities are rarely combined in a consistent manner. There is no existing framework that provides a continuous and flexible solution for PPC by seamlessly integrating a truly service-oriented architecture, an automated data integration capability, and task-agnostic orchestration of PPC tasks to realize a PPC system suitable for RMS. To address these interconnected deficits, this thesis poses the following research questions that will guide research to resolve the identified gaps:

1. How can a technologically consistent and holistic framework for service-oriented PPC be designed and realized for an industrial context?

This question targets the architectural deficit of current PPC systems. It aims to define a formal framework and software architecture that can effectively implement the core principles of SOA (see Section 2.5.2) for PPC. Answering this question will provide the structural foundation needed to support a dynamic and scalable system.

2. How can the integration of heterogeneous information systems be automated to the greatest extent possible, enabling seamless, end-to-end information flows?

This question directly confronts the data integration deficit, particularly the manual and costly nature of semantic integration. It seeks to investigate and develop methods for automated schema matching and transformation that go beyond predefined, static mappings. This involves exploring how modern LLM-driven techniques can be leveraged to interpret and align disparate data models on-the-fly, thus enabling rapid and low-effort integration of systems.

3. What principles and methods are required to define and orchestrate a reconfigurable and task-agnostic PPC system?

This question addresses the deficit of task-agnosticism and orchestration in current PPC systems. Its goal is to develop a methodology that allows the PPC system to integrate and coordinate previously unknown services without requiring manual reprogramming. This involves formalizing the description of services and their capabilities, and creating a control loop engine that can interpret these descriptions to dynamically build and execute complex, multi-step production processes. Answering this question is key to achieving plug-and-play functionality for production capabilities and enabling the reconfigurability of PPC systems required by RMS.

The methodological foundation for addressing these research questions is presented in the following chapter, before Chapter 5 develops the holistic framework for service-oriented PPC that resolves the identified research gaps.

4 Research Methodology

The objectives formulated in Section 1.3 target the design of a service-oriented PPC framework that combines architectural, integration, and orchestration capabilities into a coherent and empirically validated artifact. Addressing such an objective requires a methodology that supports the iterative construction of a complex software artifact, its grounding in existing knowledge, and its evaluation under realistic industrial conditions. This chapter therefore introduces the underlying research paradigm (Section 4.1), describes the literature review approach that underlies the preceding analysis of the state of the art (Section 4.2), and details the validation strategy applied to assess the proposed framework (Section 4.3). Together, these elements establish the methodological foundation for the conceptual, technical, and empirical contributions of this thesis.

4.1 Research Paradigm and Process Model

The research conducted in this thesis is positioned within the paradigm of DSR, as introduced by Hevner & March et al. (2004) and operationalized by Peffers & Tuunanen et al. (2007). In contrast to purely descriptive or behavioral research, DSR is explicitly concerned with the construction and evaluation of *artifacts*, i.e., constructs, models, methods, or instantiations, that solve identified classes of problems in a relevant application domain (Hevner & March et al. 2004). This characteristic aligns with the objective of this thesis, which is neither to merely describe the limitations of existing PPC systems nor to formulate purely theoretical models, but to design, instantiate, and evaluate a coherent framework that addresses concrete shortcomings of current PPC architectures.

DSR is structured around the interplay of three contributing domains: the *environment*, which defines the relevance of a research problem; the *knowledge base*, which provides theoretical foundations and established methods; and the *design science activities*, in which artifacts are built and evaluated (Hevner & March et al. 2004). For this thesis, the environment is constituted by manufacturing enterprises operating reconfigurable production systems in the context of I4.0, with their characteristic heterogeneity of IT and OT landscapes. The knowledge base comprises the theoretical foundations of PPC, SOA, data integration, and modern enabling technologies such as LLMs and AAS, as elaborated in Chapter 2 and Chapter 3. The design science activities encompass the conceptualization, implementation, and evaluation of the proposed framework.

To structure the design science activities, the six-phase process model of Peffers & Tuunanen et al. (2007) is adopted, as depicted in Figure 4.1. The phases are mapped to the chapters of this thesis as follows:

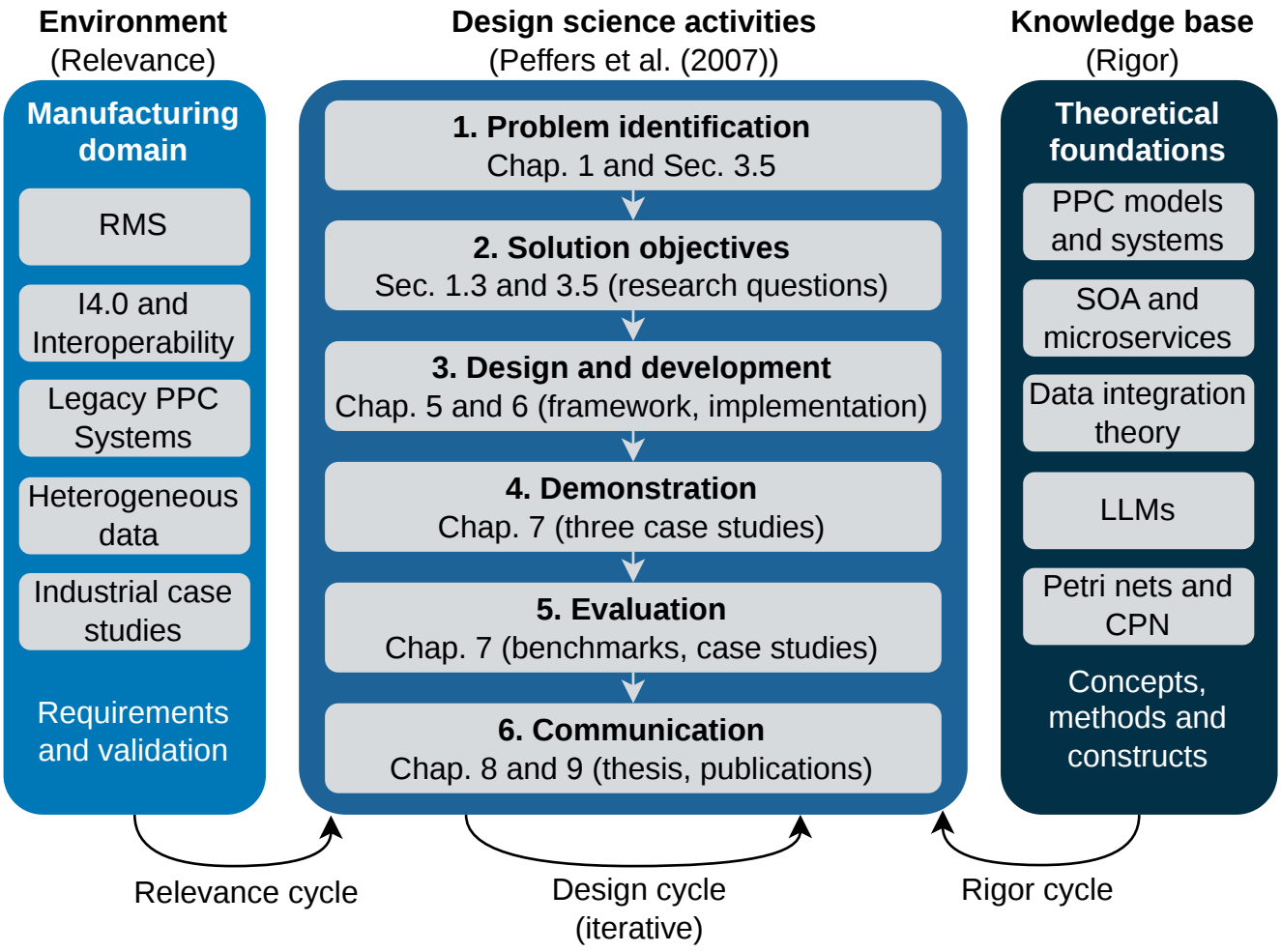

Figure 4.1: Research methodology of this thesis, mapping the DSR activities of Peffers & Tuunanen et al. (2007) to the chapters of this thesis, embedded in the three-cycle view of Hevner & March et al. (2004).

- **Problem Identification and Motivation** (Chapter 1, Section 3.5): The architectural rigidity of current PPC systems, the manual effort of semantic integration, and the lack of task-agnostic orchestration are identified as the central problems that constrain the realization of reconfigurable PPC. These problems are derived from both industrial relevance and the structured analysis of the state of the art presented in Chapter 3.

- **Definition of the Objectives of a Solution** (Section 1.3, Section 3.5): Building on the identified problems, three research questions are formulated. They define the objectives of the artifact in terms of architectural consistency, automated semantic integration, and task-agnostic orchestration.

- **Design and Development** (Chapter 5, Chapter 6): The framework is conceptualized as a service-oriented architecture combining a decentralized Middleware, a Service Registry, an LLM-based SIS, and a Petri net-based OE. The conceptual design is

subsequently instantiated as a functional software prototype, providing the artifact required for evaluation.

- **Demonstration** (Chapter 7): The applicability of the artifact is demonstrated across three progressively complex case studies, ranging from a modular disassembly station to a distributed learning factory and an industrial automotive scenario.
- **Evaluation** (Chapter 7): The framework is evaluated through a combination of controlled component benchmarks and case study analyses. The evaluation strategy is detailed in Section 4.3.
- **Communication** (Chapter 8, Chapter 9): The findings are synthesized, contextualized within the state of the art, and disseminated through this thesis and accompanying peer-reviewed publications.

In line with the three-cycle view of DSR proposed by Hevner & March et al. (2004), the research is driven by an iterative interplay of the *Relevance Cycle*, which ensures that the artifact addresses problems of practical importance through industrial case studies, the *Rigor Cycle*, which grounds the design in established theory and contributes evaluated artifacts back to the knowledge base, and the *Design Cycle*, which iteratively refines the artifact through alternating construction and evaluation. This iterative character is reflected throughout the development of the framework, where insights from preliminary implementations and early case studies informed successive refinements of the architecture, the SIS, and the OE.

4.2 Literature Review Approach

The state of the art presented in Chapter 3 fulfills two complementary functions within the chosen research paradigm: it grounds the design activities in the existing knowledge base, satisfying the rigor requirement of DSR, and it provides the systematic justification for the research deficit and the derived research questions.

The review was structured along the three thematic dimensions that correspond to the research questions: industrial software architectures, automated data integration, and smart PPC. For each dimension, relevant publications were identified through keyword-based searches in established scientific databases, in particular Scopus, IEEE Xplore, ACM Digital Library, and Web of Science. The search was complemented by forward and backward citation tracking starting from key reference publications to ensure that influential and recently published works were included. The temporal focus was set on publications from 2015 onward to reflect the rapid technological evolution in the relevant fields, while seminal earlier contributions were retained where they remain foundational.

To enable a structured and transparent analysis of the identified literature, the multi-dimensional evaluation framework introduced in Section 3.1 was applied, comprising 21 criteria in three categories. Each criterion is derived from requirements identified in the theoretical foundations of Chapter 2, ensuring traceability between the analytical lens and the underlying conceptual basis. Applying this framework consistently to all reviewed approaches enabled a comparative assessment that not only highlights individual strengths and weaknesses, but also reveals patterns of co-occurrence and absence of capabilities across the research landscape.

4.3 Validation Strategy

The evaluation of the proposed framework follows a two-stage strategy designed to balance the depth of controlled experimentation with the breadth of practical relevance. This strategy combines quantitative *component validations* with qualitative and quantitative *system-level case studies*, addressing both the rigor and the relevance dimensions of DSR. The overall structure, scope, and mapping of validation activities to the research questions are detailed in Section 7.1.

Component Validations. In the first stage, two novel core components of the framework for schema integration and process orchestration, are evaluated in isolated, controlled environments. This stage serves to quantitatively characterize the technical properties of each component, including mapping accuracy, runtime overhead, and scalability, independently of the complexities introduced by the full system. Established benchmarks and reproducible experimental setups are employed where available, enabling a comparison with existing approaches and providing the empirical basis for assessing the technical viability of the core mechanisms.

System-Level Case Studies. Building on the component validations, three case studies of increasing complexity are conducted to evaluate the integrated framework in industrially relevant settings. The case studies cover a modular disassembly station, a distributed learning factory environment, and an industrial automotive scenario. They are designed in line with the case study methodology of Yin (2018) and selected to systematically vary along several relevant dimensions, including the degree of heterogeneity, the number and type of integrated services, the spatial distribution of components, and the complexity of the orchestrated PPC processes. This deliberate variation supports analytical generalization beyond the individual application contexts.

Triangulation and Limitations. By combining controlled benchmarks with multi-context case studies, the validation strategy applies methodological triangulation: quantitative metrics, such as latency, F1-scores for mapping quality, and economic indicators such as Discounted Cash Flow (DCF), are complemented by qualitative observations regarding integration effort and

applicability in brownfield environments. This triangulation strengthens the credibility of the findings while mitigating the limitations inherent to either method in isolation. It is explicitly acknowledged that the empirical findings remain bound to the specific configurations of the considered case studies; the discussion in Chapter 8 therefore reflects on the generalizability of the results and the conditions under which the framework is expected to be applicable beyond the studied scenarios.

The methodological foundation established in this chapter provides the basis for the conceptual design of the framework in Chapter 5, its instantiation in Chapter 6, and its empirical evaluation in Chapter 7.

5 Framework for Service-oriented Production Planning and Control

As established in the preceding analysis of the state of the art, current PPC systems face significant challenges in the era of I4.0 and RMS. To address these challenges, this chapter proposes a technologically consistent and holistic framework for service-oriented PPC. The framework is built upon three foundational pillars designed to work in synergy:

1. A formalized, service-oriented architecture for interoperability based on SDM principles (Section 5.2).
2. A novel integration methodology featuring automated semantic integration via LLMs (Section 5.3).
3. A task-agnostic orchestration based on an extended Petri net formalism for dynamic service composition (Section 5.4).

By integrating these pillars into a unified system, the framework aims to provide a robust and adaptable foundation for next-generation PPC in reconfigurable manufacturing contexts.

This chapter first gives an overview in Section 5.1 of the conceptual design of the proposed framework, the rationale behind its core design decisions, and the mechanisms through which its components interact. Next, the reference architecture is presented in Section 5.2, detailing the layered structure, functional modules, and integration points. Special focus is thereby given to formalizing service-oriented PPC as a rigorous scientific framework for the later sections. This is followed by Section 5.3, which introduces the LLM-based semantic integration approach, explains how LLM-based semantic mapping can be utilized, and details how human feedback can be considered. Finally, Section 5.4 describes the task-agnostic orchestration component, outlining the schema-aware Petri net formalism, its execution semantics, and the mechanisms that ensure capability-driven, dynamic service composition. Together, these sections provide a comprehensive description of the framework's conceptual underpinnings, preparing the ground for the technical implementation presented in the subsequent chapter.

5.1 Overview of the Overall Concept and Design Decisions

This section provides a high-level overview of this integrated concept, outlining the core components and the key design decisions that underpin the architecture. The goal is to present a cohesive vision that directly resolves the identified limitations of existing approaches, thereby enabling the flexibility, interoperability, and dynamic reconfigurability required by modern manufacturing.

5.1.1 Overview of the Overall Concept

The proposed architectural framework is built upon a three-layered, service-oriented architecture designed for SDM, as depicted in Figure 5.1. This structure enforces a clear separation between the underlying physical and digital infrastructure, the data integration mechanisms, and the business application logic. This separation promotes loose coupling, simplifies the management of heterogeneous systems, and enables dynamic reconfigurability.

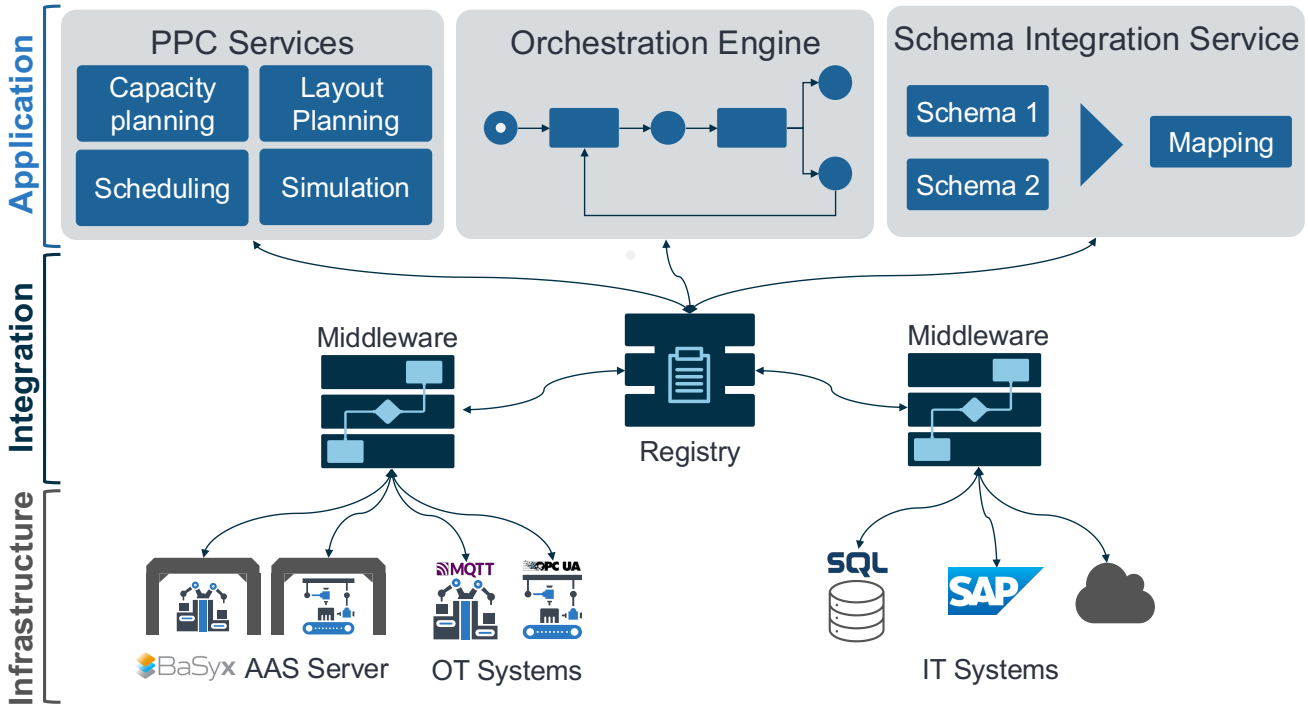

Figure 5.1: Overview of the proposed three-layered architecture for SDM with the core components.

The **Infrastructure Layer** represents the foundation of the system, encompassing all physical and digital assets. It includes heterogeneous OT systems on the shop floor (e.g., machines, AAS servers, IoT devices) and corporate IT systems (e.g., ERP, databases, cloud services) that serve as data sources and sinks.

The core of the architectural innovation lies within the **Integration Layer**, which decouples Infrastructure and Application Layer and facilitates seamless communication. The Integration Layer consists of two primary building blocks: decentralized *Middleware* components and a central *Service Registry*.

The *Middleware* serves as the decentralized backbone for data integration. It provides a standardized methodology to resolve technical, syntactic, and semantic heterogeneity, enabling interoperable communication between disparate systems in the Infrastructure Layer

without modifying them. By acting as a unified integration abstraction, it supports various communication paradigms (see Section 2.3.3.1) and integration patterns (see Section 5.2.2.5) to accommodate diverse business needs.

The *Service Registry* acts as a central directory for all services and Middleware instances. By enabling dynamic service discovery at runtime, it ensures location transparency and loose coupling, which is critical for reconfiguring the information flows required in complex PPC use cases.

The **Application Layer** hosts the modular software services that implement the business logic for PPC. Alongside standard PPC Services that cover distinct PPC tasks, such as capacity planning or scheduling, this layer also contains two foundational services that enable advanced automation and flexibility:

- **Schema Integration Service (SIS) (Section 5.3):** An LLM-based tool that automates the creation of semantic integrations for the Middleware, significantly accelerating the integration of disparate services.
- **Orchestration Engine (OE) (Section 5.4):** A task-agnostic engine that executes business logic by interpreting formal process models (Petri nets), enabling the dynamic composition and reconfiguration of PPC process models.

Although tightly coupled to the Integration Layer, SIS and OE focus on supporting schema integration and process orchestration rather than covering data transport or protocol mediation. They therefore belong to the Application Layer, where business logic functionality is defined. While the Integration Layer provides all mechanisms required for interoperable data exchange, SIS and OE increase the usability and automation integration tasks by generating integration logic and orchestrating its execution. They are not mandatory for basic operation, but enable advanced usage patterns such as automated semantic integration, process execution, and runtime reconfiguration.

5.1.2 Using the Framework

The practical realization of a reconfigurable PPC system using the proposed framework follows a structured two-phase process, as illustrated in Figure 5.2. The process is divided into a *Preparation and Integration (1)* phase, which focuses on establishing interoperable integration services, and a *Modeling and Execution (2)* phase, which focuses on declarative PPC process modeling and its execution by the OE. This separation enables the modification of PPC business logic without requiring changes to the underlying software components.

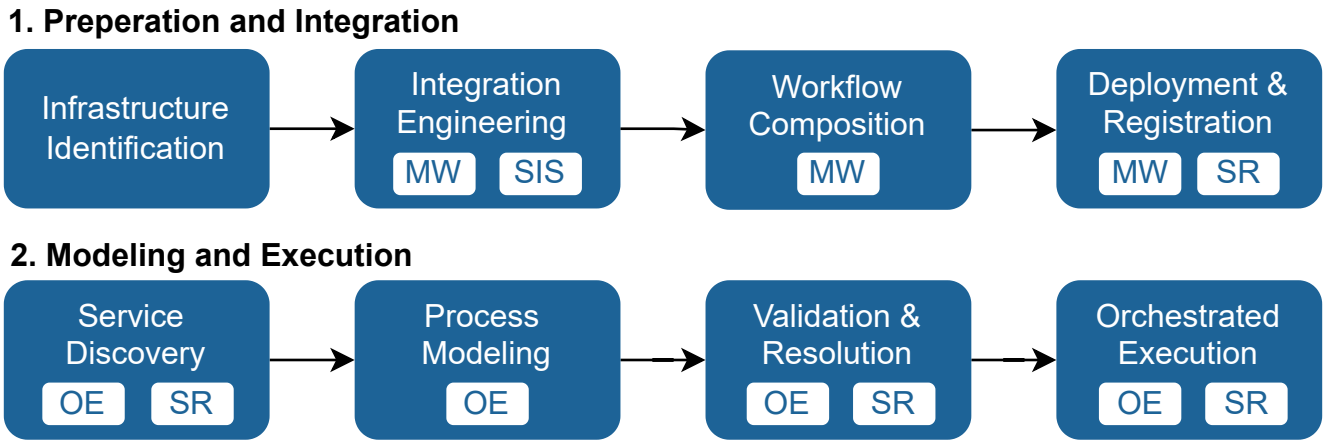

Figure 5.2: Overview of the process to utilize the framework for service-oriented PPC to create adaptive PPC systems with reference to the utilized software components in each step.

Preparation and Integration. The first phase establishes the technical and semantic foundation required for service-oriented PPC. It begins with *Infrastructure Identification*, where relevant data sources, sinks, and PPC-relevant systems within the production IT/OT landscape are identified. For each system, a corresponding Middleware instance (denoted as MW in Figure 5.2) is instantiated to interface with the identified infrastructure components and expose its data or functionality as a service.

In the subsequent *Integration Engineering* step, these Middleware instances are configured to handle technical, syntactic, and semantic heterogeneity. This step is supported by the LLM-based SIS, which automates the generation of semantic integration between heterogeneous schemas of the identified infrastructure components to reduce manual integration efforts.

Next, during *Workflow Composition*, individual integration components are combined into executable *Workflows* (more detailed description in Section 5.2.2.3). These Workflows encapsulate reusable integration logic in the Middlewares with an abstraction to a higher-level functionality rather than isolated transformation steps.

Finally, in the *Deployment and Registration* step, the Middleware instances are deployed as modular, containerized services and registered in the central Service Registry (denoted as SR in Figure 5.2). This registration makes all Workflows of the Middlewares discoverable for use during modeling and execution.

Modeling and Execution. Once the integration layer is in place, PPC process models can be defined and enacted dynamically. The phase starts with *Service Discovery*, where the OE queries the Service Registry to retrieve the set of available Workflows of the Middleware instances and independent PPC services.

Based on the discovered services, the PPC logic is specified in the *Process Modeling* step as a formal Petri net within the OE. This model declaratively defines both the control flow and the information flow between services, independent of their concrete implementations.

Before execution, the model undergoes *Validation and Resolution*. The OE verifies schema conformance and service availability for all modeled interactions. If schema mismatches or missing integrations are detected, the OE consults the SR for suitable Mappers previously generated by the SIS that can resolve these inconsistencies.

In the final *Orchestrated Execution* step, the OE executes the Petri net and dynamically routes control and data to the required services and Middleware instances via the Service Registry. In doing so, the OE enacts the PPC process exactly as defined by the model.

The central benefit of this approach lies in its reconfigurability. Changes to PPC logic are realized by modifying the Petri net model rather than redeploying software components. The OE can stop execution, load an updated model, and resume operation with the new logic, often without system downtime. Likewise, individual service implementations can be exchanged transparently. As long as a newly deployed service adheres to the expected interface, it can immediately be utilized in existing process models. If required, new integrations can be generated and registered, enabling runtime reconfiguration of PPC functionalities.

Achieving this degree of flexibility requires a robust and carefully designed technical foundation. The following section therefore presents the detailed reference architecture of the framework and explains how its components interact to form a cohesive and adaptable system.

5.2 Reference Architecture of the Integration Layer

Having outlined the conceptual overview of the architectural framework in the previous section, this section first introduces the architectural principles of the Integration Layer in Section 5.2.1. It then elaborates on the specific components of the Integration Layer, namely the Middleware (Section 5.2.2) and the Service Registry (Section 5.2.3). Finally, the reference architecture is embedded in a formalization of service-oriented PPC services in Section 5.2.4, establishing the theoretical groundwork for the subsequent discussions on data integration and orchestration.

5.2.1 Architectural Principles of the Integration Layer

The Integration Layer is designed according to a set of architectural principles that promote modularity, scalability, and reconfigurability, as motivated by the evaluation criteria presented in Section 3.1. In particular, modularity (S4) and scalability (S2) are supported directly, while

reconfigurability is enabled through discoverability (S5), location transparency (S6), and orchestrability (S8).

Central to this design is the enforcement of mediated communication, whereby all interactions between application services and infrastructure systems are routed exclusively through standardized interfaces provided by the Integration Layer. This strict decoupling (S7) renders services portable, independently deployable, and agnostic (P2) to underlying technologies, enabling software-defined composition and reconfiguration of PPC workflows.

Architectural flexibility is achieved through a deliberate combination of decentralized Middleware instances and a central Service Registry, balancing local autonomy and flexibility with global discoverability and coordination. This structure allows integration logic to be deployed close to data sources while maintaining a consistent, system-wide view of available services and workflows.

The architecture is intentionally designed to support multiple architectural paradigms (S1) at the Integration Layer, including SOA, MSA, and EDA approaches. Accordingly, the Integration Layer supports both push- and pull-based communication paradigms (see Section 2.3.3.1). In addition, both virtualized and materialized data integration patterns (I4), as introduced in Section 2.4.4, are supported, enabling the Integration Layer to be tailored to diverse PPC use cases, ranging from low-latency (S3) operational control to data-intensive automated planning and optimization (P1).

These principles provide the foundation for the detailed technical architecture of the Middleware and registry components presented in the following sections.

5.2.2 The aas-middleware: A Decoupled Engine for Data Integration

At the technical core of the Integration Layer lies the aas-middleware [1], a software designed to bridge heterogeneous IT and OT systems (Behrendt & Bail et al. 2025). Its modular architecture, as visualized in Figure 5.3, comprises five core building blocks: The Datamodel for unified data representation; Connectors, Formatters, and Mappers to systematically resolve data heterogeneity; and Workflows for composition of integration logic.

The Datamodel provides the foundation for data representation, managing schemas and their instances (I6) while offering functionalities for querying and validation. Integration is done in the Middleware based on a generic three-fold integration framework to systematically resolve data heterogeneity: Connectors handle technical integration (I1) by interfacing with diverse communication protocols (e.g., OPC UA, REST, RTDE); Formatters address syntactic

[1] The open-source project is available at: https://github.com/sdm4fzi/aas_middleware

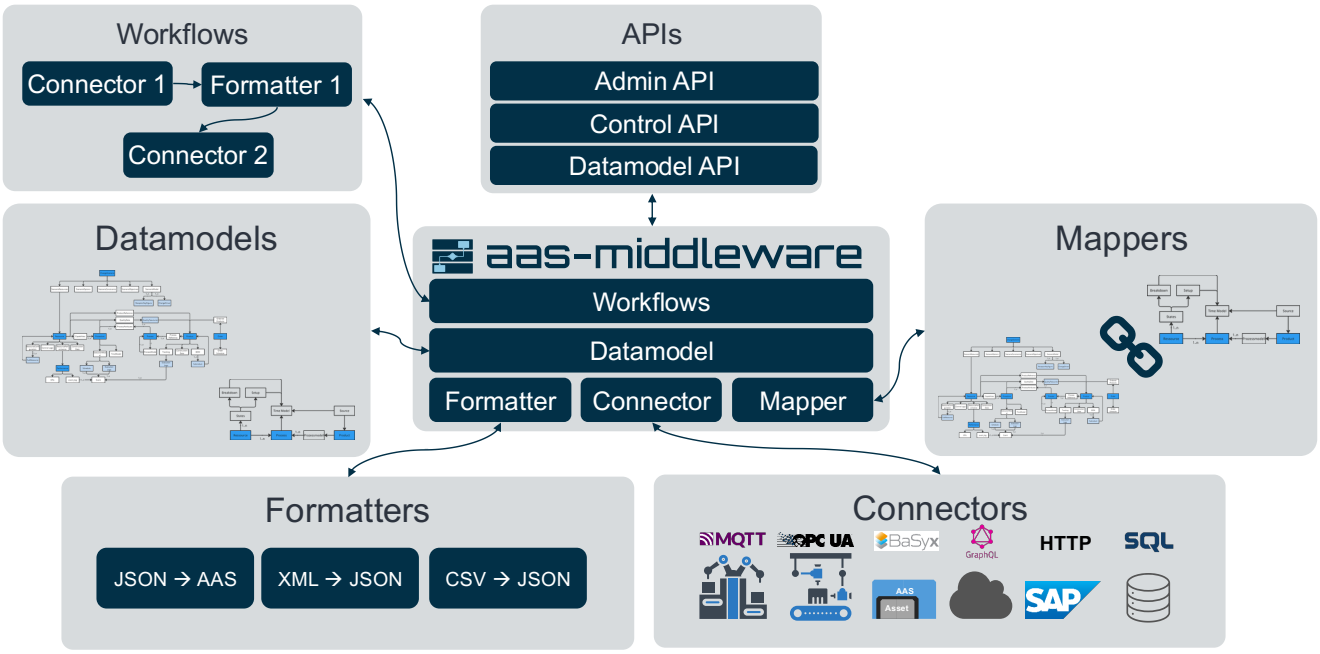

Figure 5.3: A schematic overview of the modular architecture of aas-middleware adopted from Behrendt & Bail et al. (2025).

integration (I2) by transforming data between different syntaxes, such as JSON and the AAS meta-model; and Mappers resolve semantic heterogeneity (I3) by translating data structures between different semantic contexts. Finally, Workflows provide the service composition possibilities, enabling the automation of complex integration flows by sequencing the operations of the other components in a defined and robust manner.

A critical design principle of the Middleware is its inherent flexibility and extensibility. All components are built upon basic interface definitions (see Appendix A4), allowing for straightforward extension with new Connectors, Formatters, or other custom logic. The system is fully configurable both programmatically via a Python Software Development Kit (SDK) for deep customization and dynamically during operation via a declarative REST API. This dual approach facilitates dynamic runtime adjustments to Datamodels and Workflows, enabling the system to adapt to changing requirements without the need for redeployment.

5.2.2.1 The Three-Fold Integration Framework

To systematically address data heterogeneity, the Middleware implements a three-fold integration framework. This approach deconstructs interoperability into three distinct layers, each handled by a dedicated component: *Connectors* for technical heterogeneity, *Formatters* for syntactic heterogeneity, and *Mappers* for semantic heterogeneity.

Note that the three-layer architecture, presented in Section 5.1, describes the architecture's overall decomposition into Infrastructure, Integration, and Application layers, whereas the

three-fold integration framework describes how data heterogeneity is resolved *within* the Integration Layer, specifically inside the Middleware.

The first stage of the framework, technical integration, is managed by *Connectors*. These components are responsible for resolving technical heterogeneity by handling the underlying communication protocols required to interact with various data sources and sinks. Connectors encapsulate the low-level details of establishing and maintaining connections to this diverse set of industrial and IT communication standards. In aas-middleware, Connectors are available for the most commonly used industrial communication technologies, covering OPC UA, MQTT, HTTP (REST), WebSocket, and direct real-time interfaces like RTDE for robotics.

By abstracting protocol-specific interactions, Connectors expose a standardized interface (see Appendix A4), effectively decoupling the logical information flow from the physical or technical means of data exchange. This ensures that services within the architecture can consume data from or send commands to any connected asset without needing to implement its specific communication protocol. Additionally, defining custom Connectors can be done simply by adhering to the standardized interface and registering it in the Middleware instance, making the Middleware extensible for specialized needs.

Once a technical connection is established, *Formatters* address the second stage: syntactic heterogeneity. Formatters are responsible for parsing data from its native format (e.g., XML, CSV, proprietary binary streams) into a unified internal representation, and conversely, serializing internal data into the format required by a target system.

A central capability of the Middleware is the automated, bidirectional translation between the AAS metamodel and lightweight, object-oriented data structures, such as JSON Schema or OpenAPI, using standardized Formatters. While the AAS provides a semantically rich and standardized representation of assets, its metamodel is complex and not natively aligned with common service engineering practices. Therefore, application- or client-facing services typically favor a flattened, developer-friendly schema that is easier to consume and validate. The Middleware resolves this mismatch by systematically formatting AAS metamodel components (see Figure 2.9) to primitive or structured object-oriented types.

Each AAS, Submodel, or Submodel Template, is interpreted as the *class-level* representation of a domain concept, where `idShort` becomes the class name and `submodelElements` become object attributes. Nested structures such as *SubmodelElementCollections* are recursively mapped to nested objects, while *SubmodelElementLists* are translated to arrays with homogeneous item types. Primitive AAS Properties are mapped directly to object attributes of type `string`, `integer`, `number`, or `boolean`. ReferenceElements, often used to express

Table 5.1: Mapping logic for translating AAS metamodel constructs into object-oriented schemas

AAS metamodel Element	Object-oriented Representation	Transformation Logic
`AAS`	`object`	AAS identifiers (`id`, `idShort`, semantic ID) become metadata fields; each Submodel becomes an attribute referencing a nested object.
`Submodel`	`object`	`idShort` defines the class name; each SubmodelElement becomes an attribute; semantic IDs preserved as annotations.
`SubmodelElementCollection`	`object`	Flattened into an object containing each contained element as an attribute; order irrelevant; recursive handling for nested collections.
`SubmodelElementList`	`array`	Homogeneous list mapped to an array type; item type inferred from element metamodel; ordering preserved.
`Property`	`string`, `integer`, or `number`, `boolean`	Property `valueType` determines primitive attribute type.
`ReferenceElement`	`string`	Value mapped to string to reference another entity.
`File`	`string`	Mapped to URI string attribute.
`Blob`	`string`	Base64 encoded string attribute.

cross-model links, are reduced to typed identifiers or URI-like strings in the object-oriented schema.

This translation enables services to interact with AAS-governed data as if it were a conventional object model, while the Middleware preserves the AAS as the ground truth. As a result, developers benefit from familiar object-oriented abstractions such as typed attributes and nested structures, while the system maintains full semantic compatibility with the AAS metamodel. Moreover, the availability of explicit object schemas enables automated schema conformance validation of AAS data, specifically Submodel Template conformance, which is currently not supported by existing reference implementations such as BaSyx (*Eclipse BaSyx – Industry 4.0 Operating System* 2025). This mapping logic, originally proposed in Behrendt & Stamer et al. (2023), is summarized in Table 5.1.

The final and most semantically rich stage of the integration framework - semantic integration (as explained in Section 2.4.1) - is handled by *Mappers*. Mappers perform the crucial task of transforming the structure and meaning of data from a source schema to a target schema. For example, a Mapper can translate the data structure from an asset's AAS self-description into the specific input format required by an external robot planning service. This resolves discrepancies in naming conventions, hierarchies, and data relationships.

While Mappers can be defined manually, *aas-middleware* does not provide a set of standardized Mappers since the vastness of existing industrial data models would not be feasible to consider. The Middleware only provides the framework for defining these Mappers and making them easily usable in integrations.

To resolve the issue of defining semantic mappings, a core research contribution of this thesis, detailed in Section 5.3, is the development of methods to generate these mappings using LLMs, thereby significantly reducing the manual integration effort and enhancing the adaptability of the overall system.

5.2.2.2 The Unified Data Model and Persistence

In the Middleware, the *Datamodel* component is intended to provide a unified representation of all system information. This component is designed as a searchable graph of classes and objects, which abstracts the underlying physical data sources and enables standardized, high-performance data access and manipulation. By offering functionalities like querying and schema validation, the Datamodel establishes a consistent and defined foundation for all subsequent data integration and orchestration tasks, effectively resolving the challenge of fragmented information landscapes. While this in-memory model ensures near-real-time operational responsiveness, robust and standardized long-term storage is achieved through a dedicated persistence strategy.

The architecture strategically designates the AAS as the semantic ground truth for data persistence, leveraging its status as a cornerstone of I4.0 to ensure semantic richness, interoperability, and long-term viability. In the reference implementation, this is realized by linking the in-memory Datamodel to a BaSyx AAS server (*Eclipse BaSyx – Industry 4.0 Operating System* 2025) via a dedicated persistence Connector.

This design choice, however, does not impose rigid constraints; the modularity of the architecture allows any system capable of fulfilling the Connector interface to act as the persistence backend, offering significant implementation flexibility. The persistence functionality is achieved by automatically linking each model instance in the Datamodel to a persistent

backend as the ground truth via a Connector. The Middleware ensures that these model instances are synchronized with the persistence by these Connectors.

While the AAS provides an invaluable, standardized persistence backend, experiments revealed that direct, high-frequency interactions with BaSyx AAS servers (*Eclipse BaSyx – Industry 4.0 Operating System* 2025) can cause severe performance degradation and temporary unresponsiveness, making their direct use challenging for automated PPC scenarios. To resolve this, the *aas-middleware* incorporates an intelligent caching layer, which is a key aspect of its materialized integration capability. This layer maintains a high-performance, in-memory Datamodel that serves as a rapid-access cache, while synchronization with the persistent AAS server is handled through a load balancer in the background to prevent system overload.

Conceptually, the Datamodel acts as the authoritative runtime representation of system state, whereas the AAS serves as a semantically rich, persistent backend. All read and write operations required for automated PPC are executed against the in-memory Datamodel to ensure low latency and system responsiveness. Changes to the system state are propagated asynchronously to the AAS backend and decoupled from control execution. This design prevents temporary unavailability or latency of the AAS server from affecting control logic execution, while maintaining eventual semantic consistency with the persistent AAS representation.

Crucially, the Datamodel enforces data integrity by automatically validating all data instances against their predefined schemas (e.g., JSON Schema or AAS Submodel Templates). This proactive schema checking is vital for preventing data-type mismatches and forms a non-negotiable prerequisite for the reliable execution of automated procedures and the future enforcement of service contracts. Moreover, enforcing schema definitions for data provides also a robust foundation for data integration, since clear definitions are available. By leveraging OpenAPI as a schema enforcement standard, incorporating metadata such as example data or detailed explanations is possible. This is a major enabler for the efficiency of the automated data integration, which will be explored later in Section 5.3.

The combination of a unified in-memory model, a semantically rich persistence strategy, and built-in validation mechanisms establishes the Datamodel as a reliable, high-performance data hub, underpinning the entire architecture's capacity for agile and automated production control.

5.2.2.3 Workflows for Realizing Information Flows in aas-middleware

Within the Middleware architecture, *Workflows* are integration-level constructs that encapsulate and execute reusable sequences of Connectors, Formatters, and Mappers to realize

concrete data transformations and system interactions. They are exposed via REST interfaces and can be invoked as standalone services, thereby serving as containers for integration functionality.

These Middleware Workflows must be clearly distinguished from the higher-level process models executed by the OE (later discussed in Section 5.4.2). OE process models describe PPC control logic using formal Petri net models and define *when* and *in which order* services are called within a PPC process. A service call defined by a petri net transition may therefore be bound to a Workflow as its concrete service implementation, encapsulating a complex integration routine. In this case, the OE performs a service call to the Workflow via its external interface, while remaining agnostic to the Workflow's internal realization in terms of Connectors, Formatters, and Mappers.

The composition of Workflows allows for the creation of sophisticated automation routines that bridge disparate data sources and sinks, effectively translating high-level objectives into executable actions across heterogeneous IT and OT systems. For instance, a Workflow can implement a complex planning task by first querying the current system state via a Connector, using Formatters and Mappers to transform that state into the required input format for an external planning service, invoking that service via another Connector, and finally orchestrating the resulting plan by sending commands to various OT components in the correct, synchronized order.

To support diverse operational requirements, the Middleware provides a flexible foundation for automation of Workflow execution, supporting various execution policies such as manual triggers, event-based initiation, or periodic execution, and can incorporate error-handling logic for robust operation.

Crucially, these Workflows are not static, internal constructs; they are exposed and manageable via a declarative REST API. This allows external systems to dynamically trigger their execution, observe their status, or interrupt their process flow, effectively exposing complex, multi-step business logic as a single, simple-to-call service endpoint. This powerful, API-driven orchestration capability extends the Middleware from a mere data integration tool into a dynamic process automation platform, enabling the automation of complex, multi-system PPC scenarios.

5.2.2.4 Dynamic API and Real-Time Capabilities

A core tenet of the proposed architecture is its capacity for runtime adaptability, which is fundamentally enabled by the dynamic API generation and near.real-time synchronization capabilities of the Middleware. These capabilities elevate the architecture from a static

blueprint to a responsive framework capable of managing the complexities of reconfigurable production systems.

A key feature that promotes system agility is the Middleware's ability to automatically generate comprehensive, fully functional APIs without requiring manual coding or system restarts. Based on the schemas defined within Datamodel components, the Middleware exposes both REST and GraphQL APIs, including a complete set of CRUD (Create, Read, Update, Delete) endpoints for every managed Datamodel. Datamodels can be defined either programmatically or via the Middleware's Admin API, enabling the configuration of Middleware capabilities at runtime.

To structure access to the Middleware's functionality and to clearly separate concerns, the dynamically generated interfaces are exposed through three conceptually distinct REST APIs, each targeting a specific interaction role:

- **Datamodel API**: Exposes CRUD endpoints for querying and manipulating Datamodel instances and their attributes. This API provides controlled read and write access to the integrated information space and is primarily used by application services and the OE to consume and provide data.
- **Control API**: Provides endpoints to call integration logic, including Connectors, Formatters, Mappers, and Workflows. It enables the orchestration and execution of integration and transformation processes without exposing configuration or structural details.
- **Admin API**: Enables the creation, modification, and removal of Middleware components, including Datamodels, Connectors, Formatters, Mappers, and Workflows. This API is used during integration engineering and runtime reconfiguration, for example by operators or automated tools such as the Schema Integration Service.

This separation ensures a clear distinction between data access, operational control, and system configuration, supporting safe runtime reconfiguration while preserving stable interfaces for PPC services.

In addition to its REST-based interfaces, the aas-middleware provides an exploration-oriented GraphQL API that enables clients to flexibly query specific aspects of the underlying data model. Unlike predefined REST endpoints, this interface allows consumers to formulate ad-hoc, fine-grained queries tailored to their current information needs, making it particularly suitable for exploratory analysis and tooling support. To the best of the authors' knowledge, this implementation currently represents the only GraphQL-based access mechanism for AAS data.

The dynamic API generation capability of the Middleware transforms the underlying Datamodels - whether native AAS or other formats like OpenAPI - into immediately usable and orchestratable web services. The primary advantage of this approach is that it decouples the application services from the underlying data structure's implementation, allowing the system's interfaces to evolve in lockstep with its configuration. Consequently, when a new module is added to a production line or a planning parameter is introduced, the system's interaction points can be updated dynamically, drastically reducing the effort and downtime associated with reconfiguration.

To ensure safe, deterministic, and auditable operation in industrial environments, the Middleware intentionally restricts the dynamic injection of executable logic. While Datamodels, Workflows, and integration configurations can be created and modified at runtime, arbitrary code injection is explicitly not supported. Instead, Connectors, Formatters, and Mappers are instantiated exclusively from a predefined set of standardized component implementations and configured through parameters exposed via the Admin API.

Custom integration logic therefore follows a two-step process: new component types are implemented as code during development time, adhering to the Middleware's interface definitions, and are subsequently deployed as trusted extensions. Once deployed, these components can be instantiated, parameterized, and orchestrated dynamically at runtime. This design balances flexibility with operational safety, ensuring that runtime reconfiguration does not compromise system integrity, security, or latency.

To ensure a consistent and accurate digital representation of the physical production environment, the architecture implements robust, multi-level synchronization mechanisms. Besides Connector synchronization with persistence, the Middleware allows additional Connectors to be linked to the Datamodel for synchronization and event triggers. These Connectors can be configured to automatically synchronize with this persistent state, ensuring that changes are propagated seamlessly between disparate IT and OT systems. The synchronization configuration is based on the synchronization scope (whole Datamodel, model instance within it, or individual attributes), the synchronization role (read only, write only, read write), and the synchronization direction (to persistence, from persistence, bidirectional).

This capability facilitates event-driven communication through the use of asynchronous Connectors, enabling the system to react to changes on the shop floor. For instance, a Connector to an OPC UA sensor value can be synchronized with the Datamodel attribute that stores this sensor value. The Middleware then ensures the automatic synchronization of these values in near-real-time. This ability to orchestrate state changes and trigger actions

based on real-world events is critical for closing the loop between planning and execution in dynamic manufacturing contexts.

The Middleware itself is not designed as a deterministic real-time system, since its performance is contingent on the underlying hardware and OS. However, it is engineered to function as a high-performance integration bridge that enables low-latency interaction between the IT and OT domains. It achieves this by integrating natively with established, low-latency OT protocols, such as OPC UA and RTDE. This strategic integration allows the architecture to leverage the strengths of existing OT technologies for time-critical control, while providing a powerful, flexible, and service-oriented layer for higher-level planning, orchestration, and data integration. In Section 7.4, we will explore how classical OT systems can be combined synergistically with the Middleware to provide real-time capabilities and reliability while also having high-level service-oriented configuration possibilities.

5.2.2.5 Flexible Integration Patterns: Virtualized vs. Materialized

The reference architecture is designed to flexibly support both primary data integration patterns: virtualized and materialized (see Section 2.4.4). This strategic duality allows the architecture to address the wide spectrum of temporal and state-management requirements found in complex PPC systems.

In the virtualized integration pattern, the Middleware functions as a stateless, on-the-fly transformation engine. Here, orchestrated Workflows composed of Connectors, Formatters, and Mappers fetch data from a source system and transform its technical protocol, syntactic format, and semantic schema. The result is delivered directly to a target system without persisting the transient data within the Middleware's core persistence layer. This approach is ideal for low-latency, event-driven data integration scenarios, such as translating a high-level service call into a near-real-time robot command, or for integrating new sensor values from the shop floor in a legacy system. The virtualized pattern excels where speed is paramount and the integration logic can be encapsulated as a direct, stateless data pipeline between two or more endpoints.

In contrast, the materialized integration pattern positions the architecture as a stateful information hub that maintains a consistent, unified, and persistent view of the production environment. This is achieved, as mentioned in Section 5.2.2.2, by declaring persistent Connectors that synchronize Datamodels with a persistent storage, such as an AAS server, that acts as the ground truth for all relevant production data. This mechanism realizes a data warehouse approach and guarantees data consistency across the distributed network. Effectively, this creates a robust, real-time digital representation of the entire production station's state, including its configuration, operational status, and active processes. The materialized approach

is especially useful if new data access possibilities for planning services need to be created because no existing legacy system provides this information.

The practical implementation of the architecture demonstrates a sophisticated synthesis of the materialized and virtual data integration patterns, designed to balance the benefits of standardization with real-world performance and scalability demands. The approach allows the system to support both high-frequency OT interactions and complex IT service integrations. The choice of pattern is therefore a dynamic, use-case-dependent decision.

For example, a comprehensive planning procedure can first materialize the current shop-floor state into the Datamodel. Subsequently, planning services can interact with the unified Datamodel through fast, virtualized queries. Finally, the resulting production plan is materialized back into the Datamodel, making it the new ground truth for execution. This ability to fluidly combine and orchestrate stateless data flows with stateful, persistent representations gives the framework the profound adaptability needed to manage the entire closed-loop PPC cycle, from real-time machine control to long-term, data-driven strategic planning.

Both materialized and virtualized integration approaches in the Middleware realize the GAV integration paradigm conceptually (see Section 2.4.2). In both cases, the Datamodel serves as the global schema $\mathcal{G}$ that unifies heterogeneous source schemas $\mathcal{S}$ by means of mappings $\mathcal{M}$ defined from the source schema to the global schema. For virtualized pipelines, $\mathcal{G}$ is not necessarily persisted; however, all transformations are still defined relative to the global schema, thereby preserving the GAV paradigm at the conceptual level.

The choice of the GAV paradigm over the alternative LAV approach is a deliberate one, rooted in the specific requirements of integrating operational PPC services and systems. In a GAV system, the global schema's concepts are explicitly defined as functions - the Mappers of the Middleware (Section 5.2.2.1) - over the source schemas. This procedural nature is highly advantageous in this context for three primary reasons. First, it aligns perfectly with a service-oriented architecture, where each Workflow is the concrete, robust implementation of a specific, pre-defined PPC task. Second, it provides a pragmatic and direct method for handling the complexities of legacy systems; the integration logic for a poorly documented source can be explicitly engineered within a Mapper, guaranteeing predictable performance and behavior. Third, the GAV approach allows to define Mappers as procedural functions, which are well suited for automated generation based on input and output schemas, as we will explore later in Section 5.3.

Conversely, an LAV approach, where source schemas are defined as views over the global schema, relies on complex query rewriting algorithms to satisfy a query against the global view. While powerful for ad-hoc data exploration, this introduces a level of abstraction and

potential performance unpredictability that is often unacceptable for the reliable, low-latency operational services required in a PPC environment. Additionally, the automated generation of query rewriting logic using LLMs is considerably more challenging than the generation of procedural mapping functions, as query rewriting requires global reasoning about semantic equivalence, completeness, and performance across multiple source schemas, whereas procedural mappings constitute localized, executable transformations with well-defined inputs and outputs. Therefore, GAV provides the necessary control, predictability, and task-oriented structure to build a robust integration Middleware for this domain.

5.2.3 The Service Registry

While the Middleware provides the foundational capability for integrating heterogeneous data flows, the overall architecture's scalability and maintainability depend on a robust system for managing the application services themselves. The Service Registry operationalizes the core principles of an SOA, moving beyond simple point-to-point integrations to create a dynamic and adaptable ecosystem of PPC services. Central to this system is the Service Catalog, which acts as the authoritative directory for all available services within the architecture's Application Layer. By providing a mechanism for runtime discovery and binding, it addresses a critical limitation identified in many existing digital manufacturing approaches, which often fail to implement the essential SOA characteristics of discoverability and composability required to scale effectively. Moreover, the Service Registry provides API Gateway functionalities, allowing the registry to unify decentrally deployed Middleware instances with different functionalities in a single API, avoiding the need for detailed configuration of interactions between Application and Integration Layer.

5.2.3.1 Central Register for Production Planning and Control Services

The Service Registry acts as a dynamic directory and API gateway for all PPC services. For each registered service, it stores the essential metadata required for automated discovery and consumption:

- Service Identifier
- Network endpoint (URI or address to reach the service)
- Formal interface definition (input/output schemas expressed in an OpenAPI specification)
- Functional documentation (capabilities, runtime, ...)

This catalog functions as a comprehensive and authoritative directory, containing detailed metadata for every available PPC service and Middleware component. This metadata extends beyond a simple list, capturing the necessary information to enable automated interaction.

The registry ensures that a service consumer has all the information required to interact with a provider without needing any prior, hard-coded knowledge of its implementation. By making the location of services available at runtime, the registry provides the crucial SOA characteristic of location transparency, which allows for flexible deployment and simple load balancing. The registry is therefore not merely a passive database but an active enabler of the loosely coupled, modular, and composable system structure that is fundamental to the architecture's design.

Beyond discovery, the registry also behaves as an API gateway, presenting a unified facade to clients regardless of where individual services actually reside. It continuously monitors registered services: updating health status in real time by checking availability and automatically rerouting requests to redundant instances in case of service failures-to maximize availability. Finally, its built-in authentication layer ensures that only authorized consumers may invoke services, a critical safeguard when exposing sensitive production data in a manufacturing environment.

5.2.3.2 Dynamic Discovery, Binding, and Exchange

The existence of the Service Registry facilitates a dynamic two-step process that realizes true loose coupling between service consumers and providers, a cornerstone of a mature SOA.

The first step is Discovery, where a consumer, such as the OE or another service, queries the registry at runtime to find a service that meets a specific need. Users can browse the Service Registry for suitable services based on service's metadata descriptions.

Once a suitable service is identified, the second step is Binding. The consumer uses the identifier of the service retrieved from the registry to establish a direct communication link and invoke the service. The registry provides based on the identifier direct dynamic routing to the service's location without requiring the users to consider this detail.

This discoverable and dynamically bound nature is the key mechanism that enables a "plug-and-play" environment for PPC functionalities. Service providers can be added, updated with new logic, or even physically relocated to different servers without impacting the consumers, as long as their registered interface contract remains stable. We will explore in Section 5.3, how the Middleware's Mappers can be utilized in case of interface contract changes to ensure contract backwards compatibility.

For RMS, this capability is paramount. It allows for the seamless exchange of planning algorithms - for instance, swapping a standard scheduling service for a more advanced, AI-based one - thereby enabling the PPC system to adapt its logic with the same agility as the physical production system it controls. This operationalizes the principle of composability and task-agnosticism, allowing complex planning pipelines to be assembled and reconfigured from independent, interchangeable services. Services that provide the same task can be interchanged without the service consumers consideration, allowing for decoupled and agnostic information flows.

5.2.4 Formalization of Service-oriented Production Planning and Control

To elevate the proposed architecture from a technical blueprint to a rigorous scientific framework, this section provides its formal underpinnings. It establishes a precise vocabulary for PPC services, models the system's information flows using established data integration theory, and formalizes the composition of these flows as a precursor to the automatic data integration(Section 5.3) and task-agnostic orchestration (Section 5.4). The formal basis ensures the correctness, verifiability, and task-agnostic modularity of the entire closed-loop system.

5.2.4.1 From Production Planning and Control Tasks to Formal Services

A foundational step towards automated, task-agnostic orchestration is the formal distinction between a conceptual PPC Task and its executable implementation as a PPC Service. A PPC Task represents a logical unit of work within the planning process, such as capacity planning or MRP. In contrast, a PPC Service is the concrete, self-contained, and discoverable software artifact that implements a given task. This formalization allows services to be treated as modular, interchangeable, and composable building blocks.

Definition 2 (PPC Service)**.** *A **PPC Service**, denoted Svc, is a tuple $(\mathcal{X}_{in}, \mathcal{Y}_{out}, f)$ where:*

- *$\mathcal{X}_{in}$ is the formal **input schema** defining the structure and semantics of the required input data.*
- *$\mathcal{Y}_{out}$ is the formal **output schema** defining the structure and semantics of the produced output data.*
- *f is the **service function**, an implementation that maps an instance of the input schema to an instance of the output schema, $f : I(\mathcal{X}_{in}) \to I(\mathcal{Y}_{out})$, where $I(\mathcal{S})$ denotes the set of all valid data instances conforming to a schema $\mathcal{S}$.*

For example, an MRP service $Svc_{MRP} = (\mathcal{X}_{MRP}, \mathcal{Y}_{MRP}, f_{MRP})$ would have an input schema $\mathcal{X}_{MRP}$ defining the structure for a production schedule and bill of materials, and an output schema $\mathcal{Y}_{MRP}$ defining the structure for the resulting material cardinalities. The function f_{MRP} is the specific algorithm that performs the calculation. For MRP, this service would calculate which and how many components would be required to execute the schedule.

This strict separation of interface (the schemas $\mathcal{X}_{in}, \mathcal{Y}_{out}$) from implementation ($f$) is paramount. It enables different services that perform the same conceptual task to be interchanged, provided they adhere to the same interface contract. However, for services to be composed into complex orchestrations, data must be reliably shuttled between them and the central system state, overcoming significant data heterogeneity. This is where the principles of data integration become essential.

5.2.4.2 Realizing Production Planning and Control Services via Data Integration Workflows

Having abstractly defined a PPC task as a formal PPC Service (X_{in}, Y_{out}, f), we now formalize how the service's core logic-the function f-is realized. This function is not a monolithic block of code but is implemented as an executable integration Workflow within the Middleware. This approach allows us to firmly ground our practical implementation in established data integration theory.

The Middleware's architecture is a direct instantiation of the formal Data Integration System $\mathcal{I} = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ (Definition 1). In our context:

- The Global Schema $\mathcal{G}$ represents the Datamodel of the Middleware.
- The Source Schema $\mathcal{S}$ is the union of all schemas from the heterogeneous legacy systems being integrated (e.g., ERP databases, MES APIs, flat-file repositories).
- The Mapping $\mathcal{M}$ specifies the relationship between $\mathcal{G}$ and $\mathcal{S}$. Crucially, in the Middleware, $\mathcal{M}$ is not merely a set of declarative specifications but is realized as a collection of executable Workflows $\mathcal{W}$, each designed to overcome specific integration challenges.

A Workflow $\mathcal{W}$, therefore, is the concrete implementation of a GAV integration approach (see Section 2.4.2), transforming data from one or more sources $S_i \in S$ into the global schema G (or vice versa). This transformation is achieved by composing integration primitives, each designed to resolve a specific layer of heterogeneity as previously identified:

Definition 3 (Integration Primitives)**.** *An integration Workflow is composed of three types of primitive operators:*

- *A **Connector** (c) resolves technical heterogeneity. It is a function that takes connection parameters and executes a protocol-specific request (e.g., an SQL query, a REST API call) to retrieve raw data from a source system.*

- *A **Formatter** (h) resolves syntactic heterogeneity. It is a function that parses a data stream into a structured data object that conforms to a specific schema definition language.*

- *A **Mapper** (m) resolves semantic heterogeneity. It is a function that transforms a data object from a source-specific representation $I(S_i)$ to its corresponding representation in the global schema $I(G)$ or vice versa[1].*

A complete integration Workflow, $\mathcal{W}$, is a composition of these primitives. For a simple data retrieval task from a single source s_i with schema $\mathcal{S}_i$, the Workflow can be represented as a linear function composition:

$$W_{S_i \to G} = c_{s_i} \circ h_{\mathcal{S}_\rangle} \circ m_{\mathcal{S}_i \to \mathcal{G}} \tag{5.1}$$

Data is retrieved from the source with the Connector c_{s_i}, formatted by $h_{\mathcal{S}_i}$ in the schema definition language utilized by Mapper $m_{\mathcal{S}_i \to \mathcal{G}}$, which subsequently transforms the data from schema $\mathcal{S}_\rangle$ to the global schema $\mathcal{G}$.

However, most PPC tasks span multiple systems, necessitating more complex interactions between different Connectors, Formatters, and Mappers. Therefore, a Workflow $\mathcal{W}$ is more generally defined as a directed acyclic graph (DAG) of these primitive operations, allowing for complex data flows such as forks, merges, and conditional logic.

For instance, a Workflow to "Generate a Production Schedule" might first use a Connector-Formatter-Mapper chain to retrieve open orders from an ERP, and a second chain to retrieve machine availability from an MES. A final Mapper would then consume the outputs of both chains to produce a ProductionSchedule object conforming to the global schema $\mathcal{G}$ which is saved in the Middleware's Datamodel with a persistence Connector.

This powerful concept of a Workflow allows us to make the final and most critical connection in our formalization: the encapsulation of integration complexity behind a clean service interface.

[1] Contrary to traditional integration systems that only focus on integrations into the global schema, Mappers from global schema to source schema are needed when sending data to a system with a source schema.

Definition 4 (Workflow as Service Function)**.** *Let $\mathcal{W}_{task}$ be an integration Workflow designed to execute a specific PPC task. The function f of a PPC Service (X_{in}, Y_{out}, f) is defined as the executable Workflow itself:*

$$f := \mathcal{W}_{task} \tag{5.2}$$

The input schema $X_{in} \subseteq \mathcal{G}$ represents the data required from the global model to trigger and parameterize the Workflow. The output schema $Y_{out} \subseteq \mathcal{G}$ represents the data produced by the Workflow, structured according to the global schema $\mathcal{G}$.

By this definition, the Workflow serves as an integration wrapper around legacy systems to realize a distinct PPC task. It exposes a standardized contract through its service definition while completely hiding the complex, multi-step, and heterogeneous integration logic within its implementation. This provides a robust separation of concerns, enabling a higher-level process orchestration (as will be discussed in Section 5.4) to compose and execute PPC services in a task-agnostic manner without any knowledge of the underlying system landscape and necessary integrations to realize the PPC task.

5.3 Method for Automatic Data Integration

This section presents a novel LLM-based method that automates the semantic integration process covering schema matching, mapping and data transformation (see Section 2.4.3). By allowing domain experts to specify integration logic through natural language, supported by formal schema definitions, example data, and business rules, the approach enables agile, prompt-driven semantic integration.

From an evaluation perspective, this design directly targets smart and automated PPC capabilities (P1–P6), in particular task agnosticism (P2), automated execution of PPC tasks (P1), and the integration of human decision-making in the control loop (P6).

The following subsections first detail the goals of the LLM-based integration as a pragmatic alternative to established paradigms (Section 5.3.1). A structured prompt framework for capturing integration requirements (Section 5.3.2) is proposed that serves as the foundation for LLM-based schema integration. Next, iterative refinement mechanisms and safeguards that ensure reliability, determinism, and security (Section 5.3.3) are discussed. Lastly, the operational procedure from prompt assembly to deployed integration service (Section 5.3.4) is explained.

5.3.1 Rethinking Data Integration in Manufacturing

This section addresses the practical gap between the persistent semantic heterogeneity in manufacturing data landscapes (Section 2.4) and the need for integration solutions that

remain feasible under frequent schema change and limited modeling resources in brownfield environments (Section 2.4.5). Rather than introducing another static integration method or domain model for PPC, the goal is to provide a low-overhead mechanism that can automatically produce, validate, and maintain executable data transformations under evolving schema conditions.

Accordingly, the proposed method follows five design goals that ensure practical applicability of the method:

G1 *Domain-expert usability:* Transformation intent can be specified using concise natural-language rules, without requiring expertise in formal modeling languages or logics.

G2 *Executable outputs:* Semantic integration results in a reusable, executable mapping function rather than declarative mapping tables.

G3 *Verifiability:* Correctness is ensured through systematic schema-conformance checks and automatically generated tests.

G4 *Safe execution:* Generated transformation code is executed in a sandboxed environment and subjected to security checks.

G5 *Change tolerance:* Integration logic supports iterative refinement, versioning, and rollback.

These goals align with the modular, service-oriented architecture introduced in Section 5.2 and the evolution criteria (especially scalability (S2), modularity (S4), loose coupling (S7), orchestrability (S8), semantic integration (I3), automated integration (I5), and standardized schemas (I6)) presented in Section 3.1: the semantic integration logic is encapsulated as a modular Mapper that can be composed into existing integration pipelines and evolved independently. The following sections define the structured prompt specification that operationalizes these goals, and the service procedure that turns a mapping request into a validated, deployable integration endpoint.

5.3.2 A Conceptual Framework for LLM-based Schema Mapping

The core of the proposed method is a conceptual framework that leverages a structured prompt as a formal, semi-structured and machine-interpretable specification for data transformation. This approach composes systematically a detailed request that constrains and guides the LLM toward generating a precise and executable output. This structured prompt consists of several key components, each designed to address a different facet of the mapping task.

While traditional schema matching approaches often produce a declarative mapping table that requires further interpretation to be executable, this framework is designed to directly

generate imperative, executable mapping functions (see Section 2.4.3 for a more detailed distinction). Pre-studies performed in A_Bulach (2025) and approaches from the literature (Buss & Safari et al. 2025) show that LLMs are able to generate mapping tables. However, generating imperative mappings offers two key advantages. First, semantic integration can be done as a single-step process that eliminates error-prone translation from a mapping table to imperative code. Second, the approach exploits the strength of LLMs in generating source code. Since LLMs excel with popular languages over niche ones (Cassano & Gouwar et al. 2024), the framework is designed to generate Python code.

This structured approach also directly addresses common challenges in using LLMs for data integration, as stated before in Section 3.3. By generating a reusable mapping function, it mitigates the high computational costs of repeated LLM calls. The strict guidance provided by schemas, rules, and examples helps to reduce inconsistent outputs or hallucinations, with the final output being a verifiable piece of code.

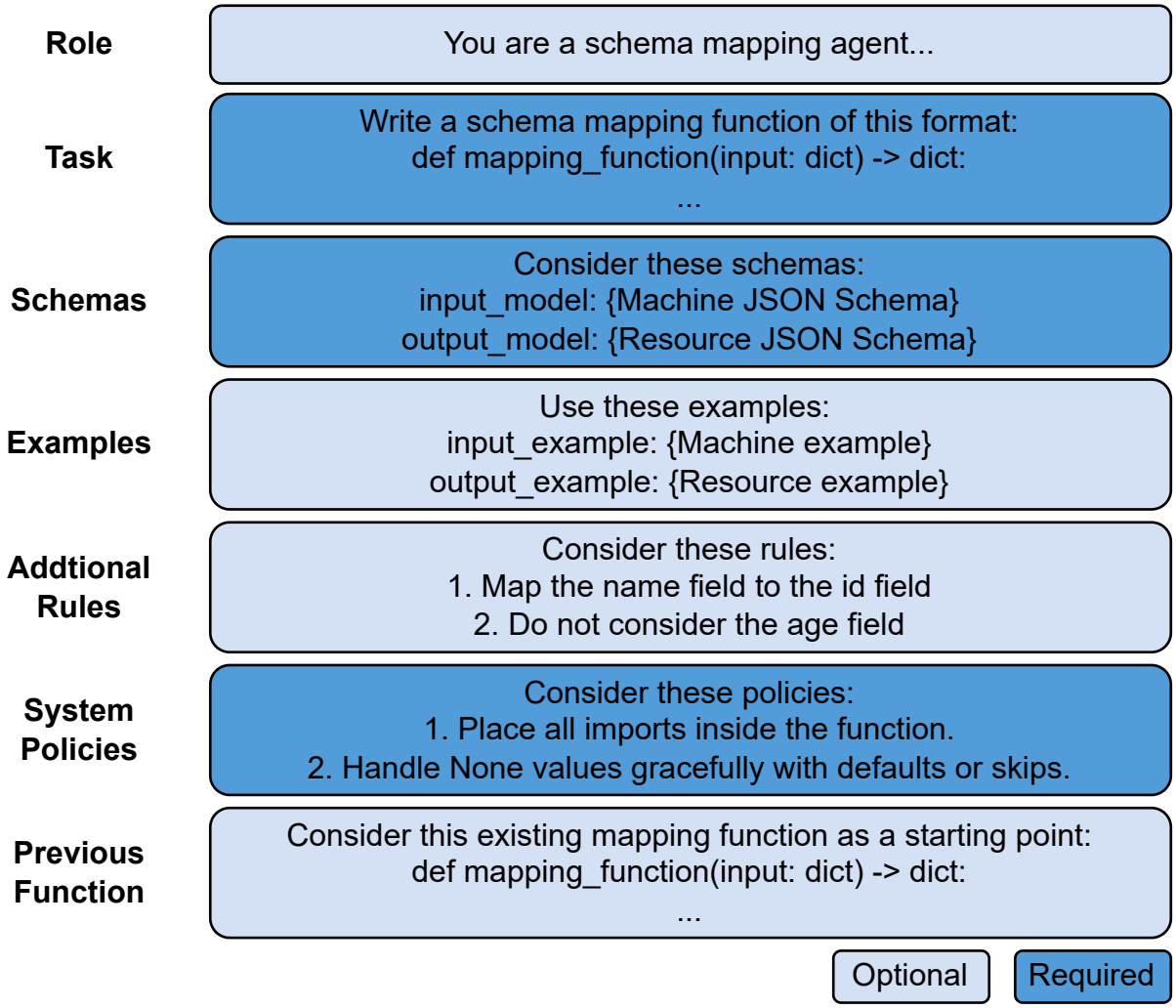

Figure 5.4: Workflow for Automated Generation of Semantic Mappers using LLMs.

The required and optional components of the structured prompt, as shown in Figure 5.4, are as follows:

Role. The prompt begins by assigning a system role to the LLM. This initial directive sets the context, priming the model to act as an expert in data integration and focus on the specific domain of the problem.

Task. This component defines the precise goal and the required output format. By specifying a function signature, the prompt instructs the LLM to generate a callable, executable artifact rather than a descriptive text answer. This ensures the output can be directly integrated into a larger data processing procedure.

Schemas. The Schemas are a required component, providing the structural foundation for the transformation. The prompt includes both the source (input) and target (output) schema of the mapping, which can be supplied in formats like JSON Schema or OpenAPI specifications. Providing explicit schemas gives the model the precise field names, data types, and nesting structures required for the mapping. Any documentation within these schemas semantically enriches the definitions, improving the LLM's understanding.

Examples. To leverage the in-context learning capabilities of LLMs, the prompt can include example instances for the input and output data of the mapping. These examples help the model infer implicit mapping patterns, such as renaming fields, restructuring objects, or formatting values, that may not be explicitly stated elsewhere. It should be noted that input and output examples must not be mapped instances, but can represent different instances, because direct input–output pairs are not available during integration.

Additional Rules. For complex transformations that go beyond direct field-to-field mapping, additional rules can be specified in natural language. This component allows a user to define custom business logic, such as "Map the name field to the id field" or "Do not consider the age field." The LLM's instruction-following ability is crucial for interpreting these rules and translating them into the correct code logic.

System Policies. This required section outlines non-functional requirements and coding standards. System Policies ensure the generated code is robust and adheres to best practices, with instructions like "Place all imports inside the function" or "Handle None values gracefully with defaults or skips."

Previous Function. To support iterative development, the framework allows for an optional Previous Function to be included. This enables the LLM to use an existing mapping function as a starting point for refinement or modification, making the process adaptable to changing requirements.

By combining these structured elements, the framework provides a comprehensive and constrained context for the LLM, guiding it to generate reliable and executable schema mapping functions. An example of an instantiated structured prompt and the resulting mapping function can be found in Appendix A3.1 and Appendix A3.2.

5.3.3 Providing Feedback to Users and Enabling Iterative Optimizations

While the initial automated generation process produces a mapping function, the inherent complexity and potential ambiguity of schema matching and mapping necessitate a framework for refinement and continuous improvement. The schema integration process is therefore extended with capabilities that create a continuous feedback loop, empowering users to iteratively review, validate, and optimize the generated mappings. This is achieved through a combination of comprehensive support tools, an intuitive correction process, and robust version management.

To build user trust and ensure the correctness of the generated logic, the SIS provides a suite of utilities for validation and transparency. First, it can generate human-readable documentation and summaries of the mapping logic, making the LLM's output transparent and easier for stakeholders to review. Second, for more rigorous verification, the system can automatically create unit tests for the mapping function, allowing for programmatic validation of its mapping functionality. These unit tests rely on example data to validate correct mapping and allow to clearly locate when possible errors exist in the mapping function. If no example data is available, the LLM tries to infer example data. Finally, users are able to perform test runs with live sample data, providing an immediate preview of the transformation output and allowing for direct verification of its functional behavior in a practical context.

Examples of automatically generated documentation and the corresponding unit tests are provided in the Appendix (see Appendix A3.3 and Appendix A3.4).

Should a user identify a discrepancy or an area for improvement through these tools, the framework facilitates an iterative refinement cycle. Users can provide corrective feedback in natural language, such as specifying a new business rule or correcting a flawed calculation. The system then initiates a context-aware adjustment process. Rather than generating a new mapping from scratch, it incorporates the existing prior code into the previously used prompt alongside the user's feedback. This treats the previous output as a foundation for revision, enabling the LLM to make targeted modifications.

To support this iterative methodology and provide a crucial safety net, the system integrates robust version management. Each generated mapping function is versioned and persisted in the service's repository, creating a complete and auditable revision history. This allows users

to review the entire history of adjustments and, if a recent change produces an unexpected or undesirable outcome, easily roll back to a previously validated version. This structured, iterative approach transforms schema mapping from a static, one-time query into a dynamic and collaborative development process. By programmatically assembling user feedback, prior code, and validation requirements into a comprehensive and context-rich prompt, the framework enables a user to progressively guide the LLM's output, turning a simple instruction into a comprehensive, version-controlled, and dynamically adjustable specification for automated data integration.

5.3.4 Schema Integration Service: From Prompt to Executable Integration

To operationalize the presented schema integration prompt, an SIS is developed that automates the presented methodology, from an initial mapping request to the deployment of an executable transformation function. Figure 5.5 visualizes this procedure.

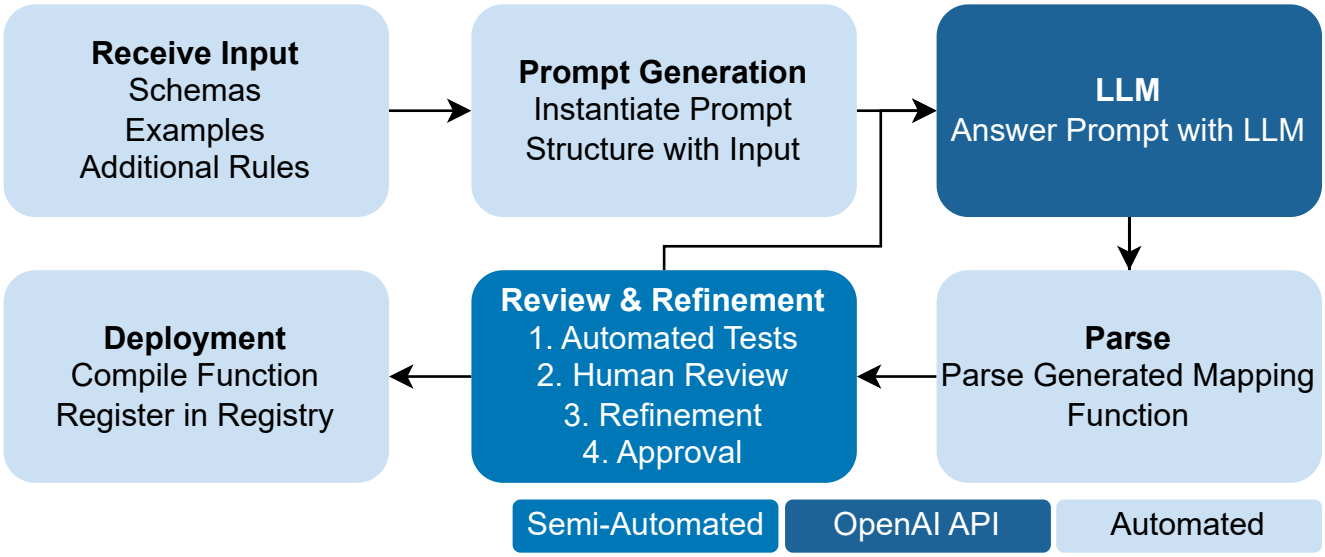

Figure 5.5: Procedure in the SIS for Automated Generation of Semantic Mappings using LLMs.

The process is initiated when a service consumer sends a mapping request containing source and target schemas and optionally example data and additional rules. Upon receiving the request, the SIS uses the provided input data to dynamically instantiate the structured prompt, as explained in Section 5.3.2.

Once generated, the complete prompt is sent to the designated LLM API for processing. The LLM, in turn, answers the prompt by generating the corresponding mapping function code. The LLM returns the mapping function as a string, which gets parsed subsequently.

The generated code is not immediately trusted; instead, it is passed to an isolated sandbox environment for parsing and rigorous validation. Within this sandbox, the code is first checked for syntactical correctness and then analyzed for potential security vulnerabilities, such as harmful system calls or code injection attacks[1]. Only after the code passes these checks, the validated mapping function is persisted in the SIS. This persisted artifact includes the function code itself, a unique identifier, a version tag, and all associated metadata, such as the source and target schemas and the examples used for its generation.

Next, in the Review & Refinement step the generated mapping function is reviewed, refined if necessary and approved by a human expert. At first, the SIS generates tests and documentation for the schema mapping function with another LLM call to help human experts in the review process. If some problems in the mappings are found, the expert can either adjust the mapping manually or iterate with the LLMs by adding additional rules that avoid the problems. Here, the previous function code is also considered in the LLM prompt to use in-context learning.

Upon approval, the system starts with deployment which produces a validated and immediately useable mapping function. At first, the SIS compiles the validated mapping function in the sandbox environment. To guarantee data integrity, the execution of the mapping function is bracketed by schema conformance checks: the input data is validated against the source schema before the transformation, and the output data is validated against the target schema afterward.

Afterwards, the mapping function is made available for on-the-fly execution. An endpoint is dynamically generated, allowing the mapping function to be invoked externally to transform source data into the target schema. Moreover, the SIS registers this mapping endpoint in the Service Registry to make this service discoverable and location transparent.

5.4 Design of a Task-agnostic Production Planning and Control System

This section builds on the reference architecture (Section 5.2) and the mechanisms for service integration (Section 5.3) by extending them toward the dynamic operationalization of the PPC system. The core idea is to represent PPC control loops as executable, formally defined Petri nets, enabling the system to execute complex planning and control tasks without hard-coded logic.

[1] In hacking, injections refer to a category of attacks where malicious code is inserted into a vulnerable system, often through user input, to manipulate its behavior or gain unauthorized access (Ray & Ligatti 2012).

First, Section 5.4.1 introduces the Schema-aware Petri Net (SAPN) model, which enriches classical Petri nets with data schemas, validation rules, and typed service interfaces. Section 5.4.2 then describes the OE that interprets these Petri nets and makes them executable at runtime. Embedding human decision-making for interaction with the SIS and controlling the OE is presented in Section 5.4.3. Finally, Section 5.4.4 outlines based on a PPC example how the application of SAPN achieves task-agnostic and resilient operation, including model modularization, runtime reconfiguration, and fallback mechanisms to ensure robustness in dynamic manufacturing environments.

5.4.1 Formalization of Schema-aware Petri Nets

The successful orchestration of a task-agnostic PPC system necessitates a formal language capable of describing complex, data-centric service process models with precision and verifiability. This section details SAPN (Behrendt & Enke et al. 2026), a specialized class of CPN, developed to serve as the foundational orchestration formalism for the proposed PPC framework.

Building upon the robust theoretical foundations of CPNs (see Section 2.5.6.1), the SAPN model extends CPNs to explicitly incorporate the data schemas that are central to SOA and data integration. This extension transforms the Petri net from a purely logical model into a data-aware framework, enabling the automated validation of service interfaces and ensuring the integrity of information flows throughout the PPC process lifecycle. This formalization directly addresses the challenge of orchestrating loosely coupled services, which requires a model that not only defines the control flow but also governs the data flow with formal guarantees.

Formally, an SAPN is defined as a tuple:

$$\text{SAPN} = (\Sigma, P, T, A, N, C, G, E, I, \sigma)$$

where:

- Σ is a finite set of color sets that represent the schemas of nodes and transitions.
- P is a finite set of places (representing states in the PPC process).
- T is a finite set of transitions, representing PPC tasks and associated service calls.
- A is a finite set of arcs (representing possible paths in the PPC process).
- $N : A \to (P \times T) \cup (T \times P)$ is the node function.

- $C : P \to \Sigma$ assigns a color set to each place.
- $G : T \to \text{Expr}_{\text{bool}}$ assigns guards to transitions.
- $E : A \to \text{Expr}$ assigns arc expressions.
- $I : P \to \text{Multiset}(\Sigma)$ defines the initial marking.
- $\sigma : T \to (\Sigma_{\text{in}} \times \Sigma_{\text{out}})$ is the *service signature function* which assigns input/output color sets to each transition.

The key innovations of the SAPN lie in the semantic enrichment of its core components (places, tokens, and transitions) which are no longer abstract entities but are formally linked to the system's data and service landscape, as visualized in Figure 5.6.

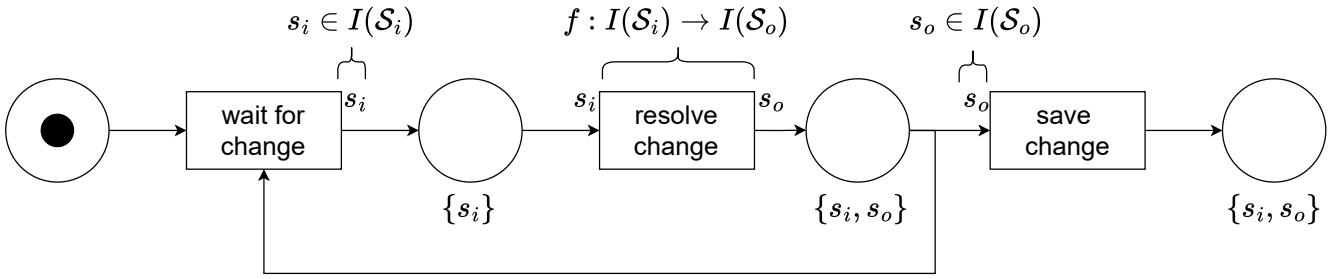

Figure 5.6: Conceptual example of an SAPN with 3 services (wait for change, resolve change, save change), schema annotations S_i (input schema) and S_o (output schema), set of possible instances $I(S_i)$ and $I(S_o)$, and specific instances s_i and s_o.

In the SAPN model, a Place (P) transcends its traditional role as a simple condition marker. Through the color-set assignment function C, each place becomes a typed data container, formally associated with a specific data schema. This establishes the critical link between the orchestration model and the underlying data integration framework. The set of all possible color sets, Σ, is precisely the universe of data types available to the net. It is formally defined as the set of all component schemas within the Global Schema $\mathcal{G}$ from our data integration system (see Section 2.4.2). Thus, the function $C : P \to \Sigma$ assigns a concrete data schema from $\mathcal{G}$ to every place in the net.

For instance, a place labeled *ProductionOrders* would be assigned the schema for a production order object from $\mathcal{G}$. The presence of a token in this place signifies more than just the existence of an order; it guarantees the availability of a data object instance that is structurally valid and conforms to the *ProductionOrders* schema. This direct linkage between the process model's state and the system's formal data definitions is fundamental to ensuring data consistency and interoperability as information flows between different PPC services.

Complementing the typed nature of places, Tokens in an SAPN are not abstract markers but are data-carrying payloads. Each token represents an instance of an information object, such as a specific customer order with a unique ID, a bill of materials for a particular product, or a machine status update. The initial marking I populates the net with the initial set of these data-tokens. A critical rule of the SAPN is that the data structure encapsulated within a token must validate against the schema of the place it currently occupies. This ensures that a service invoked by a downstream transition receives data in the precise format it expects, eliminating a common source of integration failures in heterogeneous environments.

The most significant departure from classical Petri nets is the definition of schema-aware Transitions T. In the SAPN, a transition represents a PPC service invocation, but it does so in an abstract, task-agnostic manner. The innovation lies in the service signature function σ, which annotates each transition with its formal data requirements: the set of input schemas (Σ_{in}) it needs to consume and the set of output schemas (Σ_{out}) it will produce. This signature function's domain corresponds directly to the service's formal interface $(\mathcal{X}_{in}, \mathcal{Y}_{out})$ as defined in Section 5.2.4.1.

Crucially, this specification does not bind the transition to a specific PPC service function f; instead, it defines an abstract service signature σ as a requirement. This cleanly encapsulates the underlying integration complexity, where the function f is itself implemented as an executable integration **Workflow** $\mathcal{W}_{task}$ (as conceptually described in Section 5.2.2.3 and formally defined Section 5.2.4.2). For example, a transition might be specified to require a *ProductionSchedule* schema and a *ResourceLayout* schema as input and produce a *ValidatedSchedule* schema as output. This requirement acts as a dynamic query to the Service Registry, which is responsible for finding a concrete, available service that satisfies this interface contract at runtime. This late-binding mechanism is the cornerstone of the system's flexibility, allowing services to be substituted, updated, or selected based on dynamic criteria without altering the underlying process logic.

To leverage this dynamic orchestration capability safely, the framework must guarantee that any composition of services is logically sound. The SAPN model provides this guarantee through the concept of a well-typed net. A SAPN is considered well-typed if it satisfies formal compatibility conditions for all its transitions, ensuring that the data schemas align correctly across service boundaries. The two primary conditions are:

Definition 5 (well-typed SAPN)**.** Input Type Compatibility*: For any transition t, the union of schemas of its input places must be a superset of the schema required by the service. Formally:*

$$\Sigma_{in}(t) \subseteq \bigcup_{p \in Pre(t)} C(p)$$

Output Type Compatibility*: For any transition t, the schema of the data it produces must be a subset of the union of schemas of its output places. Formally:*

$$\Sigma_{out}(t) \subseteq \bigcup_{p \in Post(t)} C(p)$$

This condition ensures that the output of a service is compatible with the data containers designated to receive it.

Taken together, these well-typedness conditions provide a formal guarantee that the orchestration logic defined by the SAPN is compatible with the service interfaces defined in Section 5.2.4.1. This enables the static verification of entire PPC process models before execution, preventing data-related integration errors at the process level and ensuring the robustness of the dynamically composed system. Additionally, possible problems in schema confirmation of the SAPN are located, serving as a starting point for the previously presented automated schema integration procedure (see Section 5.3).

In summary, the SAPN provides the formal, machine-interpretable, and verifiable blueprint for orchestrating PPC tasks. It systematically bridges the conceptual gap between the system's static data model and its dynamic process logic. This formalization is the essential foundation to build reconfigurable, agnostic PPC process models without hard-wiring a specific PPC concepts logic, as current PPC systems do. In the following, it is detailed how the SAPN formalism is used in the system's OE to bring reconfigurable PPC process models to life.

5.4.2 The Petri Net-based Orchestration Engine

While the SAPN formalism provides the blueprint for dynamic process logic, the OE is the software component that brings this blueprint to life. As a core component of the reference architecture introduced in Section 5.1, its purpose is to interpret a given SAPN model, manage its state over time, and orchestrate the complex interactions between services. In essence, the engine transforms static process definitions into dynamic, executable process models that respond to real-time events on the shop floor.

Engine Architecture. The OE serves as a self-contained microservice in the PPC reference architecture, that interacts with the Service Registry, to ensure loose coupling and extensibility enabled by dynamic service discovery. Additionally, the engine can be coupled with the SIS, proposed in Section 5.3.4, to bridge possible schema conformance problems of service contracts.

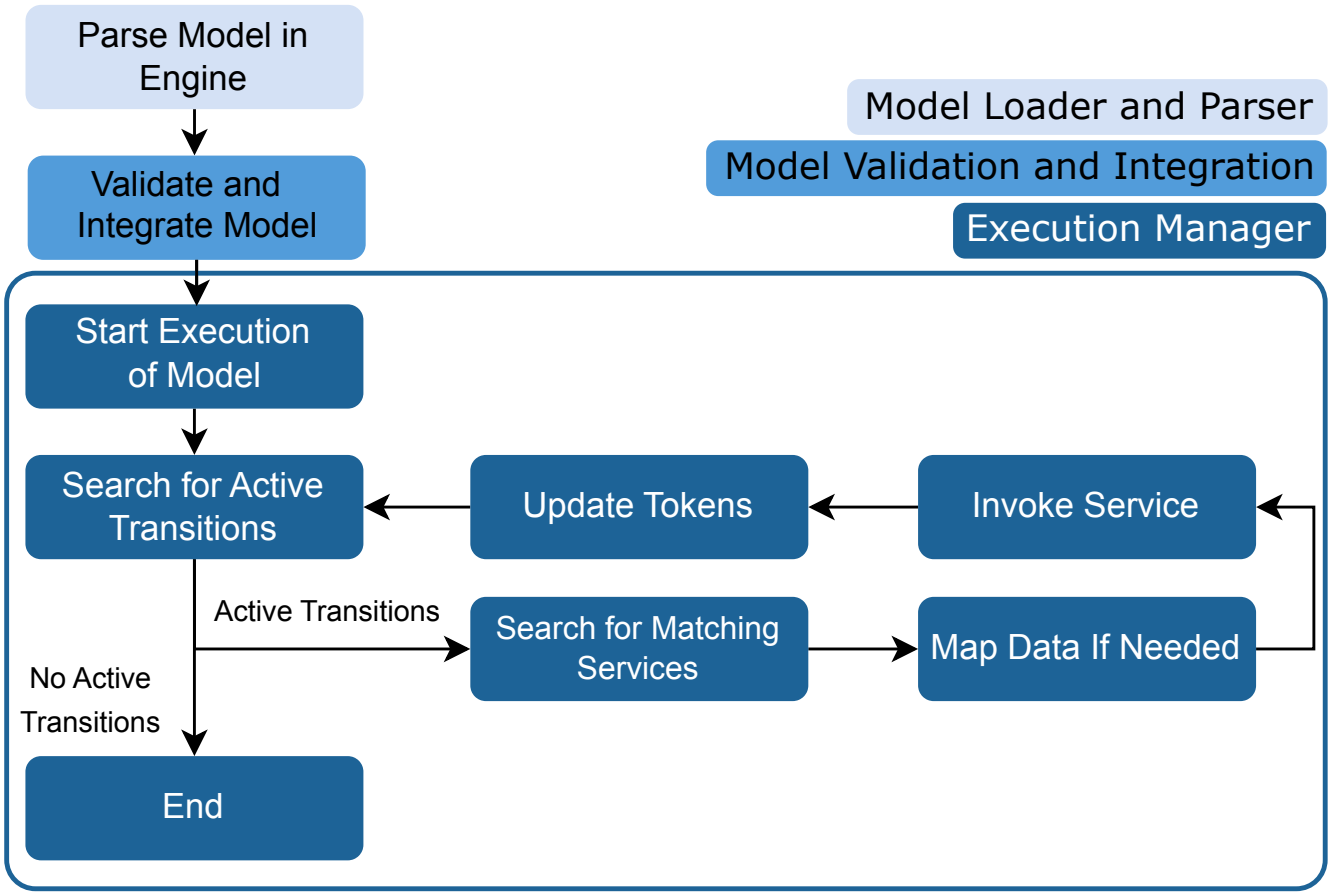

Figure 5.7: Model execution logic of the OE

Figure 5.7 visualizes the architecture and execution flow of the OE. The engine is composed of three primary, logically distinct components that manage the SAPN process models.

First, the Model Loader and Parser is the engine's entry point for process logic. It is responsible for ingesting a serialized SAPN model description and parsing it into an in-memory object graph that the engine can execute. This capability is fundamental to the system's reconfigurability, as it allows new or updated PPC process models to be loaded and activated at runtime without requiring a system restart.

Second, a model validation and integration module ensures a robust and executable SAPN model. In this step, the Service Registry is queried to obtain services that match the SAPN model definition based on the defined PPC task name. For instance, if the SAPN defines for a Transition t the task name *Scheduling* all services present in the Service Registry that provide this capability are fetched. Next, well-typedness is validated based on the SAPNs transition types and the services interface description. In case of non-conformance, the Service Registry is searched for Mappers, created with the SIS (Section 5.3), that can resolve these non-conformances of service interfaces.

Lastly, the Execution Manager executes the active SAPN model and maintains its current marking. It tracks the distribution of data-carrying tokens and ensures that all associated payloads remain valid and accessible. The component can expose this state to provide a real-time view of an ongoing planning process. As the system's control unit, it continuously evaluates the net, identifies enabled transitions, initiates the required service calls, and updates the tokens once services complete. It thus embodies the operational control flow of the entire PPC system.

The engine progresses the PPC process through a continuous, event-driven cycle consisting of four steps.

The engine inspects the current marking to determine all enabled transition, those whose input places contain the required tokens. A transition fires only when all informational preconditions are met.

For each enabled transition, the engine retrieves its abstract service requirement based on the task name provided in the SAPN model. It queries the Service Registry for concrete services that match it. A selection heuristic then selects the most suiting service based on factors such as availability, cost or runtime. This late binding ensures flexibility and resilience, as service selection is determined at runtime.

Before service invocation happens, it is checked if the selected service requires mappings to be conformant with the transition's signature function σ input and output schema. If so, the registry is again queried for Mappers that handle these transformations, which are applied to the tokens data payload before and after service invocation. This keeps orchestration logic decoupled from service implementation details and allows for dynamic integration of services. If required Mappers are not available, a semiautomated integration mechanism utilizing the reference architecture and the SIS is triggered to resolve these semantic mismatches, as will be described in Section 5.4.3.

The engine calls the selected service, passing along the data extracted from the transition's input tokens. After service completion, the engine updates its internal state by consuming the input tokens, generating new tokens containing the returned output data, and inserting them into the appropriate output places. Afterwards, the engine starts over the execution cycle.

While the above execution cycle describes the nominal case, the OE is explicitly designed to handle situations where the selected service instance cannot be executed successfully. A fallback mechanism is triggered if a service becomes unavailable at runtime, violates execution constraints (e.g., timeout or error), or fails to produce output data conforming to the transition's output schema Σ_{out}.

In such cases, the OE does not modify the SAPN model or its current marking. Instead, it re-evaluates the set of registered services that satisfy the same abstract task and service signature function σ and dynamically binds an alternative service instance. If required, the corresponding Mappers are selected and applied transparently, preserving the well-typedness of the net.

Importantly, fallback behavior is not explicitly modeled within the SAPN itself. It emerges from the separation between abstract task definitions in the SAPN and concrete service binding at runtime in the OE. This design ensures that resilience to service failures is achieved without embedding error-handling logic into the process model, thereby maintaining task-agnosticism and modularity.

5.4.3 Human-in-the-Loop for Integration and Control

The OE positions the human PPC employee not as a passive observer, but as the conductor of the orchestration process, empowered to make critical adjustments and grant necessary approvals.

This human-in-the-loop paradigm is essential for leveraging the system's flexibility to resolve problems and adapt to unforeseen circumstances. The OE is therefore designed to facilitate this collaborative control, particularly when managing the dynamic integration of services and overseeing the execution of PPC process models. While the engine can autonomously handle many aspects of service discovery and invocation, it explicitly seeks human guidance in situations that require contextual knowledge or carry significant operational implications.

One primary area for this interaction is the integration of services. During the validation phase, if the engine identifies that no conformant service is available for a specific transition, it notifies the operator. Similarly, if it discovers a new, potentially valuable service that is not yet compatible with the SAPN's schema requirements, it flags this as an integration opportunity. In both scenarios, the system presents the situation to the user, who can then decide whether to initiate an integration process through the SIS.

This interactive approach extends to the integration process itself. When the SIS is invoked to generate a new integration, the resulting mapping function can be presented to a human expert for review and approval. This step is crucial for ensuring the semantic correctness of the integration, especially in complex cases where automated analysis might be insufficient. The operator can then validate, refine, or reject the proposed mapping, providing a critical layer of oversight that prevents data-related errors.

In case of missing integrations to legacy systems that require protocol changes or complex service interactions, the aas-middleware can be utilized to create Workflows (see Section 5.2.2.3)

that encapsulate these interactions with Connectors, Formatters and Mappers. After this preparation and integration (see Section 5.1.2), the Workflows are registered as callable services in the Middleware and bound to SAPN transitions in the OE.

Beyond service integration, the human operator retains comprehensive control over the execution of the process model. The engine offers a real-time visualization of the running SAPN, allowing planners to monitor progress and understand the current state of the PPC tasks. Through a dedicated control interface, an authorized user can intervene directly in the process model. This may involve pausing the entire process to investigate an anomaly, manually selecting a specific service from a list of compatible options based on real-time criteria like cost or expected execution time, or stopping and re-planning parts of the execution. This ability to steer the orchestration at runtime ensures that human expertise remains central to the system's operation, combining the speed of automation with the nuanced decision-making of an experienced planner.

5.4.4 Modeling Production Planning and Control as Executable Petri net Instances

Building on the SAPN formalism and the OE, this subsection illustrates how a concrete PPC control loop is modeled and executed. The example shows how the same SAPN model can flexibly bind to different services, apply fallback mechanisms, and integrate new services at runtime without changing the SAPN model itself. This connects the three layers of the proposed solution: (i) the formal SAPN model, (ii) the OE, and (iii) the resolution of a real PPC problem through dynamic service integration with SIS.

From PPC task to SAPN model. The example we want to discuss is displayed in Figure 5.8 and represents a small PPC control loop that waits for breakdowns in the production, triggers a scheduling task, and deploys the newly generated plan to production.

The SAPN logic can be described as follows:

1. **Wait for breakdown**: This transition listens for breakdown events from the MES. When the MES emits a machine-failure event, a token is inserted into the transition's output place with data $s_{ps} \in S_{ps}$, describing the current state and orders of the production system.

2. **Schedule**: This transition consumes the token with $s_{ps} \in I(S_{ps})$ and outputs a schedule $s_s \in I(S_s)$, placing this data and the production system data in the output place.

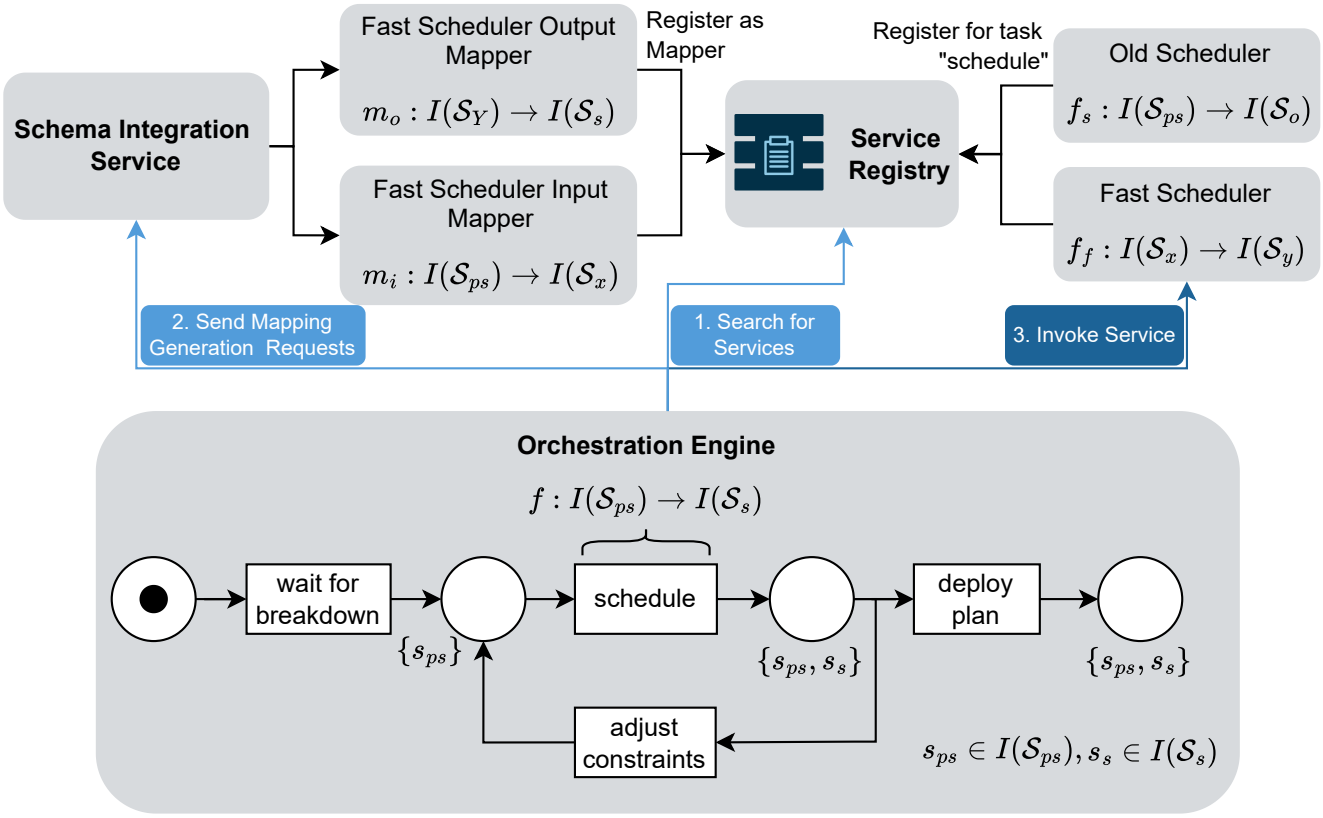

Figure 5.8: Visualization of the execution of an exemplary SAPN PPC control loop

3. **Adjust Constraints**: If no valid schedule could be generated, i.e. s_s is empty, an arc expression only activates the adjust constraints transition which relaxes scheduling constraints (e.g., available makespan) to find a solution.

4. **Deploy Plan**: If a valid schedule is found, the arc expression is fulfilled and this transition deploys the new plan to the production system.

As described in Section 5.2.4.1, a PPC service implements a PPC task such as the schedule transition, which can be described as a function. Here, the service signature function of the transition describes a function $f : I(S_{ps}) \rightarrow I(S_s)$ which transforms input data of a production system with schema S_{ps} into output data of a schedule with schema S_s.

In this model, nothing is said about which scheduling service is used. The SAPN only specifies that the PPC loop needs a service with capability *schedule* and the given input/output schemas. This is the core of task-agnosticism: the process logic is fixed as a formal model, while the concrete services can vary.

Dynamic integration, binding and fallback mechanism between multiple services. When starting an execution, the OE follows the execution cycle described above. It parses the SAPN

in the Model Loader and starts afterwards validation of the SAPN. The model is checked for well-typedness, which succeeds.

Before the execution starts, the OE scans the Service Registry for possible services for each PPC task specified by transitions (1. in Figure 5.8). If a suitable service is found, where the input and output schemas validate with the transition's service signature function, they are selected. If services exist, that implement the PPC task but have another schema, they are marked for integration.

The user of the OE is informed about these possible integrations. In our example, we want to consider a newly available *Fast Scheduler*, which should replace the *Old Scheduler* for rescheduling after breakdowns. Initially, an entry must be added to the Service Registry to make the new Fast Scheduler discoverable, providing metadata that describes the service's functionality and location.

If the user selects to integrate the new service before execution, the OE sends two mapping generation requests to the SIS (2. in Figure 5.8):

- A mapping m_i from the transition's input schema S_{ps} to the Fast Scheduler's input schema S_x
- A mapping m_o from Fast Scheduler's output schema S_y to the transition's output schema S_s

After reviewing and authorizing the generated mappings in the SIS, the mappings are registered in the Service Registry, ready for usage in the OE to make the Fast Scheduler available for use. This demonstrates the strength of the service-oriented approach with a reactive integration component and a task-agnostic orchestration: new services can be efficiently integrated into running cycles, allowing for modular extensions.

Now two schedulers are available for PPC:

- a Fast Scheduler: fast, rough-cut capacity model for reactive replanning.
- an Old Scheduler: slower, but more accurate used in weekly planning.

During executions, the OE has to select one of the two available services. A selection heuristic (e.g., prioritize high-fidelity if enough time is available) chooses one of the services for executing the scheduling based on metadata provided by the Service Registry, in our case the fast scheduler. The OE identifies that mappings are required and executes it with the Mapper m_i to transform the token's data to conform with the fast scheduler's service interface. Now, the scheduling service can be invoked (3. in Figure 5.8). If the scheduling is successful,

the output data is mapped again with m_o to validate with the transition's service signature function.

The advantage of dynamic binding and routing in the OE comes at play in cases where the Fast Scheduler is not available due to maintenance or failed in generating a valid schedule. The OE automatically falls back to an alternative implementation, i.e. the Old Scheduler with the same capability.

In both cases, the SAPN model is identical and requires no modification; only the bound service instance changes. The fallback mechanism is thus an emergent property of (i) abstract task definitions in the SAPN and (ii) runtime service discovery in the OE.

From formalism to PPC solution. This example demonstrates how the proposed approach connects:

- **Formalism**: The SAPN encodes the breakdown-handling loop as a well-typed, data-aware Petri net with abstract tasks.
- **Engine software**: The OE loads and validates the model, discovers suitable services, applies schema mappings, and executes the net.
- **PPC solution**: The system reacts to a machine breakdown, evaluates alternatives via one of several scheduling services (with fallback mechanisms), and deploys the revised plan to an MES.

Because the control loop is defined as a serializable SAPN, it can be stored, versioned, exchanged, and reloaded at runtime. The same pattern applies not only to disruption handling, but also to other PPC tasks such as order release, capacity planning, or what-if analysis. The process logic is treated as data, while the concrete services and mappings can evolve independently-realizing the vision of a task-agnostic, resilient PPC system.

5.5 Rationale for Key Design Decisions

The design of the proposed framework results from a series of deliberate decisions that directly address the research gaps presented in Section 3.5 and meet the evaluation criteria defined in Section 3.1. Each design element is purposefully aligned with one or more of the three major evaluation categories-Software Architecture and Service Orientation (S1–S8), Data Integration and Interoperability (I1–I7), and Smart and Automated PPC (P1–P6)-ensuring a comprehensive and traceable conceptual foundation. Beyond the conceptual rationale, each design choice also provides a concrete anchor for evaluation in the case studies, e.g., by

assessing scalability, efficiency, latency, and integration flexibility under realistic industrial conditions.

Integration of SOA and SDM Paradigms. This architecture synergistically combines the paradigms of **SOA** and **SDM** to achieve a high degree of modularity (S4), loose coupling (S7), and scalability (S2). The SDM paradigm enables the independent evolution of hardware, software, and application logic by decoupling functional layers through a Middleware, thereby enhancing technological flexibility and reducing integration effort. In parallel, SOA principles-implemented via a Service Registry-provide mechanisms for dynamic service discovery (S5), location transparency (S6), and runtime composition (S8).

Together, these design choices mitigate the monolithic rigidity of conventional PPC systems and establish a dynamic, evolvable foundation for RMS, addressing architectural criteria S1–S8. Additionally, lightweight service invocation and decentralized communication patterns help reduce system latency (S3), supporting responsiveness requirements for real-time control. To validate the architecture's capabilities, the later evaluations will analyze *latency*, *scalability*, and *runtime flexibility* in the case studies, especially regarding how many services can be orchestrated responsively under real manufacturing loads.

Decentralized Middleware with a Three-Fold Integration Framework. To address data heterogeneity in a structured and scalable manner, the framework employs a decentralized Middleware. This Middleware decomposes integration into three manageable steps: *Connectors* resolve technical integration across communication protocols (I1), *Formatters* handle notational integration across data formats (I2), and *Mappers* perform semantic integration (I3). By structuring integration along these layers, the design supports both scalability and maintainability (S2, S4). Moreover, standardized schema interfaces (e.g., AAS) ensure interoperability through established industrial data models (I6), while open and modular definitions promote transparency and adoption (I7). This modular decomposition explicitly supports brownfield integration scenarios by allowing legacy systems to connect via service wrappers-preserving existing IT investments and ensuring backward compatibility.

From an evaluation perspective, this structure allows case studies to investigate how flexible and efficient the Middleware integrates heterogeneous systems, how well AAS-based interfaces scale in real industrial contexts, and how the Middleware interoperates with other technologies. These criteria can be measured through integration effort and runtime data throughput in the experiments.

LLMs for Semantic Integration. The application of **LLMs** for semantic mapping represents a strategic design decision targeting automated semantic integration (I3–I5), whereby the design of the SIS was guided by five method-level design goals (G1–G5).

LLMs enable domain-expert usability (G1) by interpreting transformation intent directly from structured natural-language descriptions and formal schema definitions, supporting a *hybrid integration approach* that combines classical data models with LLM-based semantic inference. Beyond declarative correspondences, the framework deliberately leverages LLMs to generate *imperative* mapping logic (G2) (Section 2.4.3), enabling complex data transformations that would be difficult to express using static mapping tables alone.

To address concerns regarding correctness and trustworthiness, verifiability (G3) and safe execution (G4) are ensured through lightweight but systematic safeguards: structured prompts constrain the solution space, schema validation and automatically generated tests significantly reduce structurally invalid mappings, and sandboxed execution mitigates security risks. Change tolerance (G5) is supported by iterative refinement of mappings, versioning, and controlled redeployment of integration services.

For quantitative evaluation, this component is assessed in a dedicated benchmark (Section 7.2) by measuring mapping accuracy and robustness across heterogeneous PPC schemas, evaluating the scalability of imperative code generation for large, nested industrial schemas, and quantifying the reduction in manual integration effort.

Petri nets for Service Orchestration. For the coordination of services, Petri nets were chosen as the orchestration formalism due to their formal semantics, analyzability, and dynamic execution properties. In contrast to script-based orchestration, Petri nets offer a declarative and verifiable model suitable for concurrent and distributed process models. This directly supports orchestrability (S8) and task agnosticism (P2). The choice of orchestration (rather than choreography) ensures centralized control and observability (P4–P6), while the Middleware's Workflow abstraction allows for modular compositions that can also support choreographic interaction where decentralized execution is desired. Although alternative process modeling approaches such as BPMN are in principle applicable, they do not provide the same level of rigorous formal semantics necessary for dynamic, analyzable execution as Petri nets. Together with the Service Registry, this enables flexible coordination of heterogeneous PPC services and forms the basis for higher-level automation capabilities (P1–P5).

For later validation, a dedicated benchmark (Section 7.3) evaluates the *scalability*, *runtime overhead*, and *responsiveness* of the OE, while its applicability in complex autonomous PPC procedures involving multiple services, high-frequency event updates, and dynamic reconfiguration is considered in three case studies.

Modularity and Adaptability Across Contexts. A crucial architectural principle is *modularity*, which ensures that the framework's components - Service Registry, Middleware, SIS, and OE - can be adopted independently or in combination. This design guarantees adaptability to

varying industrial contexts: less complex systems may employ manual mapping or simpler orchestration logic, as considered in case study 1 (Section 7.4) and 2 (Section 7.5), while more advanced deployments can fully exploit LLM-based automation and dynamic orchestration, as investigated in case study 3 (Section 7.6). Hence, the architecture remains scalable and practical across a broad spectrum of use cases, from partially integrated brownfield systems to fully automated I4.0 environments (fulfilling S2, S4, P1–P2).

In synthesis, the framework directly operationalizes the principles derived from the evaluation framework. It systematically addresses research gaps in (1) architectural modularity and orchestration (S1–S8), (2) semantic and automated data integration (I1–I7), and (3) autonomous and task-agnostic PPC capabilities (P1–P6).

5.6 Summary and Contributions

This chapter presented an integrated approach for enabling flexible, service-oriented PPC in RMS. The core outcome is a coherent architectural and methodological foundation that spans data integration, semantic alignment, and task-agnostic process model execution.

First, the reference architecture (Section 5.2) introduces a software-defined PPC stack organized into Infrastructure, Integration, and Application layers. At its center, the *aas-middleware* resolves technical, notational, and conceptual heterogeneity through Connectors, Formatters, and Mappers, supported by an AAS-based persistence model and a high-performance Unified Datamodel. Dynamic API generation, synchronization mechanisms, and a Service Registry enable scalable, modular PPC ecosystems. The architecture formally treats information flows via the GAV paradigm, establishing a global semantic view that underpins automated mapping and orchestration.

Building on this foundation, the Section 5.3 introduced an LLM-driven method for automatic schema mapping and data transformation. A structured prompt specification enables the generation of executable mapping functions without labelled data or manual coding. An end-to-end procedure provides validation, sandboxed execution, versioning, and iterative refinement, allowing domain experts to reliably produce and evolve integration logic within modern service-oriented environments.

Finally, Section 5.4 uses the software-defined architecture and semantic integration capabilties to realize task-agnostic service orchestration for PPC. SAPN is introduced as the formal basis that integrates data schemas with control-flow logic, making PPC processes verifiable, serializable, and reconfigurable at runtime. The OE operationalizes these models via late binding to services in the Registry and supports advanced capabilities such as fallback

mechanisms and data validation. Most importantly, the engine has mechanism for adaptive alignment of non-conformant interfaces of services to avoid integrations.

Together, these contributions establish a cohesive framework that unifies service-orientation, semantic transformation, and dynamic orchestration in a single software-defined PPC approach. This establishes the technological foundation for future-proof, adaptive, and automated PPC systems aligned with the principles of I4.0.

6 Implementation

This chapter details the technical realization of the service-oriented PPC framework proposed in Chapter 5. It provides a transparent account of the software architecture, the technology stack employed, and the core logic of the system's primary components. The objective is to translate the abstract concepts of the framework into a tangible and functional software system, thereby establishing the concrete foundation for the empirical validation and case studies presented in Chapter 7.

The implementation is broken down into three main sections, each addressing a key pillar of the architecture: the Middleware and Service Registry, the SIS, and the OE. An overview of the most important open-source software used in these implementations is given in Table 6.1.

Table 6.1: Open-source software components used in the implementation of the service-oriented PPC framework.

Software	**Role in the System**	**Project / Reference**
FastAPI	Implementation of REST-based services and APIs	https://fastapi.tiangolo.com/
Pydantic	Schema definition, validation, and serialization of data models	https://docs.pydantic.dev/
Graphene-Pydantic	Automatic generation of GraphQL schemas and resolvers from Pydantic data models	https://github.com/graphql-python/graphene-pydantic
Docker	Containerization and isolated deployment of services	https://www.docker.com/
HashiCorp Consul	Service registry, service discovery, and health checking	https://www.consul.io/
Eclipse BaSyx	AAS server for persistent data storage	https://www.eclipse.org/basyx/
LangChain	Prompt templating, LLM interaction, and sandboxed code execution	https://www.langchain.com/
OpenRouter	Unified API access to multiple large language models	https://openrouter.ai/
Angular	Web-based user interfaces for SIS and OE	https://angular.io/
Python asyncio	Asynchronous execution and concurrency in the OE	https://docs.python.org/3/library/asyncio.html

6.1 Development of the Middleware and Service Registry

The Integration Layer, comprising the *aas-middleware* (Behrendt & Bail et al. 2025)[1] and the Service Registry, forms the communication and data integration backbone of the entire system. This section describes its implementation, focusing on how the conceptual architecture was translated into a robust, scalable, and extensible software artifact.

6.1.1 Technology Stack and Architecture

The core Middleware component is implemented in Python, chosen for its extensive ecosystem of libraries for web development, data science, and system integration. In the implementation of the aas-middleware, only open-source software is used, as shown in Table 6.1.

The *FastAPI* web framework was selected as the foundation for all services due to its high-performance and asynchronous capabilities, which are essential for handling concurrent requests in a real-time production environment. FastAPI's native integration with *Pydantic* for data validation ensures that all data exchanged between services strictly adheres to the defined schemas, a critical requirement for maintaining data integrity across the distributed system. Moreover, the compatibility of Pydantic and FastAPI with Json Schema and OpenAPI enable explicit schema definitions. The GraphQL API of the middleware is implemented using *Graphene-Pydantic*, which enables the automatic derivation of GraphQL schemas and resolvers directly from the Pydantic-based data model.

To ensure consistent, isolated, and portable deployments, all services, including each Middleware instance, are containerized using *Docker*. This approach simplifies dependency management and allows the entire system to be deployed seamlessly across different environments, from local development machines to the production-scale Learning Factory used for validation in Chapter 7.

For custom extensions of the Middleware, such as defining new Connectors or Formatters, standardized interfaces are defined in the Middleware, as specified in Appendix A4.

The Data Model is implemented as an in-memory, searchable object graph. This design provides a high-performance caching layer, enabling microsecond-level access times for services that need to read or write system state. The graph structure allows for efficient traversal of relationships between different data entities (e.g., finding all products associated with a specific order).

While the in-memory model provides speed, a persistence mechanism is required to act as the ground truth and ensure data consistency. The in-memory model is transactionally

[1] The open-source project is available at: https://github.com/sdm4fzi/aas_middleware

synchronized with a persistent backend. In the reference implementation, this synchronization is handled by a dedicated Connector that communicates with a *Eclipse BaSyx* AAS server (*Eclipse BaSyx – Industry 4.0 Operating System* 2025). By materializing the canonical system state in the standardized AAS format, our implementation aligns with established I4.0 standards and ensures interoperability with other compliant systems.

6.1.2 Service Registry Implementation: Consul

For the Service Registry, a production-grade, open-source tool, *HashiCorp Consul*[1], was chosen. Its selection was motivated by its proven robustness, scalability, and rich feature set, including service discovery and health checking. However, other existing Service Registry implementations could be used interchangeably.

The mechanism for service interaction with Consul is straightforward and automated:

- **Service Registration:** Upon startup, each service makes an API call to the Consul agent, registering itself with a unique ID, a service name (representing its capability, e.g., "scheduler"), its network address, and port.
- **Health Checking:** Each service exposes a standard health endpoint (`/health`). Consul is configured to periodically poll this endpoint. If a service fails to return a successful response within a specified timeout, Consul automatically marks it as unhealthy and removes it from the pool of available services for discovery.

This dynamic registration and health-checking mechanism provides the fault tolerance and load-balancing capabilities that are fundamental to the resilience of the overall system, as demonstrated in the validation chapter.

6.2 Integration of the LLM Modules

This section details the implementation of the LLM-based SIS. This service automates the creation of semantic integration with Mappers, which represent the most complex and knowledge-intensive aspect of data integration.

6.2.1 LLM Interaction and Prompt Assembly

The core function of the service is to generate a Python mapping function based on high-level specifications. The procedure is as follows: A user sends a request to the service containing the source and target data schemas (in JSON Schema format), optional data examples, and a set of rules expressed in natural language.

[1] available at: https://www.consul.io/

The system programmatically assembles a structured prompt, as conceptually defined in Section 5.3.2. To manage this process, the *LangChain* library is utilized. LangChain's templating features allow for the consistent and reliable construction of complex prompts that include the schemas, examples, rules, and instructional context for the LLM.

For interacting with the LLM APIs, the *OpenRouter* API aggregator is used. This provides a unified interface to access multiple state-of-the-art LLMs (e.g., OpenAI's GPT series, Google's Gemini, and Anthropic's Claude series).

Executing code generated by an external, non-deterministic source like an LLM introduces significant security risks. To mitigate these, a rigorous, multi-step security protocol was implemented to validate and execute the generated Python code within a secure sandbox environment.

1. **Static Analysis:** The raw code generated by the LLM is first subjected to static analysis. This step checks for basic Python syntax errors and scans for the use of disallowed imports (e.g., *os*, *subprocess*, *shutil*) that could interact with the host filesystem or execute arbitrary shell commands.

2. **Isolated Execution:** Code that passes the static analysis is then executed within a sandboxed environment. The *LangChain sandbox*, which leverages a secure execution environment, was employed for this purpose. This containerized approach strictly confines the code's execution scope, preventing it from accessing the network, filesystem, or any environment variables of the host system, thereby mitigating risks of code injection or other malicious activities.

3. **Functional Validation:** Within the sandbox, the generated function is invoked with the example input data provided in the initial request. The function's output is captured and validated against the target JSON Schema.

Only code that successfully passes all three stages of this protocol is deemed safe and functionally correct. The mapping is then made available by exposing the generated code via a dedicated service endpoint, which is registered in the Service Registry as a new mapping service.

6.2.2 Service and UI Implementation

To facilitate human-in-the-loop interaction, a simple web front-end was developed using the *Angular* framework. This user interface (UI) allows a domain expert to upload or define schemas, write natural language rules, provide examples, and trigger the generation process. The UI displays the generated Python code, the LLM's explanatory documentation, and the

validation results. It also enables the user to provide feedback and initiate iterative refinements, as described in the conceptual framework. Screenshots of the UI of the SIS can be found in Appendix A5.

6.3 Realization of the Orchestration Engine

This section describes the implementation of the task-agnostic OE, the component responsible for executing the SAPN models and orchestrating the system's behavior.

The engine is implemented as a FastAPI application which consists of three main components, as explained in Section 5.4.2.

The Model Loader and Parser exposes endpoints to import or export SAPN models as well as endpoints to adjust an existing SAPN model. This decoupling enables new or modified control loops to be loaded and activated at runtime via a simple REST API call to the engine's admin interface. The engine can therefore be reconfigured with new PPC logic without requiring a restart, which is a critical capability for RMS.

The engine's Model Validation and Integration and Execution Manager components are custom-built interpreters implemented in Python. A custom implementation was chosen over existing libraries to allow for tight integration with the specific semantics of SAPNs, particularly the schema-aware data token handling, dynamic service binding logic with the Service Registry and the integration with the SIS.

The engine's execution algorithm is asynchronous, concurrent, and event-based, designed to handle multiple process instances simultaneously. The execution algorithm uses asyncio-based concurrency, enabling multiple SAPN instances to progress independently while sharing the same runtime environment.

A UI was created for the OE to demonstrate how a user would design, control, and monitor SAPN-based PPC models. Screenshots of the UI of the OE can be found in Appendix A6.

7 Validation and Evaluation

This chapter presents the empirical validation of the holistic framework developed in this thesis. The primary objective is to systematically verify the framework's capability to enable flexible, automated, and reconfigurable PPC with respect to the identified evaluation points from Section 5.5 and the evaluation criteria from Section 3.1. The evaluation is specifically designed to provide concrete evidence addressing the three core research questions: the realization of a technologically consistent framework (RQ1), the automation of data integration (RQ2), and the realization of a task-agnostic, service-based PPC system (RQ3).

The validation approach follows a two-stage structure, as introduced in Section 4.3. First, targeted *component-level evaluations* assess the performance of individual framework elements in isolation, focusing on quantitative metrics such as mapping accuracy, orchestration latency, and scalability. These tests provide a controlled environment to benchmark specific technical capabilities against their intended functional requirements. Second, *system-level evaluations* demonstrate the applicability of the integrated framework in representative manufacturing scenarios through industrial and laboratory-based case studies. These case study evaluations reflect varying degrees of complexity, ranging from modular disassembly stations to learning factories and integrated industrial environments.

Each evaluation stage is designed to capture both quantitative and qualitative insights. While the former provides measurable evidence of technical performance, the latter captures practical aspects such as ease of integration, adaptability to changing conditions, and the degree of manual intervention required. The evaluation design, metrics, and setup are detailed in Section 7.1, followed by the presentation and discussion of component-level evaluations in Section 7.2 and Section 7.3 and system-level case studies in Section 7.4, Section 7.5, and Section 7.6.

7.1 Experimental Design and Key Metrics

To ensure a rigorous and transparent assessment, a two-stage evaluation strategy is employed, as visualized in Figure 7.1. This structured approach allows for the isolated validation of novel core components before evaluating their synergistic performance within the integrated system.

Stage 1: Component Validations (Section 7.2 and Section 7.3). The first evaluation stage focuses on the quantitative analysis of the automated data integration methodology and the OE. First, RQ2 is addressed by evaluating the LLM-based service for semantic schema mapping in an isolated, controlled environment. The objective is to measure the method's effectiveness and efficiency in data integration, independent of the complexities of

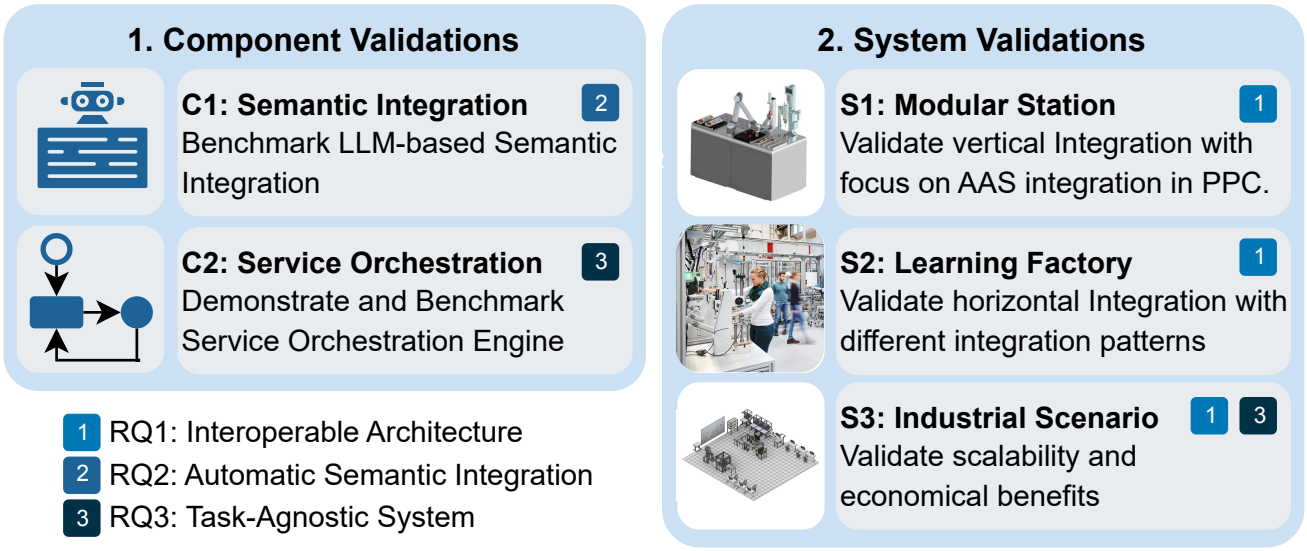

Figure 7.1: Overview of component and system validations and their mapping to the posed research questions.

the full system architecture. Additionally, RQ3 is quantitatively assessed by evaluating the performance and scalability of the OE on published benchmark problems. This allows to evaluate the orchestration capabilities in an isolated manner. Besides this, the reliability and fault tolerance of the OE is demonstrated by simulating service failures and dynamic service substitution tests in the orchestration.

Proving the viability of these core capabilities, i.e. data integration and service orchestration, is a prerequisite for its application in the subsequent, more complex case studies.

Stage 2: System Evaluations (Section 7.4, Section 7.5, and Section 7.6). The second stage undertakes a holistic assessment of the entire framework's functionality, performance, and practical utility. This stage directly addresses RQ1 through the implementation and analysis of three progressively complex, practice-oriented application scenarios with diverse data sources and integration of legacy systems. Moreover, RQ3 is addressed by orchestrating complex information flows in these PPC use cases, extending the isolated evaluations from stage 1.

The following case studies, as visualized in Figure 7.1, are considered:

- *Modular Station*: At a flexible, modular disassembly station, the framework's basic functionality, component interoperability, and the fundamental integrity of the service-oriented architecture are validated. This scenario serves to confirm the technological foundation of the end-to-end software architecture (RQ1).

- *Learning Factory*: Within the learning factory environment of the LFGP (Lanza & Moser et al. 2015), the framework's horizontal scalability is investigated. The focus here is on integrating heterogeneous legacy systems and orchestrating various existing PPC services into more complex Workflows, demonstrating the framework's capability to operate in brownfield environments (RQ1).

- *Industrial Scenario*: In collaboration with an industrial partner of the automotive supply sector, the system's performance is demonstrated in a highly complex and dynamic production scenario. This case study evaluates the combination of horizontal and vertical integration in a demanding industrial setting. It showcases the task-agnostic OE's ability to dynamically compose services for closed-loop PPC, thereby demonstrating the full potential of the framework to manage complexity and reconfigurability (RQ1, RQ3).

To conduct this two-stage evaluation systematically, a set of precise key metrics has been defined for each stage. These metrics provide the quantitative and qualitative basis for the analysis presented in the subsequent sections.

7.1.1 Metrics for Component Validations

7.1.1.1 Data Integration Metrics

The evaluation of the automated data integration method in Section 7.2 is based on a comparison against a manually created gold standard mapping. Table 7.1 summarizes the utilized metrics, which are grouped into three categories: *Accuracy*, *Effectiveness*, and *Efficiency*.

To measure *Accuracy*, we employ established metrics from schema matching and Natural Language Processing (NLP): *Precision*, *Recall*, and *F1-Score*. For each mapping task, the comparison with the gold standard yields a classification of every object and attribute as true positive (tp), false positive (fp), or false negative (fn). Based on the counts of TP, FP, and FN, the metrics are computed as defined in Equation 7.1:

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN} \qquad \text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7.1}$$

The *Effectiveness* of the integration method is evaluated by quantifying the percentage of generated mapping functions that can be parsed, compiled, and executed on the benchmark instances without runtime or validation errors.

Table 7.1: Evaluation Metrics for Automated Data Integration

Metric Category	**Metric**	**Purpose / Definition**
Accuracy	Precision	Proportion of correctly identified mappings among all mappings generated by the system.
	Recall	Proportion of correctly identified mappings among all mappings contained in the gold standard.
	F1-Score	Harmonic mean of Precision and Recall, providing a single balanced accuracy measure.
Effectiveness	Executable Mapping Rate	Percentage of generated schema mappings that can be parsed, compiled, and applied on the benchmark instances without errors.
Efficiency	Computation Time	Time required for automatic mapping generation.
	Token Cost	Number of input and output tokens required during mapping generation.
	Human Effort Reduction	Output token length approximates the reduction in manual coding effort compared to a baseline manual integration process.

Finally, *Efficiency* is assessed using both computational and practical cost measures: the required computation time, the token usage during mapping generation, and the expected reduction in manual implementation effort. We use the number of output tokens, i.e. the amount of code in the mapping function, as an indicator of the reduced implementation effort and potential efficiency gains with this automation. This is based on the observation that manual coding time scales approximately linearly with code length. Together, these metrics quantify the expected speed-up and cost reduction achieved by the automated integration method.

In summary, the Accuracy, Effectiveness, and Efficiency metrics jointly provide insight into the practical success of the proposed method in reducing manual effort for semantic integration tasks, thereby addressing the central goal of RQ2.

7.1.1.2 Service Orchestration Metrics

To quantitatively assess the core capabilities and scalability of the OE (RQ3) in Section 7.3, its performance and adaptability are demonstrated and evaluated using established process model benchmarks from the academic literature. Table 7.2 summarizes the metrics used to evaluate the OE. They are grouped into two categories: *Performance* and *Adaptability*.

Table 7.2: Evaluation Metrics for Service Orchestration

Metric Category	**Metric**	**Purpose / Definition**
Performance	Execution Time	End-to-end execution time for a process model instance, measured from initial trigger to final completion. Indicates system responsiveness.
	Throughput	Number of process model instances completed per unit time (e.g., processes/s). Measures processing capacity.
	Resource Utilization	CPU and main memory usage during benchmark execution. Quantifies computational efficiency.
	Network Overhead	Amount and size of messages exchanged between orchestrated services. Critical for evaluating communication efficiency in distributed systems.
	Performance Under Load	Ability to sustain acceptable latency and throughput under high system load.
Adaptability	Dynamic Reconfiguration Capability	Ability to replace an existing service with a new schema-compatible one at runtime without process model interruption. Evaluates plug-and-play flexibility essential for reconfigurable production systems.

To evaluate *Performance*, we measure the responsiveness, computational efficiency, and communication overhead of the OE. Following the benchmark methodology of Skouradaki & Ferme et al. (2016), process model patterns are used to stress-test the orchestration logic. These atomic patterns enable an isolated assessment of fundamental orchestration behaviors under controlled conditions, giving clear insights into the scalability of the approach.

Execution Time and Throughput provide direct measures for runtime behavior, while Resource Utilization and Network Overhead highlight potential bottlenecks in computation and inter-service communication - an aspect emphasized in research on distributed and decentralized architectures (Safi & Murad et al. 2011).

Additionally, the performance under load is assessed by simulating a high system load and observing how Latency and Throughput evolve. This is done by scaling the number of concurrently executed process model instances.

Manufacturing environments typically impose stringent performance demands due to high data volumes and event frequencies. The metrics defined above are therefore essential

to verify that the orchestration logic scales effectively and remains robust under industrial operating conditions.

Finally, *Adaptability* is evaluated through controlled runtime reconfiguration tests. This assesses the system's ability to dynamically swap services (e.g., integrating a faster or updated version of a service) without requiring process model redesign or manual intervention, thereby demonstrating the reconfigurable nature expected in modern production environments.

With these metrics, a rigorous quantitative basis for evaluating the OE is given that enables direct comparison with publicly available benchmark results (Skouradaki & Ferme et al. 2016) for process model management.

7.1.2 Metrics for System Evaluation

In the System Evaluation, the assessment of the integrated framework across the three case studies is based on system-level metrics that capture the overall operational behavior of the service-oriented PPC approach. These metrics are grouped into three categories: *Performance & Scalability*, *Adaptability*, and *Economic Viability*. Table 7.3 summarizes the metrics and their purposes.

Performance metrics quantify the operational efficiency and responsiveness of the system. Response Time captures how quickly the framework reacts to events, Data Throughput and System Load evaluate its continuous processing capability, and Scalability measures whether performance degrades gracefully as system load increases.

Adaptability evaluates the Robustness and Reconfigurability of the system. These two factors are essential for PPC of RMS. Robustness captures the system's ability to compensate for failures or disruptions, while Reconfigurability measures how easily new services or process logic can be integrated without architectural changes.

Finally, *Economic Viability* assesses the practical business impact of the proposed framework. As proposed by (Wiendahl & ElMaraghy et al. 2007), the economic performance of RMS is assessed using a DCF analysis based on (Newnan & Lavelle et al. 2019). The DCF method evaluates the net present value (NPV) of the system over time by discounting all cash inflows and outflows with an assumed discount rate of 10 %, thereby enabling a consistent comparison of alternative investment and reconfiguration strategies. The effect of operational flexibility and adaptability enabled by the proposed service-oriented PPC framework on the economic performance of the production system is assessed. This metric highlights the productivity gains and lifecycle cost reductions enabled by the approach.

Table 7.3: System-Level Evaluation Metrics Across Case Studies

Metric Category	**Metric**	**Purpose / Definition**
Performance	Response Time	Time for triggering service calls in the PPC control loops.
	Data Throughput	Volume of data (e.g., messages or events per second) that the framework can process reliably. Indicates processing capacity under realistic workload.
	System Load	Number of service calls in the execution of the PPC control loops.
	Scalability	Ability to maintain acceptable performance under increasing load. Evaluated by varying the number of active services, data traffic volume, and process model complexity.
Adaptability	Robustness	Ability to handle unexpected events such as temporary service unavailability or communication failures. Assesses effectiveness of fallback mechanisms, error-handling routines, and recovery strategies.
	Reconfigurability	Effort required to integrate new services or modify process models. Reflects “plug-and-play” capability essential for RMS.
Economic Viability	Operational Benefit	Quantitative assessment of the business value achieved through increased changeability based on DCF.

Together, these system-level metrics demonstrate the feasibility and practical value of the proposed service-oriented, task-agnostic PPC framework in real-world industrial settings, while also shedding light on the data integration strategies employed to address heterogeneous manufacturing environments.

7.2 Component Validation: Performance of Automated Data Integration

This section presents a quantitative validation of the LLM-supported data integration methodology developed in Section 5.3. The primary objective is to empirically assess the functionality and performance of the automated semantic mapping service, thereby providing a conclusive answer to Research Question 2: How to automate the integration of heterogeneous information systems to the greatest extent possible?

The evaluation focuses on measuring the accuracy, effectiveness, and efficiency of the generated mapping functions against a variety of real-world data models and mapping complexities with respect to selected state-of-the-art LLMs.

7.2.1 Experimental Setup

To ensure a rigorous and relevant evaluation, a comprehensive benchmark suite was established. The suite is composed of PPC-related data models and schemas, which will be later utilized in the system evaluation case studies. The used data models represent typical models used in industrial practice for PPC systems and services covering different a reas and tasks of PPC. Table 7.4 gives an overview of the utilized schemas.

Table 7.4: Overview of Schemas Used in the Evaluation

Schema	**Purpose / Description**
SDM Reference Model	Holistic production data model, adopted from (Ellwein & Neumann et al. 2023), serving as the global integration schema.
prodsys	Schema for production system simulation and optimization, used within the open-source software prodsys (Behrendt & Danz et al. 2025) [1].
Morpheus	Proprietary MES used at the LFGP; internal data model not publicly available.
Intralogistic Map Format	Specialized schema for describing facility layouts for transport route planning, based on Layout Interchange Format (LIF) (Verband Deutscher Maschinen- und Anlagenbau e. V. (VDMA) 2024).
Datenberg	Internal schema used for dashboard integration with the Datenberg analytics platform[2].
Kinexon	Proprietary Data model used in the API of the real-time location service Kinexon on the shopfloor[3].
Tulip	Schema utilized in the Tulip app-based MES system for Pick-by-Light integration at the LFGP[4].

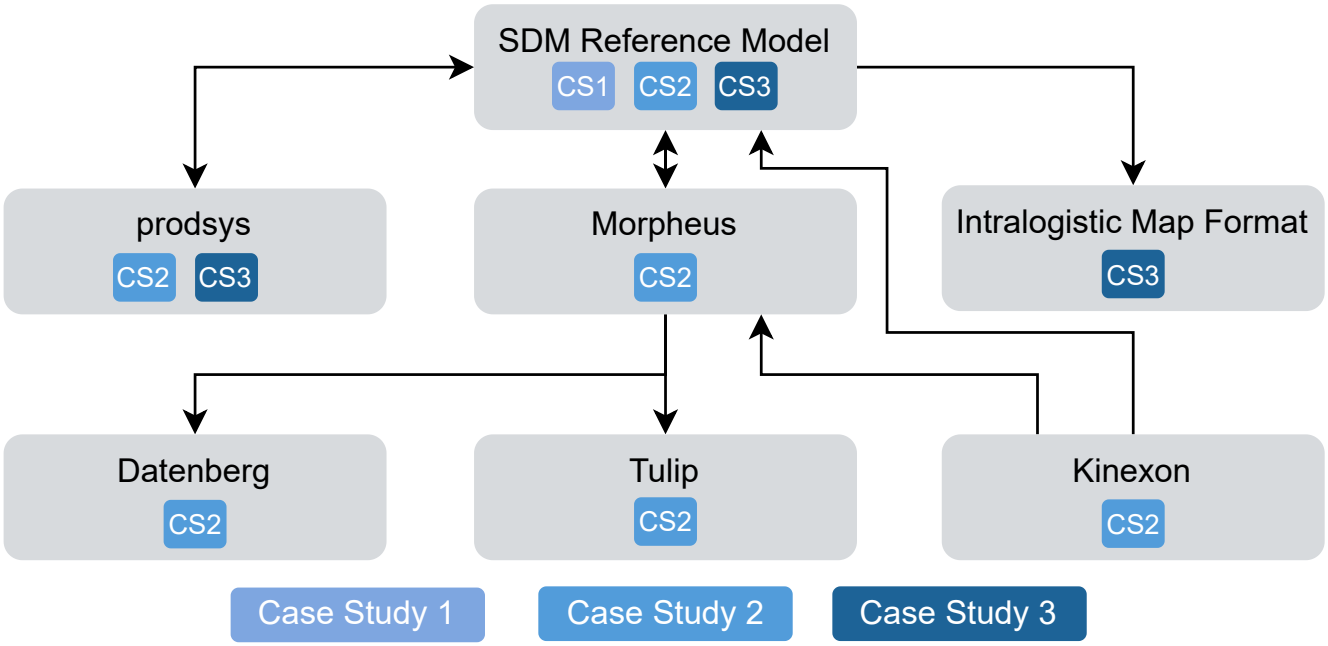

Figure 7.2: Overview of schema mapping relations in the schema integration benchmark and their association with the respective case studies.

The benchmark comprises a total of 41 mapping tasks derived from real integration requirements encountered in the Learning Factory and industrial case studies. Due to proprietary constraints of several involved systems, the concrete schemas and mappings are not publicly released. However, all schemas and mapping tasks used for the evaluation are documented in Appendix A7 to ensure methodological transparency.

Figure 7.2 provides an overview of the relationships between the schemas involved in the benchmark. It illustrates which schemas are connected by individual mapping tasks and indicates in which case studies each schema is utilized.

To analyze performance with respect to task difficulty, the mapping tasks were categorized into three complexity classes: 8 easy, 15 medium, and 18 hard mappings. Easy tasks involve the integration of individual classes with direct field mappings, medium tasks require transformations across multiple classes with moderate structural or semantic complexity, and hard tasks involve complex transformations affecting multiple classes or complete data models. The classification was primarily guided by the complexity of the integration and the associated manual effort required to implement the corresponding gold-standard mappers. These mappers serve as the definitive reference against which the LLM-generated outputs are compared.

[1]Available at: https://github.com/sdm4fzi/prodsys
[2]See https://www.datenberg.eu
[3]See https://www.kinexon.com
[4]See https://www.tulip.co

A detailed description of all mapping tasks and their individual difficulty classifications is provided in Appendix A7.

LLM Models and Cost Evaluation. To assess the influence of the specific LLM on the mapping generation process, a set of commercially or publicly available state-of-the-art models was selected for the study. Table 7.5 gives an overview of the utilized models and their licensing model, available context length, and cost. All models were accessed via OpenRouter[1], a platform for uniform access of LLMs from different providers.

Model	Publisher	License	Context length	Input cost	Output cost
DeepSeek R1	DeepSeek	Open source	163.8k	$0.5	$2.15
Gemini 2.5 Flash	Google	Proprietary	1.05M	$0.3	$2.5
Gemini 2.5 Pro	Google	Proprietary	1.05M	$1.25	$10
GPT-5	OpenAI	Proprietary	400k	$1.25	$10
Grok 4	xAI	Proprietary	256k	$3	$15

Table 7.5: Comparison of LLMs by vendor, license, context length, and input and output token pricing per million tokens.

LLM and Prompt Evaluation. The core of the experiment involves systematically testing each LLM's mapping performance while varying provided informational context in the components of the structured prompt, as described in Section 5.3.2. In the experiments, three different *mapping cases* were considered:

- **S (Schema-only):** Only the JSON schemas of the input and output data models were provided.
- **SED (Schema, Examples, Description):** Schemas were augmented with field descriptions, docstrings, and one or more examples of valid input and output data instances.
- **SEDR (Schema, Examples, Description, Rules):** The most enriched prompt, containing all information from SED, supplemented with explicit, human-written rules in natural language to guide complex transformations.

With this, an evaluation of the importance of the information context (documentation, examples, or additional rules) for resolving the mapping tasks of the benchmark is possible.

Benchmark Evaluation Execution. In execution of the benchmark evaluation, a request is sent to the LLM for each mapping task, LLM, and mapping case described above. The result is parsed and compiled, and executed on the data instances of the mapping task. If an error occurs here, the mapping is marked as non-executable, influencing the executable mapping rate.

[1] OpenRouter can be accessed at https://openrouter.ai/

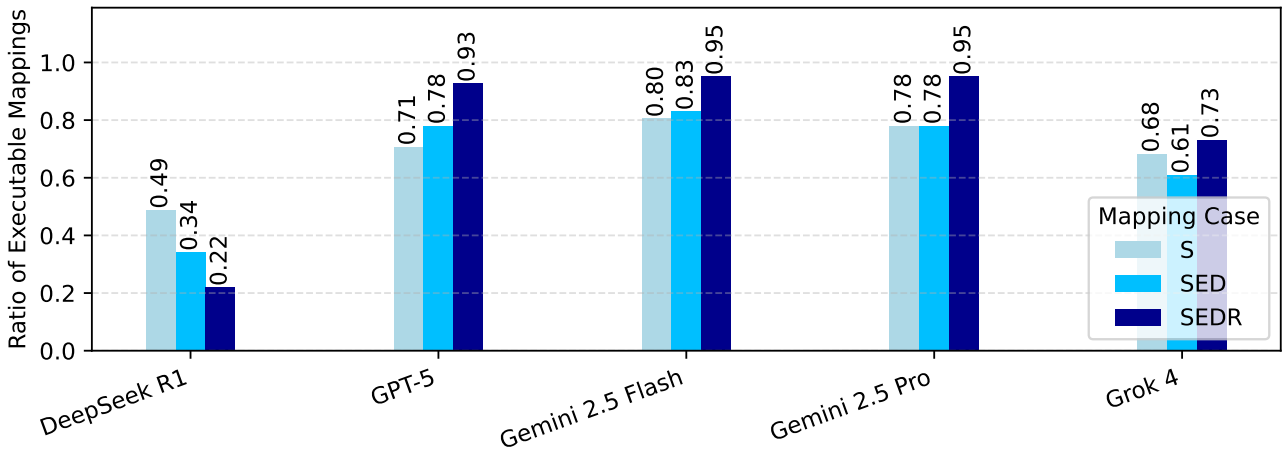

Figure 7.3: Ratio of executable mapping generation by LLM and mapping case.

If no error occurs, the output of the mapping function is compared against the Gold Standard using the accuracy metrics (Precision, Recall, F1-Score) defined in Section 7.1.1.1.

7.2.2 Results and Analysis

In the following, the results of the benchmark evaluation described above are presented. The analysis considers effectiveness, accuracy and efficiency of the approach based on the metrics defined in Section 7.1.1.1.

Figure 7.3 visualizes the effectiveness, i.e. the ratio of executable mapping functions generated by the LLMs, per LLM and mapping case. The results indicate that both Gemini models showed the best performance for the SEDR mapping case. Additionally, a clear trend is visible that the Effectiveness correlates with additional context provided to the LLM since SED and SEDR mapping cases outperformed mapping case S for all models but DeepSeek R1. Lastly, we see that the Gemini Models and GPT-5 have a significantly higher percentage of successful mapping generations than DeepSeek R1 and Grok 4. Detailed analysis showed that these models have a higher tendency to output a mapping function that results in compilation errors due to malformed code, requiring more complex preparation before compilation. An example of the benchmark study covering for one task the structured prompt, generated mapping function, generated documentation, and unit tests can be found in Appendix A3.

As illustrated in Figure 7.4, the accuracy is highly dependent on both the model choice and the prompting strategy. Gemini 2.5 Pro, GPT-5, and Gemini 2.5 Flash emerge as the top performers, achieving remarkable average F1-scores of 0.89, 0.85, and 0.84, respectively, when provided with the full context of schemas, examples, descriptions, and rules (SEDR). In contrast, DeepSeek R1 struggled, with its F1-score peaking at 0.34 in the schema-only

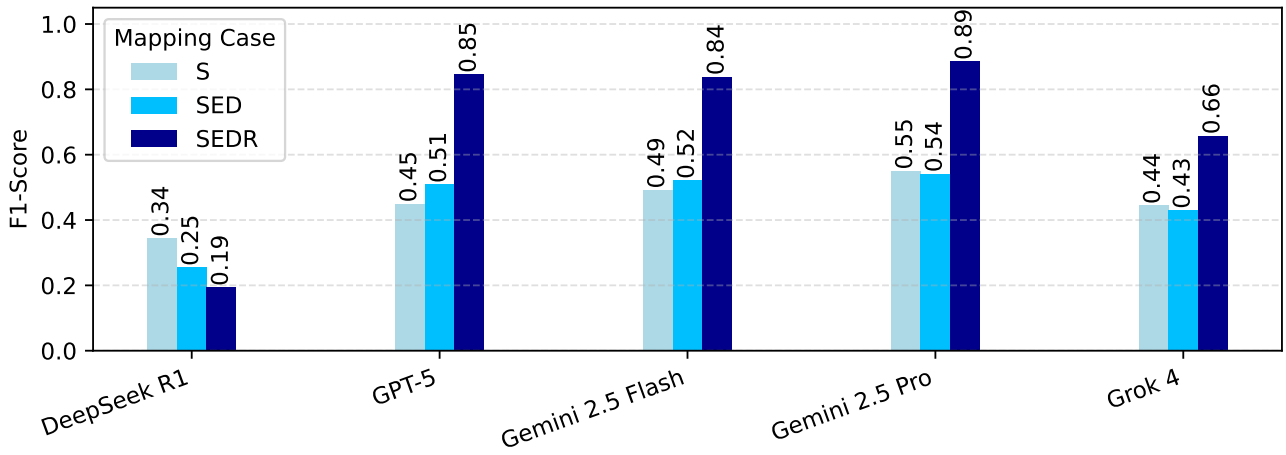

Figure 7.4: Aggregated F1-Scores by LLM and mapping case.

(S) case and dropping to 0.19 with the addition of rules (SEDR), suggesting difficulty in interpreting complex instructions. The lower performance of Grok 4 and DeepSeek R1 can, at least in part, be explained by the lower Effectiveness since unsuccessful mapping generations were considered with an F1-score of 0. Across all models, the SEDR case consistently yields the highest accuracy, demonstrating the critical importance of providing explicit, human-written rules for complex transformations. In Appendix A7, the detailed mean and standard deviation scores for accuracy metrics are displayed in Table A7.3.

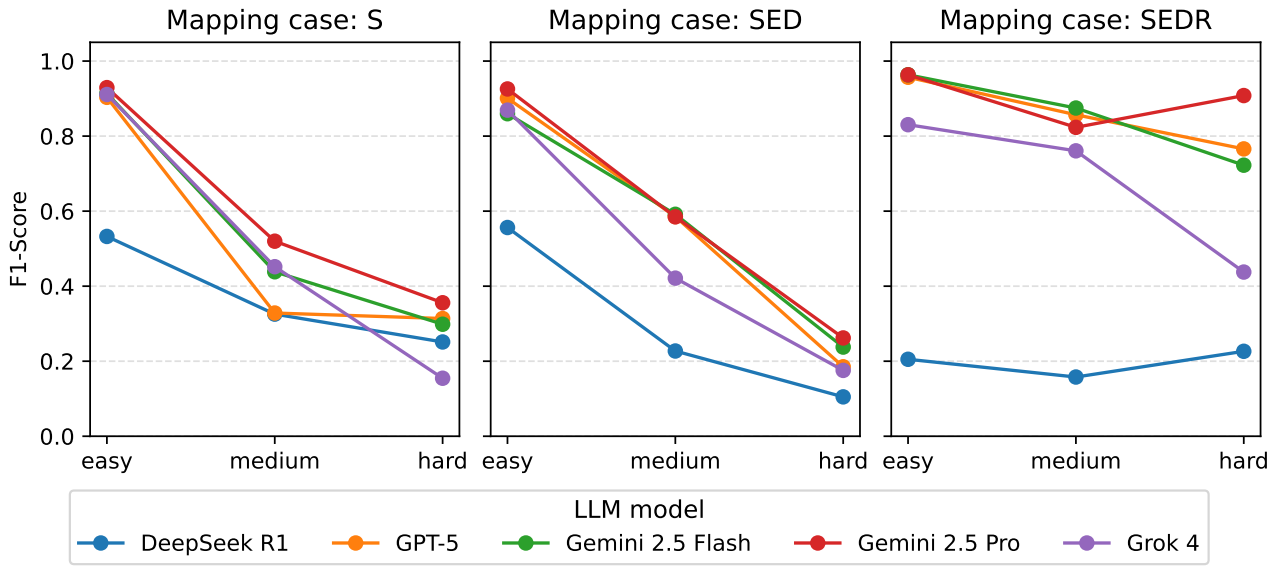

Figure 7.5: F1-Score by mapping case, LLM and mapping task difficulty.

The influence of task complexity is further detailed in Figure 7.5. For easy tasks, most models perform well regardless of the mapping case, often achieving near-perfect F1-scores. However,

as complexity increases to medium and hard, the performance for the S and SED cases declines sharply. The SEDR case shows remarkable resilience, maintaining a significantly higher F1-score even for the most difficult mappings. This confirms that while LLMs can infer simple mappings from schemas and examples, complex, domain-specific logic requires the explicit guidance provided by rules. However, this insight holds also for manual integrations: the higher the complexity of an integration task, the higher the understanding of the data models required to do the integration and the higher the amount of possibilities to do these mappings (Doan & Halevy et al. 2012).

In many of the hard mapping tasks, the LLMs generated correct Python functions exceeding several hundred lines of code. This underscores a key advantage of the approach: while crafting the natural language rules still requires expert knowledge, it is significantly less demanding and faster than manually writing, testing, and debugging hundreds of lines of code. The potential time savings for an integration expert can amount to several hours per hard mapping. During manual development of the benchmark mapping functions, implementation times ranged from several minutes for easy mapping tasks up to 12 hours for the most demanding mapping tasks.

Furthermore, the automatic integration approach was designed to support maintainability and manual review processes. Besides generating mapping functions, the approach allows to prompt LLMs to produce descriptive documentations and a set of unit tests for the generated mapping functions. This automated generation of documentation and validation code simplifies the review process for a human expert, allowing for quick verification of the mapping logic.

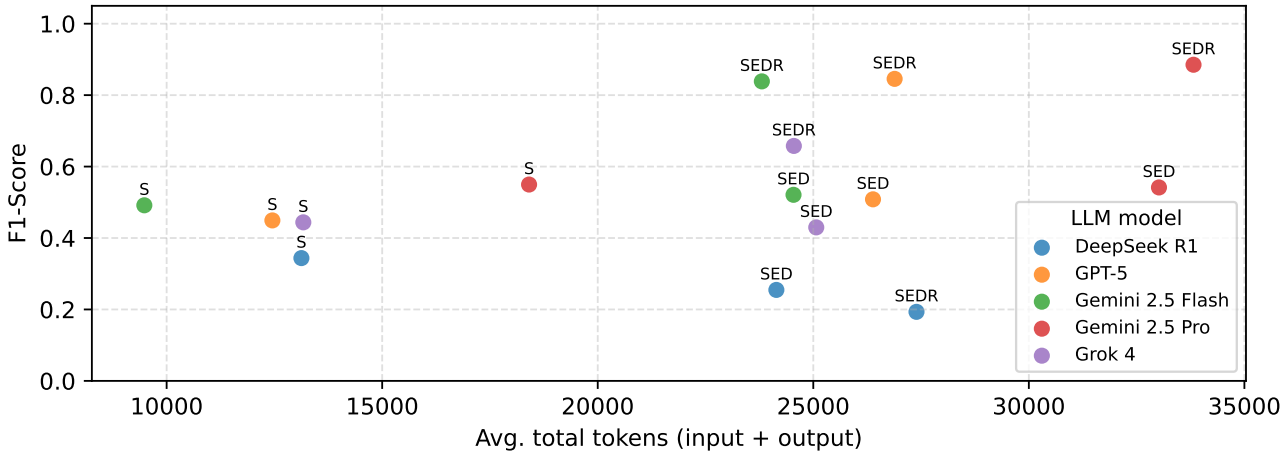

Figure 7.6: F1-Score vs. Average total tokens by LLM and mapping case.

Lastly, we want to analyze the efficiency of the approach. The relationship between performance and computational cost is depicted in Figure 7.6 and Figure 7.7. The first plot shows that higher performance (achieved with SED and SEDR cases) naturally comes with higher

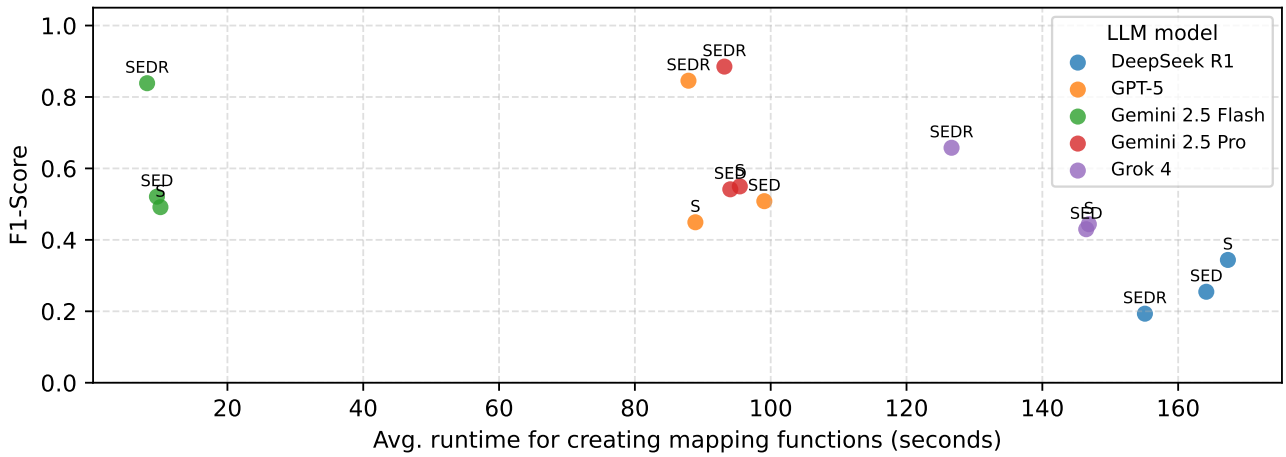

Figure 7.7: F1-Score vs. Average Runtime by LLM and mapping case.

token counts due to the richer context provided. Figure 7.7 presents a more practical analysis of efficiency. Here, clear trade-offs emerge:

- **Gemini 2.5 Flash** offers the fastest generation times, with an average runtime of approximately 10 seconds, but at the cost of slightly lower F1-scores.
- **Gemini 2.5 Pro** and **GPT-5** occupy a balanced position, delivering top-tier accuracy with moderate runtimes (around 90-100 seconds).
- **DeepSeek R1** and **Grok 4** exhibit longer runtimes (over 120 seconds) without a corresponding increase in performance, making them less efficient for this task.

When considering the overall performance of the LLMs, we see that a small accuracy improvement of Gemini 2.5 Pro and GPT-5 comes at the cost of a reduced efficiency due to longer and more costly mapping function generations when compared to Gemini 2.5 Flash. As a practical guidance we therefore suggest to go with the faster and cheaper models that have a sufficient context length. Especially Gemini 2.5 Flash showed here a suiting trade-off of performance, speed, and cost.

In summary, generating mapping functions with the LLMs for the benchmark mapping tasks takes between a couple of seconds up to 2.5 minutes and costs between a fraction of a cent for easy tasks up to $0.41. When comparing this to current manual procedures that require several hours spend for programming, the potential for time savings and cost reduction is immense.

To estimate the potential time and cost reduction, we assume that writing one hundred lines of mapping code for the manually created gold standard requires one hour of programming time. Additionally, we assume that specifying supplementary natural-language rules takes place at

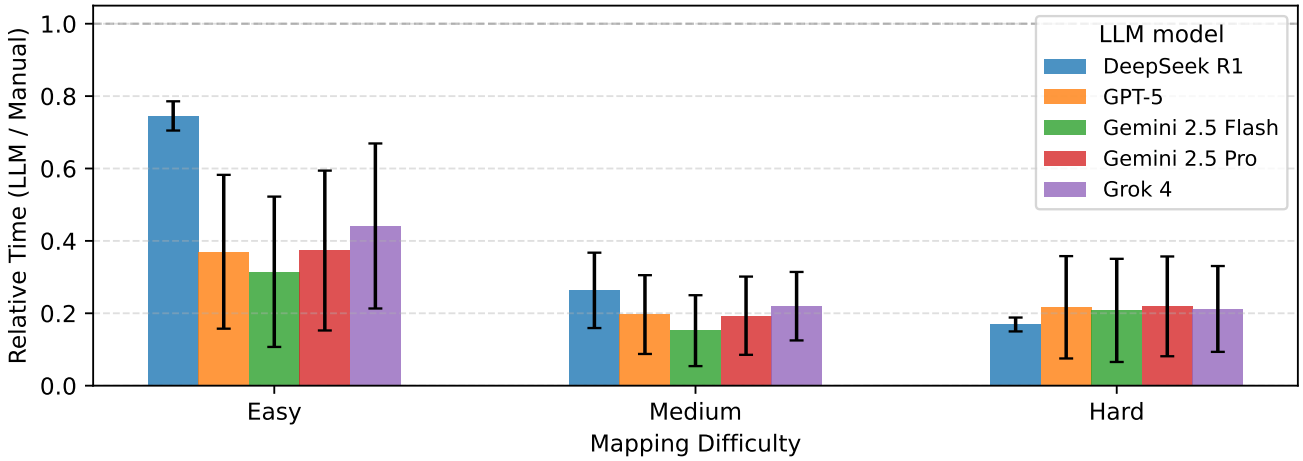

Figure 7.8: Relative time required of the LLM-based integration approach compared to a manual integration by human experts.

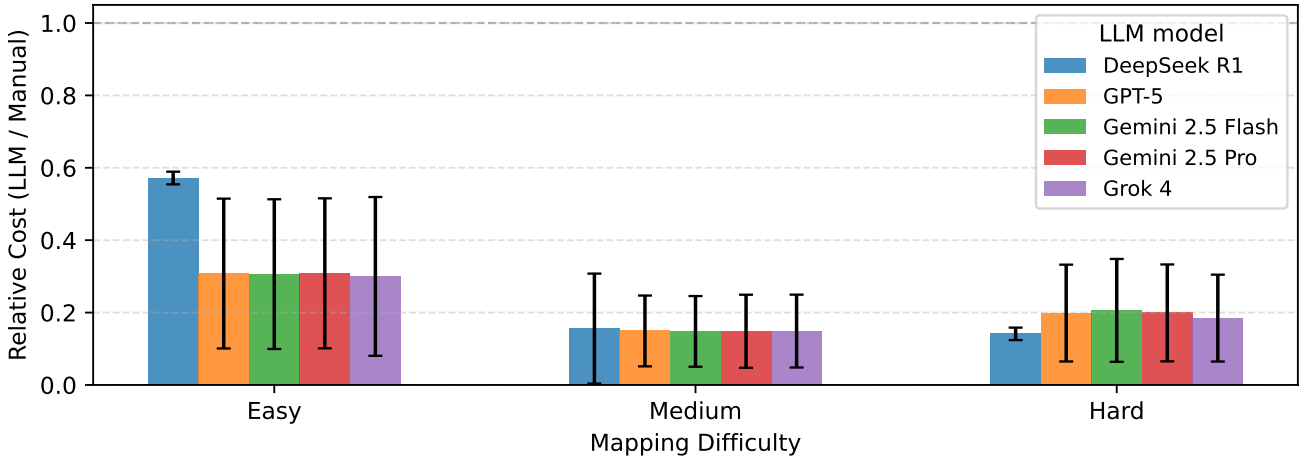

Figure 7.9: Relative cost of the LLM-based integration approach compared to a manual integration by LLMs and mapping difficulty.

a rate of ten words per minute. Labor costs are calculated using a typical software-engineering rate of $100 per hour. These assumptions are deliberately conservative, since manual coding of complex mappings is often slower in practice, while formulating natural-language rules is typically faster.

Based on these assumptions, the cost of manually implementing the gold standard is compared to the combined cost of providing the additional rules and executing the LLM inference required to generate the corresponding mapping function. For this quantitative cost and time analysis, only executable mapping results and the SEDR case are considered.

Figure 7.8 visualizes the ratio of time required to generate the mapping functions with the LLM-based approach compared to the gold standard time requirement. The figure shows that LLM-

based mapping substantially reduces the effort needed to produce executable transformations. In easy mapping cases, the savings are modest because the manual implementation is short and the fixed overhead of preparing additional rules and running the model constitutes a noticeable portion of the total LLM-based effort. As the mapping tasks become more complex, the advantage of the LLM approach grows: medium-difficulty mappings clearly benefit from reduced human effort, and for hard mappings the reduction becomes pronounced. The most capable models require only around 20% of the time needed for manual implementation, demonstrating that LLMs scale far more efficiently with increasing complexity. This trend is reflected across all examined models, with the most advanced models yielding the largest reductions.

Figure 7.9 considers the ratio of cost between the LLM-based approach and the manual implementation of the gold standard, which reveals the same pattern as time reduction analysis before. When translating the time requirements into monetary values and adding the model-specific inference costs, the LLM-based approach remains highly competitive. For simple mappings, costs are similar or slightly lower than manual coding, while for medium-difficulty mappings noticeable savings emerge. For hard mappings, the difference becomes substantial: manual implementation in these cases typically ranges between $100 and $1000, whereas the LLM inference cost remains only a few dollars, rendering it negligible relative to human labor. Overall, the results illustrate that the cost of LLM-assisted mapping not only scales favorably with task difficulty but becomes increasingly advantageous as the complexity of the schema mappings increases.

7.2.3 Interim Conclusion

The quantitative evaluation demonstrates the viability of the proposed LLM-supported data integration methodology. The experiments confirm that modern LLMs can successfully automate the generation of semantic mapping functions between complex, domain-specific data models. The results highlight that performance is a function of the chosen LLM, the intrinsic complexity of the mapping task, and, most critically, the quality of the contextual information provided in the prompt. While simple mappings can be resolved with basic schema information, complex transformations require richer context, and the inclusion of explicit rules (SEDR mapping case) is essential for achieving high accuracy in real-world scenarios. Lastly, we see the potential for immense cost and time savings when comparing the obtained mapping time and cost to the expected costs of generating these integrations manually. These findings provide a strong affirmative answer to Research Question 2. The study demonstrates that LLM-based services can automate a significant portion of the semantic integration process, drastically reducing the manual effort and cost required for data integration. The approach represents a paradigm shift from fully manual implementation to expert-guided automation,

where the role of the integration specialist is to provide high-level context and rules rather than writing low-level code. The method is validated as a core, functional component of the overall service-oriented PPC framework.

7.3 Component Validation: Performance of Service Orchestration Capabilities

This section presents an isolated, quantitative evaluation of the OE. The primary objective is to validate its performance, scalability, and adaptability against established academic benchmarks. By rebuilding and evaluating standardized problems, the capabilities of the OE can be objectively measured, allowing its performance to be positioned relative to existing technologies. This validation is a critical prerequisite for assessing its role in the later system evaluations. The evaluation is structured into two distinct parts. First, a demonstration of the advanced orchestration capabilities, such as dynamic binding and fault tolerance, is provided. Second, a benchmark study is conducted to assess the performance overhead of the core orchestration logic.

7.3.1 Experimental Setup and Methodology

To ensure a fair and reproducible comparison, all experiments were conducted within a controlled environment. The experiments were performed on a dedicated machine equipped with an Intel Core i9-12900K CPU (3.20GHz, 24 threads) and 64GB of RAM. The OE and all mock services were containerized using Docker to ensure isolation and reproducibility.

Although the absolute hardware specifications differ from the original benchmark environment, the effective computational resources available to the OE are comparable. In the original setup, the WfMS was deployed on a machine with 12 CPU cores clocked at 800 MHz, yielding an aggregate compute capacity of approximately 9.6 GHz.

To ensure comparable effective compute, the OE was explicitly constrained to eight worker processes. Moreover, the Python-based implementation further limits effective CPU utilization due to the Global Interpreter Lock (GIL), restricting execution to approximately one third of the available CPU capacity per worker process. As a result, the effective compute budget utilized by the OE in the benchmarks is intentionally constrained to an estimated aggregate compute budget on the order of 8-9 GHz and remains on the same order of magnitude as in the original benchmark, despite the higher nominal clock frequency of the host CPU.

This deliberate limitation ensures that the measured performance reflects orchestration and execution behavior rather than advantages stemming from excessive hardware parallelism.

7.3.2 Demonstration of Orchestration Capabilities

To demonstrate the advanced capabilities of the OE, a representative industrial process model was modeled, as defined in the provided SAPN definition, as displayed in Figure 7.10. This process model, named "ERP to Material Planning Process Model," orchestrates a sequence

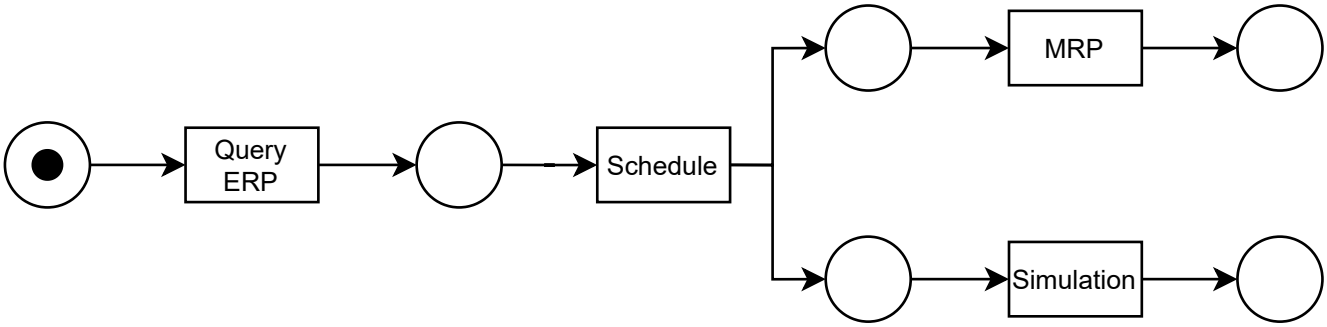

Figure 7.10: Visualization of the demonstration SAPN covering ERP-based scheduling, material planning, and simulation.

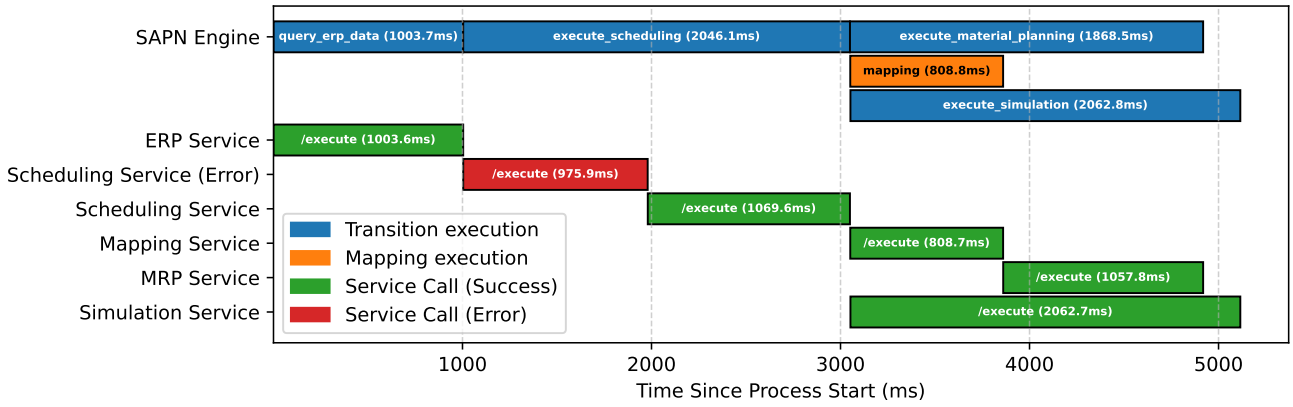

Figure 7.11: Execution timeline of the demonstration process model, showcasing parallel service calls, a service call error followed by a successful retry (fallback mechanism), and an automated mapping execution.

of services to fetch order data, schedule production, plan material requirements, and simulate the outcome. The execution of this process model, visualized in Figure 7.11, showcases several key features of the framework.

Concurrent Service Execution. After the initial *Query ERP* transition, the process model splits. The *Schedule* transition produces tokens for two output places, enabling the parallel execution of the *MRP* and *Simulation* transitions. The timeline clearly shows the overlapping execution of the corresponding MRP and Simulation services.

Dynamic Binding and Service Fallback Mechanism. The timeline illustrates a fault-tolerance scenario. The first attempt to call the Scheduling Service fails (red bar) deliberately, simulating a failure which could occur in practice due to a service crash or network issue. The OE, through its integration with a Service Registry, dynamically binds to an alternative, healthy service that provides the requested *Schedule* capability and successfully completes the call on the second attempt.

Automatic Schema Mapping. The transition from *Schedule* to *MRP* requires a data transformation. The mapping (orange bar) step shown in the timeline represents the automated execution of a data mapping function discovered by the engine in the registry to align the output schema of the scheduling service with the input schema required by the MRP service.

This demonstration confirms the engine's ability to handle complex, real-world orchestration scenarios involving concurrency, fault tolerance, and data schema heterogeneity. The task-agnostic and schema-aware properties of SAPNs proved as a strong formalism to enable dynamic service binding, increasing fault-tolerance and enabling automatic resolving of data heterogeneity.

7.3.3 Benchmarking of Performance and Scalability

To quantitatively assess the performance overhead and scalability of the approach, a benchmark based on the well-established patterns from Skouradaki & Ferme et al. (2016) is implemented.

The patterns used in this benchmarks represent fundamental control-flow structures found in most process models. An overview of the patterns of this benchmark is given in Table 7.6. A more concise visualization and description of the individual petri nets can be found in Appendix A8.

Pattern	**Description**
Sequence (SEQ)	A simple chain of two sequential service calls.
Exclusive Choice (EXC)	A conditional gateway that routes the process model to one of two branches.
Parallel Split & Synchronization (PAR)	Two services executed concurrently, with the process model synchronizing before continuing.
Arbitrary Cycle (CYC)	A loop structure that executes a service call a predefined number of times.
Complex Synchronization (EXT)	A more complex pattern involving multiple parallel and sequential steps.
Mixed (MIX)	A composite process model combining all the above patterns to represent a complex, realistic process.

Table 7.6: Overview of the process model patterns based on Skouradaki & Ferme et al. (2016).

These patterns were executed with the OE and compared against the published results for three other systems from the original study: two commercial Workflow Management Systems (WfMS A and WfMS B) and one open-source system (Camunda).

Measurements were performed under the same conditions as the published benchmark values: 1500 individual clients issuing requests at an interval of 1 seconds. The benchmarks ran for 600 seconds with a 30-second warm-up period.

The aggregated results of the micro-benchmark comparison are presented in Figure 7.12. The figure provides a side-by-side comparison of the four systems (WfMS A, WfMS B, Camunda, and the OE) across the six process model patterns for the three key metrics.

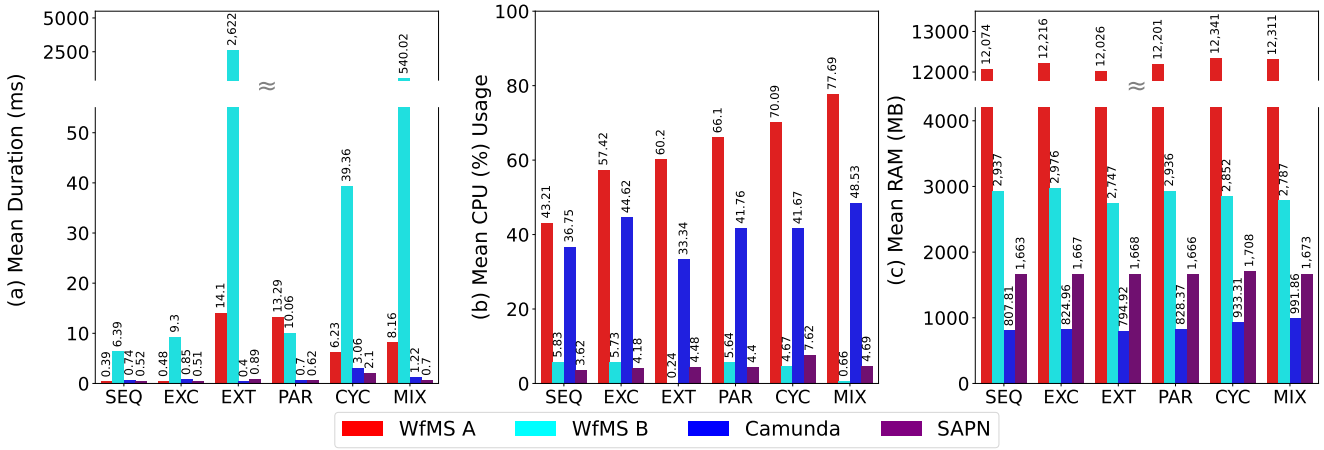

Figure 7.12: Comparison of performance metrics for the micro-benchmark experiments. (a) Mean process model execution duration (latency) in milliseconds. (b) Mean CPU utilization percentage. (c) Mean RAM consumption in megabytes.

As shown in Figure 7.12(a), the OE consistently exhibits very low mean execution duration across all patterns. For the simpler patterns (SEQ, EXC, PAR), its latency is sub-millisecond and highly competitive with Camunda, while being significantly faster than WfMS A and WfMS B. In the more complex EXT and MIX patterns, where WfMS B shows extremely high latency (2,622 ms and 540 ms, respectively), the OE maintains its low overhead, with durations of 0.89 ms and 0.70 ms.

Regarding resource consumption, the OE demonstrates remarkable efficiency. Figure 7.12(b) shows that its CPU usage is the lowest across all patterns, with a mean below 8% across all experiments. This is in stark contrast to the other systems, particularly WfMS A, which consistently uses over 40-70% CPU. Similarly, as seen in Figure 7.12(c), OE's memory footprint is minimal (under 2 GB in all cases), whereas WfMS A and B consume several gigabytes of RAM, indicating a much heavier architecture. Yet, Camunda showed here better performance considering memory usage. A detailed overview of the obtained performance metrics can be found in Appendix A8 in Table A8.1.

The sustained throughput of the OE is visualized in Figure 7.13. For most patterns, the engine consistently processed over 1475 process model instances per second. The overall

throughput for the five main patterns (SEQ, EXC, EXT, PAR, CYC) was stable at approximately 1412 process models per second (see Table A8.1). The MIX pattern, being more complex, resulted in a slightly lower but still stable throughput. This demonstrates the engine's capability to maintain high and predictable performance over time.

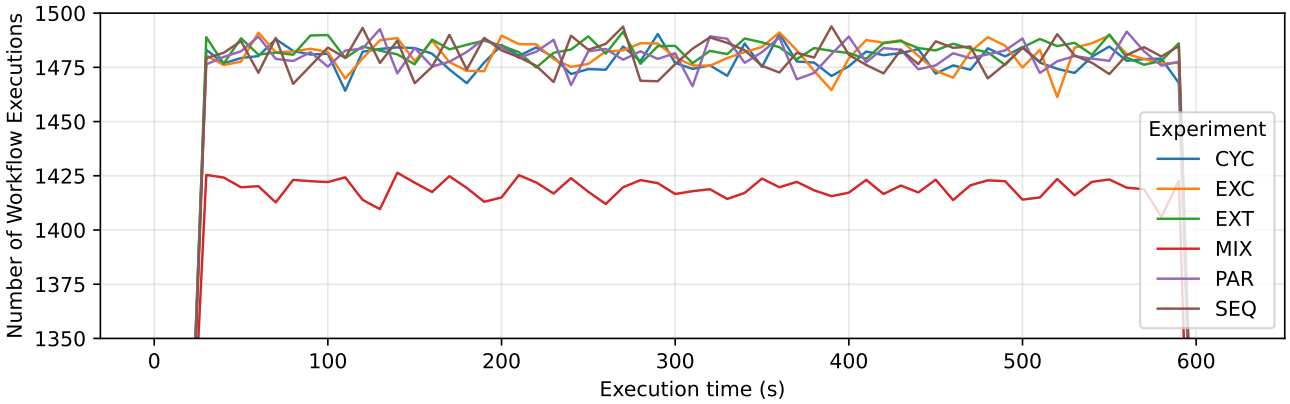

Figure 7.13: Throughput of the OE measured as completed process model executions per second during the benchmark experiments.

The results clearly indicate that the OE is lightweight and performant. Its low latency, sub-millisecond for most patterns, can be attributed to its event-driven, in-memory processing architecture, which avoids the overhead associated with heavy state-persistence models. This is particularly evident in the minimal CPU and RAM consumption, which suggests that the engine imposes negligible overhead on the system, allowing resources to be dedicated to the actual service business logic. The high and stable throughput confirms that this lightweight design does not compromise scalability and that the engine is well-suited for high-frequency orchestration scenarios typical in I4.0 environments. Compared to the benchmark systems, the SAPN-based OE offers a competitive performance profile, combining the low latency of systems like Camunda with low resource consumption, making it a highly efficient solution for service orchestration.

7.3.4 Interim Conclusion

This section has provided a rigorous, component-level validation of the OE as the execution core of the proposed service-oriented PPC framework. The qualitative demonstration confirmed the OE's ability to execute SAPN-based process models with concurrent transitions, late service binding, automated schema mapping, and runtime fallback, illustrating its suitability for dynamic and fault-tolerant orchestration in heterogeneous industrial environments.

The quantitative micro-benchmarking results substantiate these capabilities with strong performance evidence. Across all evaluated workflow patterns, the OE achieved sub-millisecond

mean execution latencies for simple patterns (SEQ, EXC, PAR) and maintained consistently low overhead even for complex patterns (EXT, MIX), with execution times below 1 ms. Under a sustained load of 1500 concurrent clients, the engine delivered a stable throughput of approximately 1,400–1,500 process model executions per second, demonstrating predictable scalability over time.

From a resource-efficiency perspective, the OE showed very low CPU utilization (below 8% on average) and a memory footprint under 2 GB across all experiments, significantly outperforming the evaluated commercial WfMS solutions and matching or exceeding the efficiency of lightweight open-source alternatives. These results indicate that the OE's event-driven, in-memory execution model imposes minimal orchestration overhead, allowing computational resources to be primarily allocated to service execution rather than workflow management.

Overall, the combined qualitative and quantitative findings confirm that the OE is a lightweight, scalable, and high-performance orchestration engine, capable of supporting both high-frequency control loops and complex planning workflows. This validated performance profile establishes a solid foundation for its integration into the subsequent system-level evaluations in industrial PPC scenarios.

7.4 Case Study 1: Vertical Integration and Plug-and-Play Capability on a Modular Station

This case study evaluates the integration of the proposed framework in a real industrial demonstrator: a modular and reconfigurable disassembly station equipped with a handling robot and interchangeable process modules (Behrendt & Wurster et al. 2023). The scenario highlights the core capabilities of the framework: automated vertical IT/OT integration, dynamic system understanding via AAS, and Workflow-driven execution. By integrating self-descriptive modules, heterogeneous communication protocols, and automated planning services, the case study demonstrates how the framework enables seamless reconfiguration and execution without manual intervention. The following subsections describe the scenario, execution, results, and key findings in detail. The implementation of this case study was performed in A_Bartenbach (2022) under the supervision and technical guidance of the author.

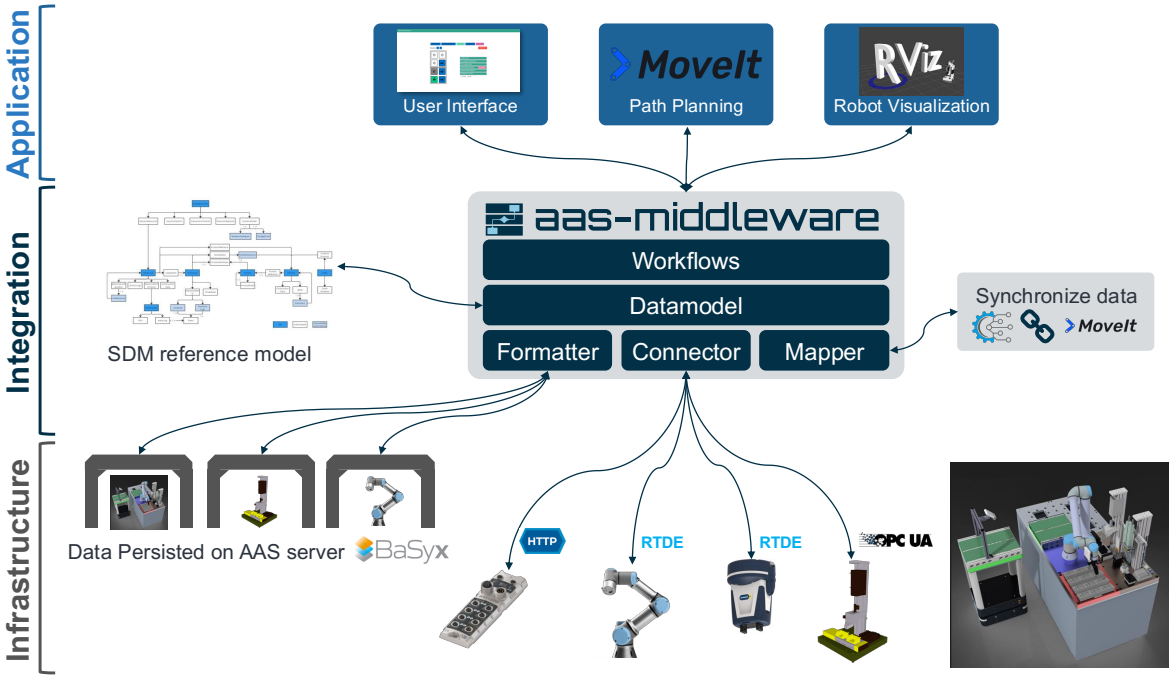

Figure 7.14: Conceptual overview of the modular automated disassembly station showing connections managed by aas-middleware.

7.4.1 Scenario and Objective

This case study demonstrates the framework's capabilities using a modular disassembly station, designed for flexible and reconfigurable manufacturing. The station, conceptually illustrated in Figure 7.14 and physically depicted in the photograph within the same figure, comprises a handling robot and various process modules equipped with universal hardware and electrical interfaces. These modules can be flexibly relocated, with RFID sensors at each

potential location automatically identifying the module, its precise position, and orientation. A key enabler for this adaptability is the AAS provided by each station module, which offers a self-description detailing its specific capabilities, processes, handling points, and communication interfaces (e.g., OPC UA or Robot Teach Pendant Ethernet (RTDE)). Moreover, to enable reconfigurable operation of the station, a set of Workflows are utilized in the Middleware to connect dynamic trajectory planning based on a Robot Planning Service (RPS) (see (Behrendt & Wurster et al. 2023) for more details) with flexible process execution.

The primary objective of this case study is to prove the framework's capability for end-to-end, automated vertical integration. This entails managing the entire information flow from a high-level command from a web-based User Interface (UI) (see Appendix A9 for a detailed view) down to the PLC-level control of the physical assets (robot, process modules). A secondary objective is to demonstrate the plug-and-play capability of the framework, necessary for RMS and associated PPC systems. This assesses how easily new or replaced modules can be integrated and how the system dynamically adapts its operations without manual reconfiguration.

7.4.2 Execution

The modular disassembly station was integrated into the proposed framework, leveraging self-descriptive AAS instances as the foundation for automated system understanding and control. The aas-middleware, acting as the central integration and orchestration hub, implemented the core components of the framework for this scenario.

The framework's Datamodel maintains a real-time digital representation of the station's state, including module configurations and operational statuses. This is realized by utilizing a standardized data model, i.e. the SDM Reference Model (Ellwein & Neumann et al. 2023), represented as AAS instances and stored persistently on a BaSyx server. This standardized representation is crucial for achieving semantic interoperability across the system.

The Integration components, i.e. Connectors, Formatters and Mappers, of aas-middleware bridge the inherent heterogeneity between various systems, as described in Table 7.7. These components are used for the orchestration of information flow in the Middleware with Workflows. The Workflows provide high-level interfaces to control the station based on capabilities such as *get station setup* or *execute process*. These Workflows are used in the UI to monitor and control the station, involving three key types.

1. Update station configuration. RFID Connectors identify modules and their positions and orientations. This data is used in BaSyx Connectors for updating the central Datamodel

Table 7.7: Integration Components Used in Case Study 1

Component Type	**Interface / Name**	**Function**
Connectors	RTDE (Robot)	Control robot joint movements based on planned trajectories.
	RTDE (Gripper)	Activate and control robot gripper movement.
	OPC UA (Process Execution)	Trigger execution of process steps on process modules.
	OPC UA (Process Status)	Observe and read process execution status for synchronization.
	OPC UA (RFID)	Read RFID values to identify module types, positions, and orientations.
	BaSyx (AAS Client)	Read and write AAS data for module self-descriptions and station configuration.
	HTTP (RPS Client)	Communicate with the RPS to request trajectory planning.
Formatters	Datamodel $\rightarrow$ AAS	Transform Middleware reference model objects into the AAS metamodel for persistence on BaSyx.
	AAS $\rightarrow$ Datamodel	Convert AAS data into reference model objects for use in the Middleware.
Mappers	Reference Model $\rightarrow$ RPS	Map the SDM-based station self-description to the internal input schema required by the RPS.

(AAS instances) with the current physical layout. This Workflow exemplifies the plug-and-play capability at the lowest level.

2. Plan process sequence. Upon receiving a high-level target process sequence from the UI, the Middleware queries the Datamodel for suitable process modules based on their provided capability. Their locations are then used to determine necessary intermediate handling steps (robot movements) based on module hardware interface specifications extracted from their AAS. For each handling step, the Workflow invokes an external Robot Planning Service (RPS), using Mappers for precise data translation. The RPS returns a suitable robot trajectory. The Middleware then combines these trajectories with commands to Connectors for process module activation and robot gripper activation into a complete, executable sequence. This process exemplifies automated, context-aware planning.

3. Execute process sequence. This Workflow executes the planned sequence, sending commands in the correct order to the Robot (via RTDE Connector), Robot gripper (via RTDE Connector), and Process Modules (via OPC UA Connector). It leverages synchronization mechanisms (OPC UA boolean flags) to ensure proper synchronization and precise timing between robot handling actions and module executions, which is crucial for reliable execution and collision avoidance.

The entire process, from module detection to task execution, relies fundamentally on AAS self-descriptions, enabling the system to automate the understanding of module capabilities and interfaces, facilitating dynamic planning and execution without manual control logic reconfiguration.

7.4.3 Results and Findings

The performance and behavior of the framework, specifically aas-middleware, were assessed using runtime data from two operational scenarios, further detailed in Behrendt & Bail et al. (2025). The primary objectives were to: (1) demonstrate effective orchestration involving multiple, heterogeneous IT and OT components, showcasing real-time interaction and synchronization; and (2) quantify communication characteristics (frequency, latency, message size) to evaluate its ability to bridge the IT/OT gap and handle demanding planning tasks.

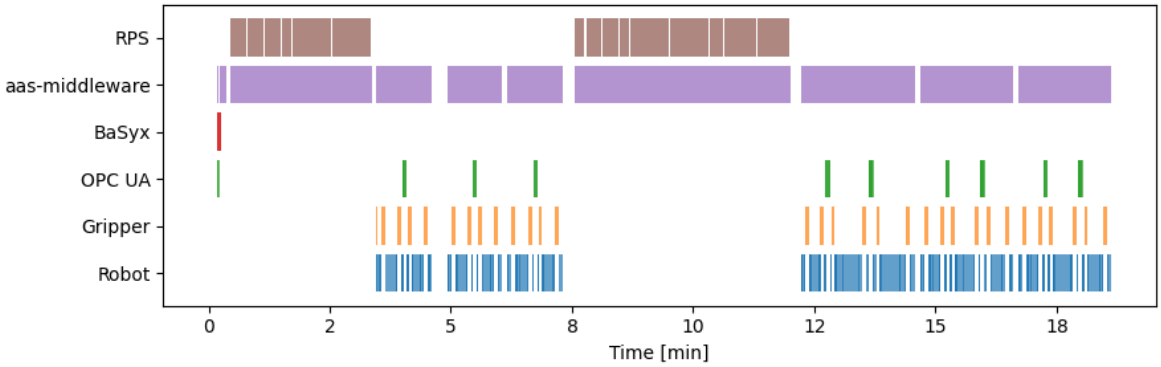

Figure 7.15: Event timeline for a planning and execution sequence involving replanning between process steps.

The event timelines provide critical insight into the dynamic orchestration capabilities. Figure 7.15 illustrates a scenario involving replanning between two process sequences. It shows how the framework first updates the station configuration (visible through BaSyx and OPC UA interactions). Subsequently, the RPS is initiated multiple times to generate robot trajectories. The execution trigger from the UI then prompts the framework to coordinate the Robot, Gripper, and OPC UA modules for execution. After the first process sequence, a second one is provided from the UI, which is then planned and executed. The second OPC UA interaction

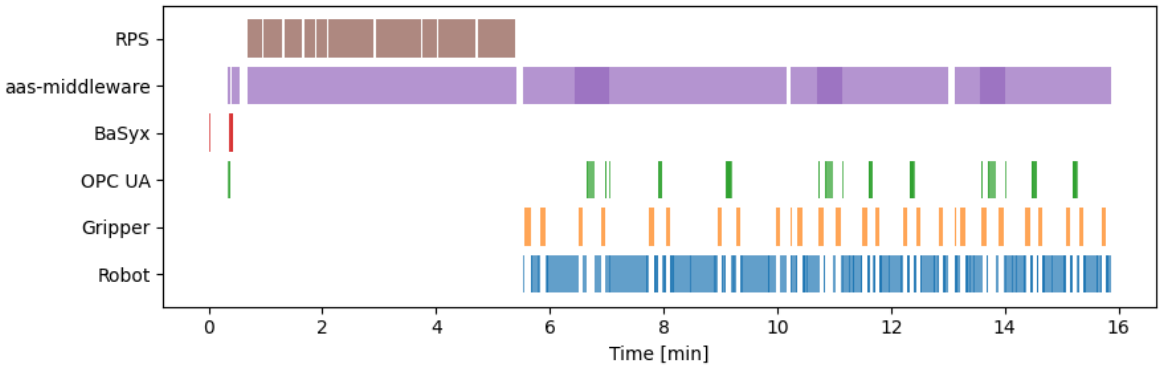

Figure 7.16: Event timeline for the planning and execution of a complex three-step process sequence.

during execution highlights the boolean-flag synchronization mechanism used to align robot handling with process module execution. This demonstrates the system's adaptability and the Middleware's central role in managing dynamic changes and orchestrating interactions among diverse, heterogeneous system components.

After performing this action, a demo module was repositioned for a different setup to demonstrate dynamic reconfiguration capabilities and intelligent replanning of the execution. Figure 7.16 details a subsequent planning and execution of a three-step process sequence. By first reading the station configuration via OPC UA and updating BaSyx AAS data, the subsequent planning steps consider the updated configuration. These timelines emphasize the framework's capability to manage complex operations in an automated manner. Both timelines validate the framework's capacity to execute complex, dynamically generated plans across heterogeneous resources and protocols. Moreover, automated configuration and operation of the station based on AAS self-descriptions fulfill the need for robust real-time orchestration of production resources.

Analysis of interaction frequency (Figure 7.17) reveals a high volume of exchanges managed by aas-middleware, with the robot having the most interactions. This reflects the robot's pivotal role in physical execution and underscores the framework's continuous real-time coordination with station components, including triggering process modules via OPC UA signals.

Examining the communication characteristics latency (Figure 7.18) and message size (Figure 7.19) reveals distinct patterns confirming effective IT/OT integration. OT communication (Robot via RTDE, Process Modules via OPC UA) exhibits very low latency, essential for real-time control and synchronization, with small message payloads (status updates, triggers).

It should be noted here that the interaction latencies for Gripper and Robot cover not only the trigger but also the execution time, hence the values are in the range of seconds. However,

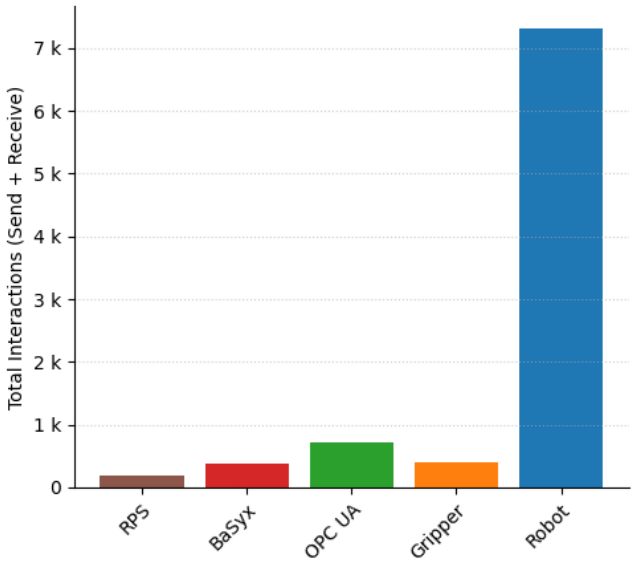

Figure 7.17: Number of Interactions per Resource managed by aas-middleware during the evaluated period.

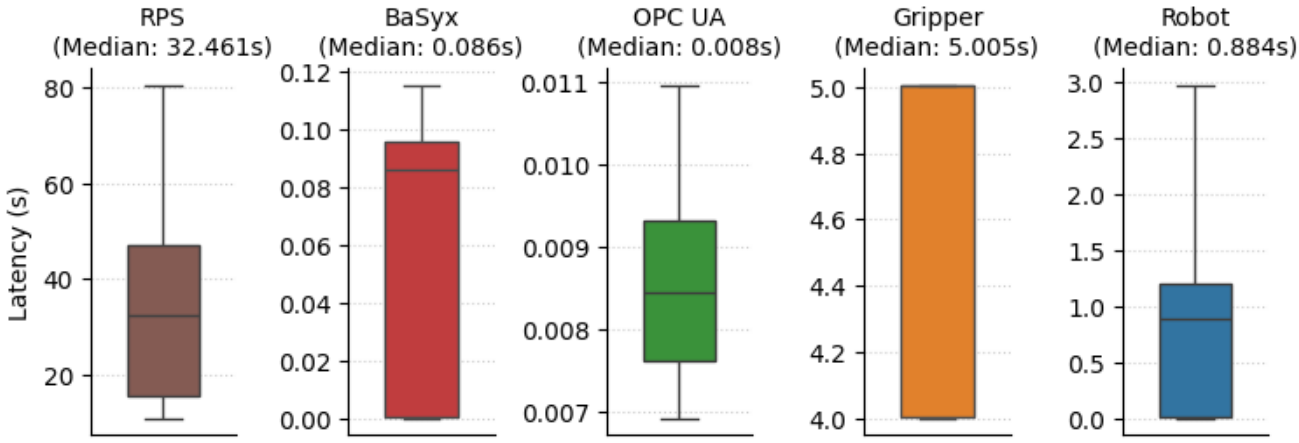

Figure 7.18: Distribution of interaction latencies (send/receive round-trip) for different resources.

the latencies of OPC UA interactions, with a median below 9 milliseconds, demonstrate the feasibility for fast and real-time synchronization between IT and OT systems. In contrast, IT service interactions (RPS, BaSyx) show higher latencies, which are acceptable for less time-critical tasks, and often involve larger payloads (configuration data, complex plans). This clear differentiation demonstrates aas-middleware's proficiency as a bridge between IT and OT, effectively managing the disparate communication requirements inherent in smart manufacturing environments.

The plug-and-play capability was assessed by the ease with which new or relocated modules could be integrated. The reliance on AAS for self-description meant that once a module's

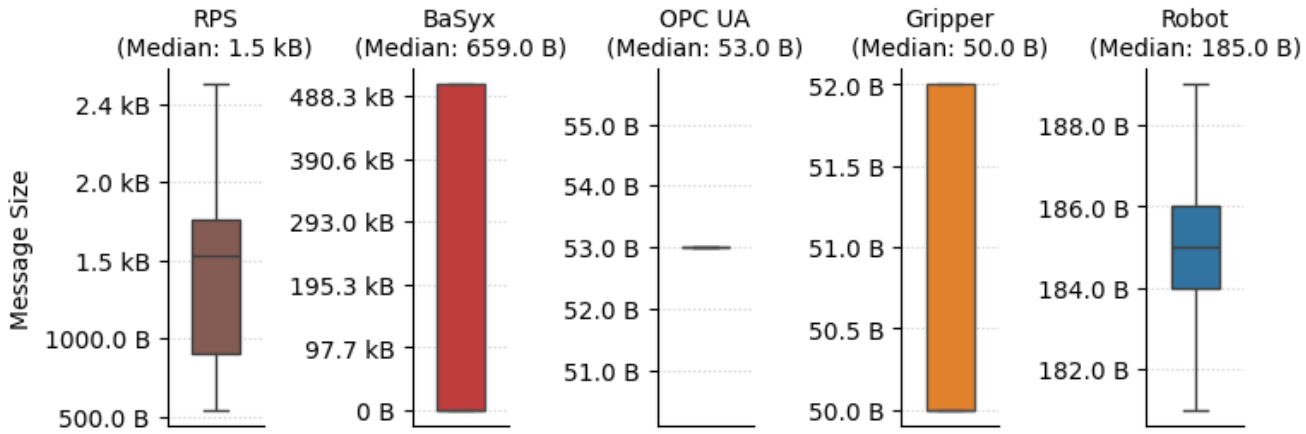

Figure 7.19: Distribution of message sizes for different resources.

AAS was available on the BaSyx server, aas-middleware could automatically discover its capabilities and integrate it into the Workflow execution.

When a process module was replaced with another, the system automatically adapted the planning and execution Workflows without requiring any code changes or manual operations. All information for the RPS and operation execution is contained in the AAS-based SDM reference model, making the station's operation with the Workflow-based PPC system agnostic to utilized process modules or setup. This seamless integration highlights the power of standardized self-descriptions with AAS for reconfigurable and automated PPC.

7.4.4 Interim Conclusion

In summary, the results validate the objectives of vertical integration and plug-and-play functionality by showing that the proposed framework effectively addresses the central challenges of modern manufacturing environments. Data heterogeneity is resolved through the modular Integration Layer which processes diverse formats, protocols, and schemas. System fragmentation is overcome by the Middleware that seamlessly integrates IT and OT components.

Automation is enabled through the use of AAS self-descriptions within automated Workflows, enabling dynamic, on-demand planning and execution. Real-time synchronization requirements are met through low-latency OT communication and an explicit synchronization mechanism.

Finally, plug-and-play capability is demonstrated by the system's ability to integrate and adapt to changing module configurations based on their AAS descriptions. Together, these results confirm the framework's effectiveness as an IT/OT integration bridge capable of handling both rapid OT interactions and complex IT service calls, thereby achieving advanced vertical integration and high reconfigurability.

7.5 Case Study 2: Horizontal Scaling and Distributed Systems in a Learning Factory

This case study evaluates the framework in a large-scale, distributed production environment by integrating diverse IT and OT systems within the LFGP (Lanza & Moser et al. 2015) at the wbk Institute of Production Science at the Karlsruhe Institute of Technology (KIT). In contrast to Case Study 1, which focused on vertical integration and plug-and-play capabilities at the station level, this case study demonstrates the framework's ability to horizontally scale across multiple independent systems, protocols, Middleware instances, and organizational boundaries. The Learning Factory represents a realistic brownfield setting with heterogeneous, partially legacy systems that must interoperate to support complex, value-adding processes such as real-time visualization, DPP, and simulation-based services. This section details how the Middleware instances were deployed, how integration patterns were selected, and how the resulting distributed architecture performed under real production conditions.

7.5.1 Scenario and Objective

To evaluate the framework in a complex, multi-system environment, a comprehensive case study was conducted within the LFGP. This setting represents a representative "brownfield" scenario, characterized by a heterogeneous landscape of both modern and legacy IT systems, each operating in a functional silo. An overview of the integration architecture is depicted in Figure 7.20.

The primary objective of this case study was to demonstrate the framework's horizontal scalability and its ability to orchestrate complex, value-added processes by integrating disparate systems. The evaluation focused on three key aspects:

- **Integration of Diverse Communication Paradigms:** Assessing the framework's capacity to seamlessly handle polling (pull), webhooks / SSE (push), RESTful APIs (request-response), and specialized protocols like those used by Eclipse Data Space Connectors (EDC).
- **Support for Distributed Middleware Deployments:** Validating the feasibility and benefits of a decentralized Middleware architecture, where multiple Middleware instances collaborate across network boundaries (e.g., on-premise to cloud).
- **Analysis of Integration Patterns:** Evaluating the trade-offs between virtualized and materialized integration patterns in achieving specific functional and non-functional requirements.

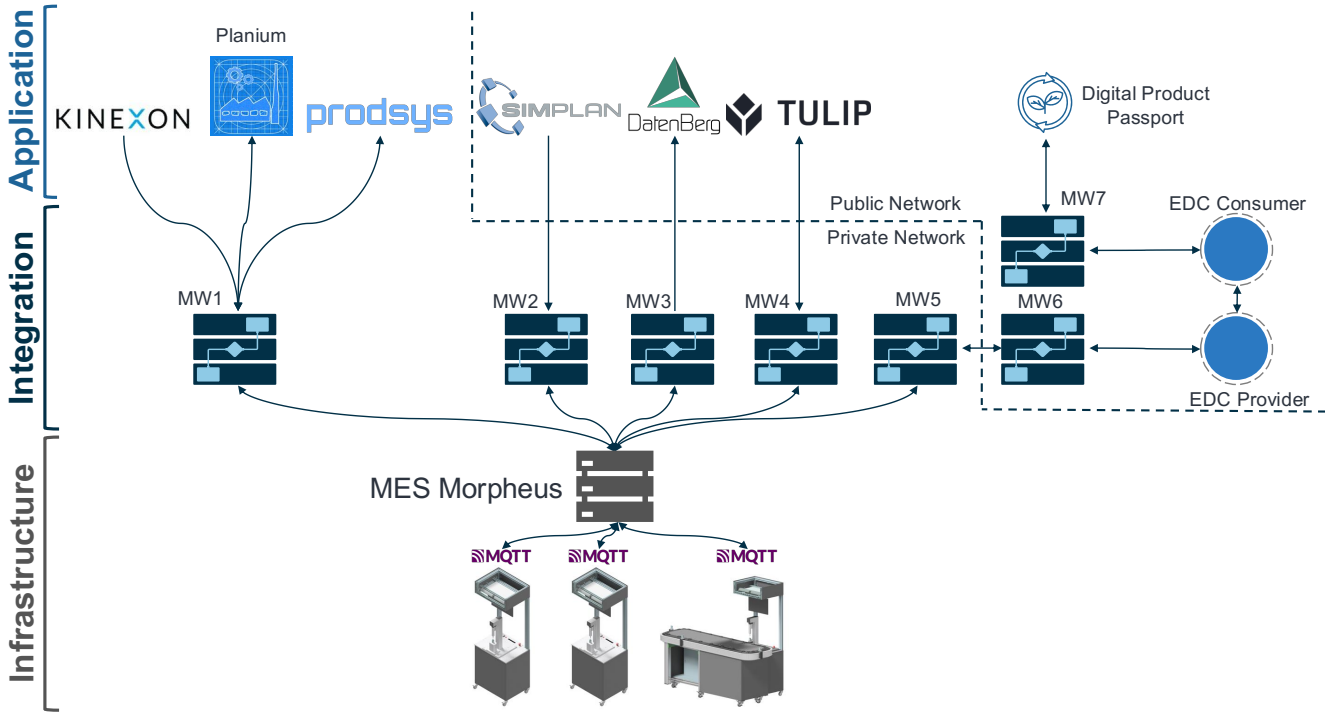

Figure 7.20: Conceptual overview of the LFGP showing connections managed by aas-middleware.

In the following, we want to elaborate how different legacy systems were integrated and why these integration patterns were selected.

7.5.2 Execution and Integration Scenarios

Several distinct integration sub-use-cases were implemented to target different aspects of the case study's objectives. Each Middleware instance was deployed as a containerized service.

MES & Datenberg: Virtualized, Push-Based KPI Dashboarding. The first scenario involved creating a real-time Key Performance Indicator (KPI) dashboard. The legacy MES was configured to emit webhook events upon the completion of production operations. A Middleware instance subscribed to these events, performing a virtualized, push-based integration. Upon receiving an event, the Middleware executed an in-memory schema mapping from the MES data model to the one expected by the Datenberg dashboard API and immediately forwarded the transformed data. This demonstrates a lightweight, stateless transformation pipeline ideal for event-driven monitoring.

MES & DPP: Decentralized, Hybrid Integration. The following scenario was originally conceptualized in Gleich & Behrendt et al. (2024) and was implemented in the master thesis by A_Zirbs (2025) under the supervision and technical guidance of the author. The

scenario aims to automatically generate and securely share a standardized DPP based on an AAS data. The DPP information is shared with an EDC to demonstrate state-of-the-art cross-company secure data sharing in data spaces (see Appendix A10 for more information on data spaces and the EDC). This integration required a complex, decentralized three-Middleware architecture to bridge network boundaries - on-premise to cloud - and enforce a clear separation of concerns:

MES-Side Middleware (On-Premise): this Middleware was located in the internal private network and listened for production events by subscribing to the MES with webhooks . Upon receiving such an event, it performed data enrichment by making synchronous API calls back to the MES to fetch additional order details. The consolidated data was then transformed into the open and standardized SDM Reference Model (Ellwein & Neumann et al. 2023) and pushed to the provider Middleware. Utilizing a standardized data model for DPP information is essential, as it ensures that the information transferred via the EDC remains interoperable across organizational boundaries, enabling secure and scalable cross-company access in public or federated data spaces.

Provider Middleware (Cloud): This Middleware resided in a publicly accessible cloud. It received the SDM-formatted data and orchestrated the secure data transfer using the EDC protocol, acting as the data provider.

Consumer Middleware (Cloud): Acting on behalf of the external partner, this Middleware received the data from the EDC, transformed it from the SDM format into the final AAS model, and materialized the integration by persisting the complete AAS data in a dedicated BaSyx server.

This hybrid approach combined virtualized data preparation with the materialization of the final digital asset.

MES & Tulip: Virtualized Human-in-the-Loop Process. To orchestrate a human-in-the-loop assembly process, the MES was integrated with an app in the Tulip cloud platform, which controlled a physical pick-by-light system. This was achieved through a simple, virtualized integration flow. An MES webhook, triggered by a change in order status, prompted a Middleware instance to map the event data to a Tulip API call. Upon receiving the data in the Tulip app, the pick-by-light signals were updated, thereby instructing the worker which parts to pick next. This demonstrates how decentralized control with standardized apps can be realized by using the Middleware's integration capabilities to make these apps available.

MES & SimController/prodsys: Request-Driven, Chained Mapping. This scenario involved providing production data to the material flow simulation tools SimController [1] and prodsys. The integration was request-driven and virtualized, initiated by a user request from the simulation front-end. This integration pattern was used to avoid reducing network traffic necessary in a materialized or push-based integration since user requests come infrequently and system data in the MES changes with a high frequency. The integration highlights the framework's ability to perform multi-step mappings, transforming data from the MES format to the internal SDM format, and then from the SDM format to the specific schema required by the SimController simulation engine.

MES & Planium & Kinexon: Multi-Source, Mixed-Pattern Communication. The following scenario was originally implemented in the master thesis by Neßler (2025) under the supervision and technical guidance of the author. The scenario involved creating a real-time DT of the shop floor in the Planium planning tool. This DT considered browser-based 3D visualization of shop floor activities or simulated activities using prodsys. This required integrating and fusing data from two distinct sources: the MES (for process status) and a Kinexon Real-Time Location System (for asset positions) and integrating them in Planium and prodsys. This integration utilized a mix of patterns:

- *Polling (Pull):* The Middleware actively polled the Kinexon API at regular intervals to fetch the latest location data, as the Kinexon hub does not support push-based events.
- *Materialized Caching:* The fetched location data was materialized in the Middleware's in-memory data model to serve as a high-performance cache for Planium.
- *Push (Webhook & SSE):* The Middleware simultaneously listened for MES webhooks to receive process status updates. It then combined the cached location data with the live process events and streamed the fused data to the Planium front-end using SSE, enabling a smooth, real-time 3D visualization.
- *Pull (Rest):* Requests from the Planium app for production system configuration, existing event logs or the interaction with prodsys were request-based since these interactions happened with low frequency but considered large amounts of data.

7.5.3 Results of the Evaluation

To quantitatively assess the performance and behavior of the integrated systems, observations were made over 4 days containing a 30-minute production scenario. The following analysis

[1] SimController is a commercial, high-fidelity simulation platform developed by SimPlan AG based on Siemens Tecnomatix Plant

details the interaction volumes, temporal patterns, latencies, and message sizes for each involved component, in order to analyze robustness and scalability of the PPC system in this horizontal integration approach. The results were obtained in a 4 day monitoring period containing both standby and active production.

The total number of interactions (both sent and received) for each service, shown in Figure 7.21, reveals a significant disparity in communication load. The components related to the real-time DT (Kinexon (the source), Planium Middleware (the aggregator), and Planium (the consumer)) dominate the interaction count, each handling over 10,000 messages. This is a direct result of the high-frequency polling of the Kinexon system and the continuous streaming of fused data to the Planium front-end. In contrast, services like the MES and the EDC-related Middleware show a moderate load (thousands of interactions), reflecting event-driven triggers from production and the multi-step protocol of the EDC. Components triggered by less frequent user actions, like Tulip, prodsys, or SimController, exhibit the lowest interaction counts.

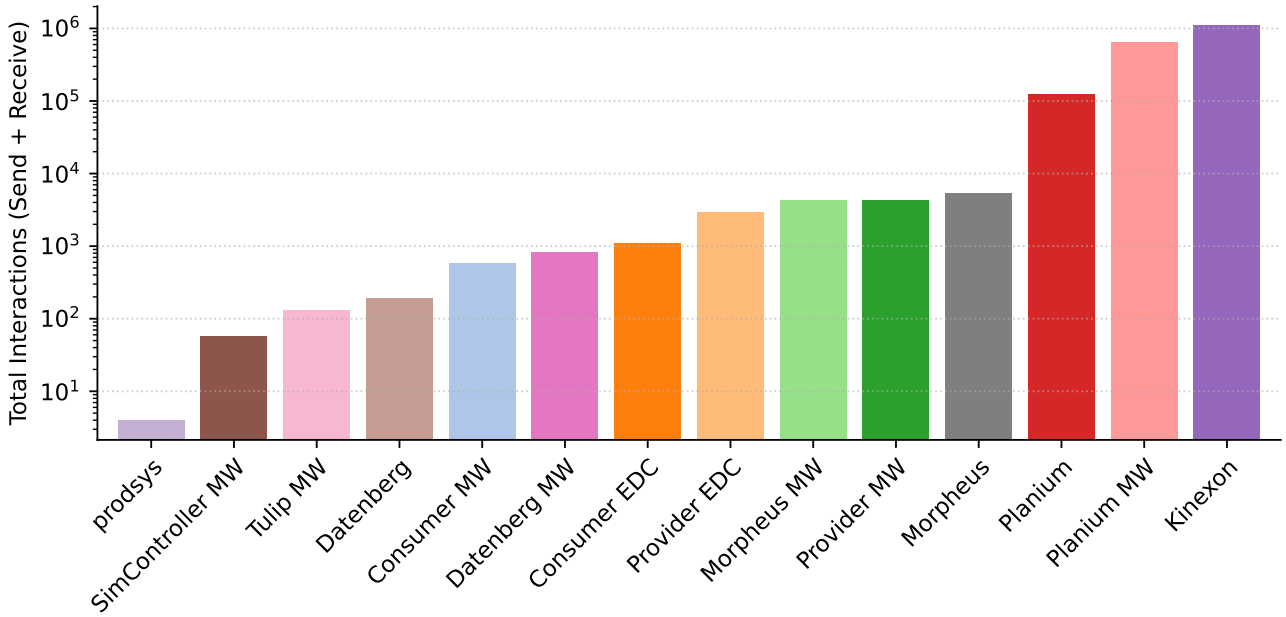

Figure 7.21: Total interactions per service during the observation period.

The timeline plots in Figure 7.22 and Figure 7.23 provide a temporal view of these interactions. The zoomed-in view (Figure 7.23), in particular, clearly visualizes the different communication paradigms employed. Kinexon's periodic polling appears as regularly spaced vertical dashes, representing the pull-based mechanism. In contrast, the event-driven webhook notifications from the MES and the resulting high-throughput streaming from the Planium Middleware show up as dense, sometimes bursty, clusters of activity. The EDC-related Middleware instances

also exhibit bursty behavior, corresponding to the negotiation and transfer of individual product passports.

A detailed analysis of communication latency and message sizes for all interactions in Case Study 2 is provided in Appendix A11. Here, the main findings are briefly summarized.

Lightweight, in-memory transformations (e.g., Consumer Middleware mapping to AAS) show very low latency, typically in the microsecond range. Interactions involving protocol negotiation or external communication-such as the Provider and Consumer EDC-operate in the tens of milliseconds. The highest latencies (seconds) occur for prodsys and SimController calls, where synchronous requests trigger computationally intensive simulations.

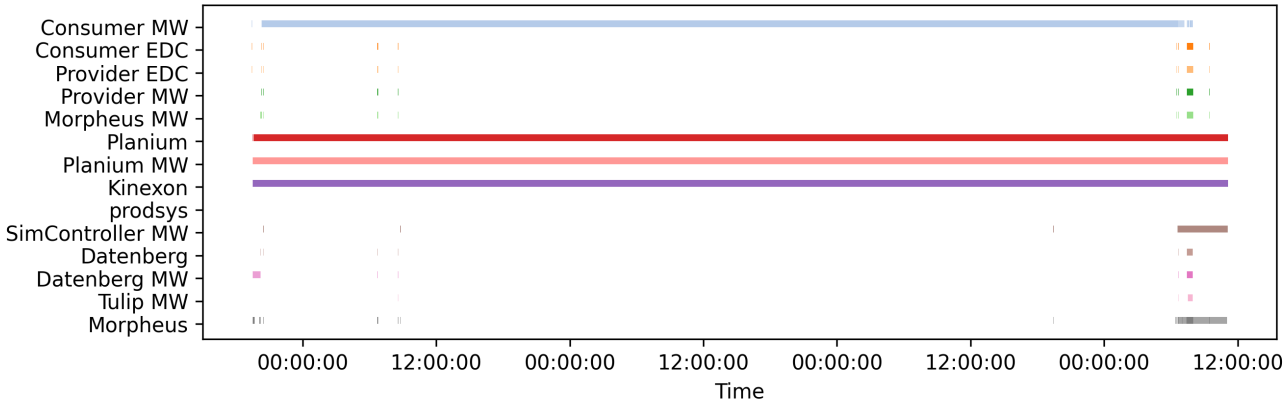

Figure 7.22: Timeline of all interactions over the full duration of the test.

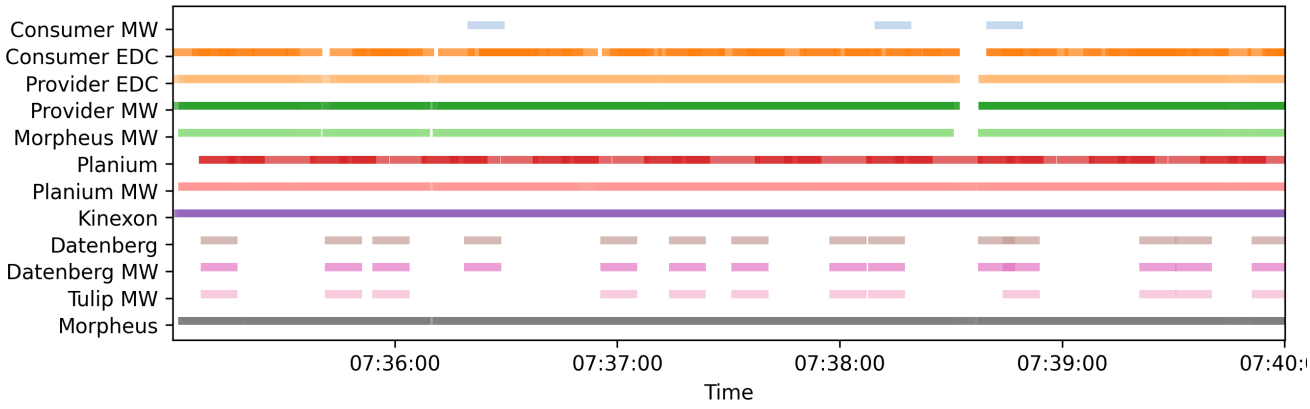

Figure 7.23: Zoomed-in timeline over 5 minutes showing distinct communication paradigms (e.g., polling vs. event bursts) during running production.

Most interactions (e.g., Tulip triggers or EDC messages) exchange very small payloads of only a few bytes. In contrast, simulation-related services (prodsys, SimController) and data-rich sources such as Kinexon transmit large messages (kB to MB), reflecting model data, layout

files, or sensor datasets. Here, avoidance of high frequency interactions is, if possible, key to efficiently realizing integrations.

7.5.4 Interim Conclusion

The integration scenarios of this case study provided clear evidence of the framework's performance characteristics, scalability, and the practical implications of selecting different integration strategies. Overall, the results reveal where bottlenecks may arise, how communication paradigms should be chosen, and which architectural decisions best support distributed PPC applications.

The performance evaluation shows that most integrations operate with very low latency. However, the distributed DPP scenario exposed potential bottlenecks caused by the EDC's negotiation protocol, which introduces noticeable delays. For production use, such interactions should therefore be decoupled through asynchronous, message-queue-based communication. Similarly, the dependence on an AAS server in the DPP Workflow highlights that high-frequency read operations require an additional caching layer to avoid overload. The in-memory caching approach demonstrated in the Planium scenario proved highly effective in mitigating this issue.

The case study also clarified the trade-offs between push- and pull-based communication. Push mechanisms (webhooks, SSE) offered clear advantages for real-time, high-frequency updates and are therefore the preferred option for EDAs, as demonstrated in the Datenberg and Planium integrations. Pull-based approaches, by contrast, remain necessary when interacting with legacy systems that cannot emit events, such as Kinexon, or when handling low-frequency, user-initiated requests, such as simulation triggers in the SimController Workflow.

With respect to integration patterns, the results illustrate when materialized or virtualized approaches should be applied. Materialized integration is essential when caching is required to bridge differing performance characteristics between systems, when data from asynchronous sources must be fused-as in the DT scenario, or when creating persistent, standalone data assets such as the DPP. Virtualized integration, in contrast, is most suitable for lightweight, event-driven pipelines in which the middleware acts as a stateless transformer, as seen in the Datenberg dashboard or Tulip pick-by-light integration. This avoids unnecessary resource consumption and reduces the need for maintaining data consistency across different storage layers. More complex, multi-system scenarios, such as in the MES–Planium–Kinexon integration, benefited from combining multiple patterns to meet heterogeneous performance and reliability requirements.

Finally, the DPP use case clearly demonstrated the architectural advantages of a decentralized middleware landscape. Separating data acquisition, secure transfer, and final persistence across distinct middleware instances simplified operations, enabled independent deployment cycles, and facilitated integration across security domains and organizational boundaries. This validates the suitability of a distributed middleware architecture for scalable and modular PPC applications operating in heterogeneous environments.

7.6 Case Study 3: Automated Production Planning and Control in an Industrial Scenario

The final case study aims to validate the entire framework's functionality, scalability, and economic impact within a complex, industrial-scale scenario. The case study observes a fully automated, closed-loop PPC of an RMS in a realistic setting that mirrors the challenges faced by manufacturers today. This evaluation (Behrendt & Enke et al. 2026) was conducted in collaboration with an automotive supplier, using the reconfigurable production system at the ARENA2036 research factory as a physical testbed (Behrendt & Ungen et al. 2023).

7.6.1 Scenario and Implementation

The production system used in this case study is a modular, highly reconfigurable RMS, as visualized in Figure 7.24. It consists of convertible assembly stations that can host up to two process modules each, a logistics system based on Automated Guided Vehicles (AGVs), and a standardized OT Gateway enabling software-defined control of all shop-floor assets. This setup allows the physical layout, resource allocation, and logistics infrastructure to be freely rearranged and makes the system an ideal testbed for studying the benefits of automated PPC in reconfigurable manufacturing environments.

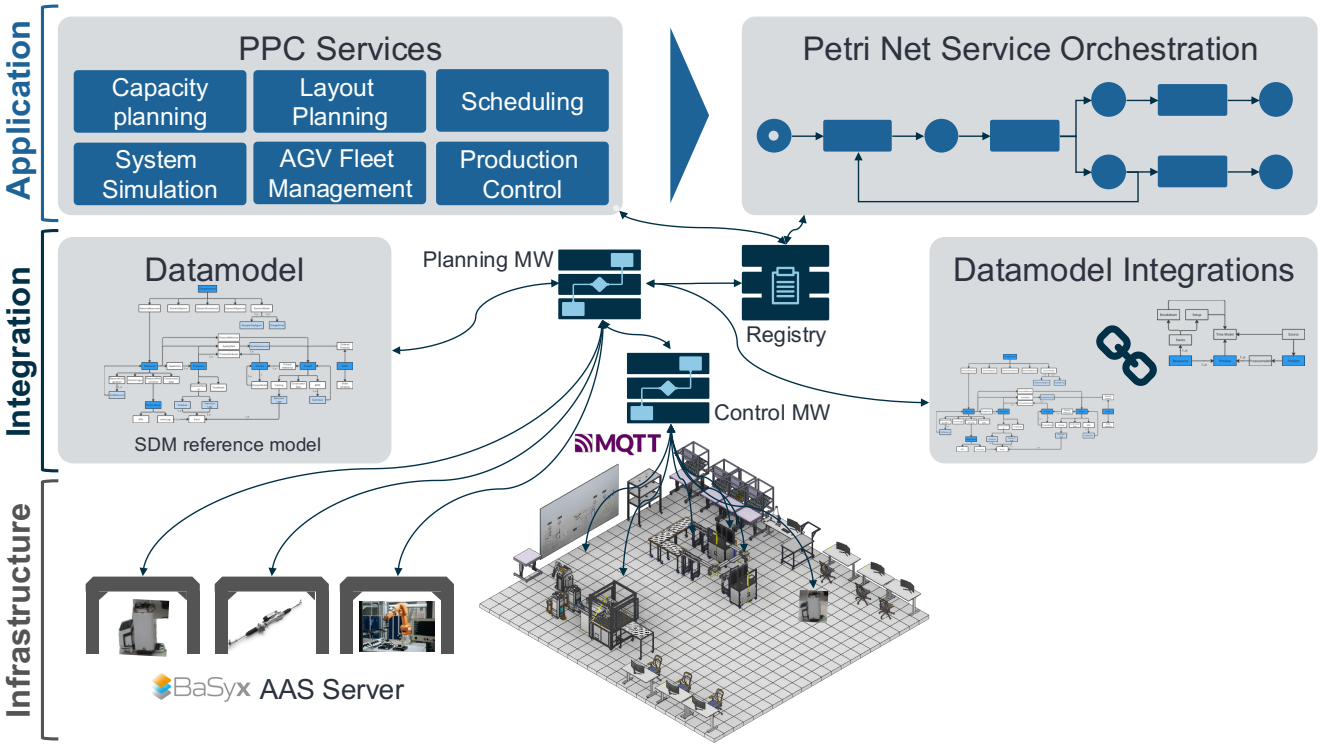

Figure 7.24: Overview of the production system of this case study and the architecture of the associated PPC system.

Two use cases with different levels of complexity were evaluated. The *demo use case* covers three consecutive assembly steps of the product and represents a compact scenario that is primarily used to validate the feasibility of the closed-loop PPC process model. In contrast, the *extended use case* models the complete 20-step assembly process and corresponds to a full industrial-scale planning and control problem.

Both cases consider the same 48-month forecast of twelve product variants with demand fluctuations of approximately a factor of six, as visualized in Figure 7.25. While the demo scenario is suited for rapid experimentation, the extended scenario stresses the scalability of the OE, planning services, and IT architecture under realistic long-term load conditions.

The primary goal was to automate the entire PPC process, from long-term capacity planning based on quarterly forecasts to short-term, reactive control in response to shop-floor disruptions.

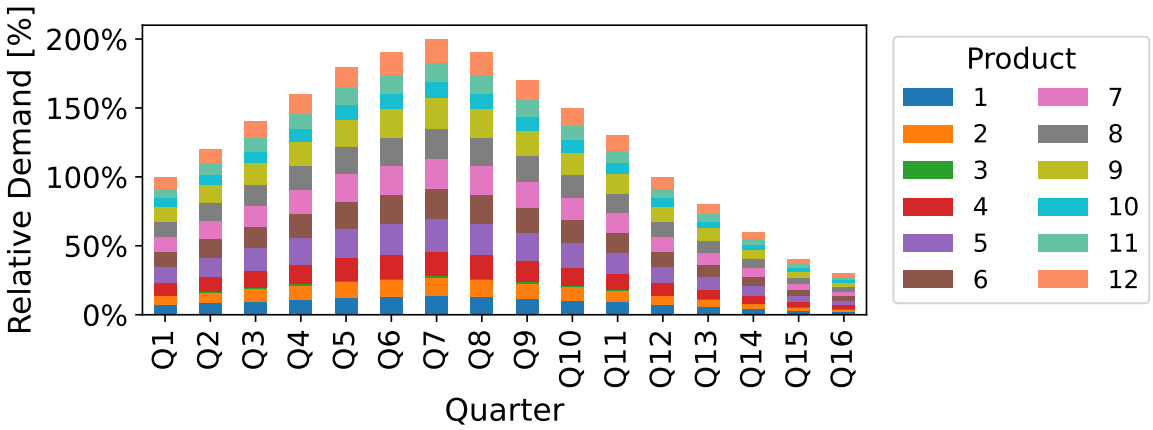

Figure 7.25: Scenario product mix of the 12 product variants in each quarter over a 48-month lifecycle, showing the high volatility in relative demand.

To achieve this, a closed-loop control architecture was implemented using the SAPN formalism, as described in Section 5.4. The orchestration logic defines two primary, concurrent process models - planning and control - executed by the OE. For each process model, a dedicated Middleware (denoted as MW in the visualizations) instance is used to accommodate fundamentally different requirements regarding data volume, computational expense, and latency.

This separation is essential, as strategic planning tasks involve high-volume data processing and long execution times, while operational control requires low-latency, high-availability interaction with shop-floor assets. Executing both concerns within a single process model or middleware instance would risk blocking real-time reactions during computationally intensive planning phases.

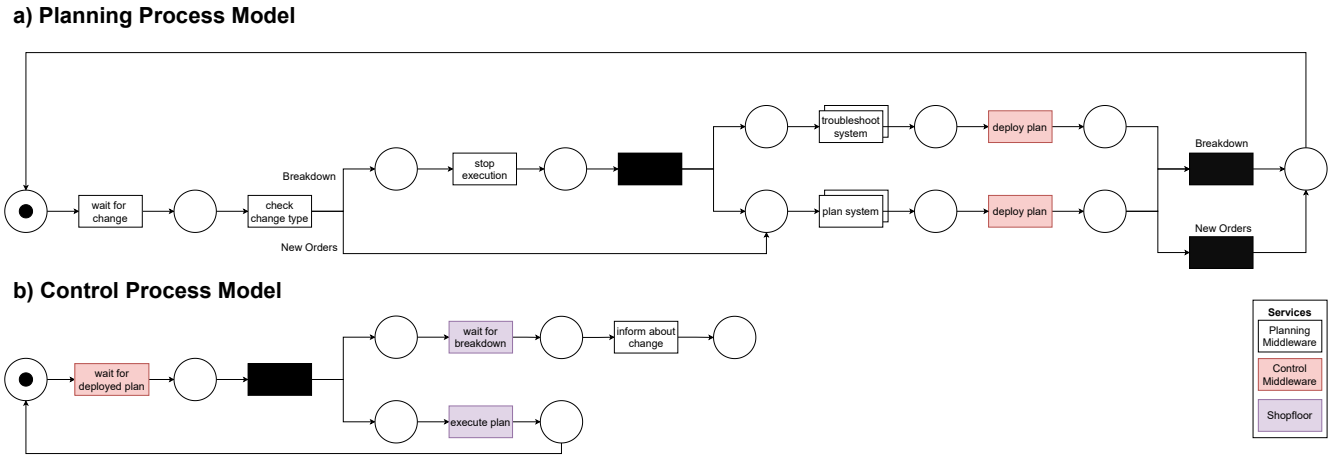

Figure 7.26: Visualization of the high-level planning (a) and control (b) process models that orchestrate the system's response to both long-term forecasts and real-time events.

Planning Process Model. A comprehensive planning cycle (see Figure 7.26a) controls the planning of the system. It is triggered at the start of each quarter to adapt the production system to new demand forecasts. An encapsulated process model, *Plan System Process Model* (Figure 7.27a), is used that involves all planning and validation services to generate an optimized production plan. This comprises: capacity and layout planning, scheduling, AGV route planning and simulation. It has the authority to reconfigure the system by adding, removing, or relocating production resources.

Beyond the quarterly planning cycle, the Planning process model also encapsulates the system's response to unforeseen breakdowns in the *Troubleshoot System Process Model* (Figure 7.27b). This process model is triggered by real-time events during execution and ensures that every disruption is resolved efficiently. To do so, faster planning services and parameters are utilized. The planning process model shows how SAPN can combine both strategic and tactical planning in a unified PPC orchestration model.

Control Process Model. A continuous, high-frequency loop (see Figure 7.26b) is responsible for executing the deployed plan and reacting to real-time events. In case of a disruption (e.g., a machine breakdown) or if all scheduled events have been completed, it informs the planning Middleware.

The closed-loop PPC integrates four classes of planning and validation services:

- capacity and layout planning with *prodsys* (meta-heuristics + simulation)
- scheduling with the commercial *flexis Scheduler* and a CP-SAT fallback

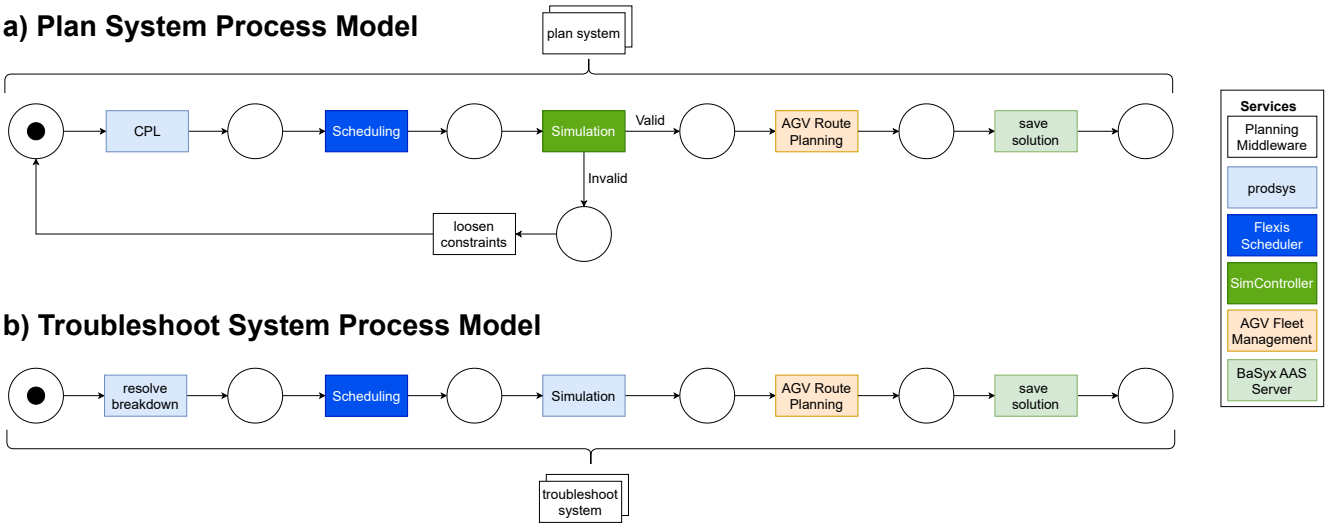

Figure 7.27: Detailed visualization of the service-level process models for (a) the comprehensive quarterly Plan System process model and (b) the rapid, event-triggered Troubleshoot System process model.

- discrete-event simulation using either high-fidelity *SimController* or faster *prodsys*
- AGV route planning and virtual commissioning based on LIF (Verband Deutscher Maschinen- und Anlagenbau e. V. (VDMA) 2024)

An overview of planning results - layouts, schedules, KPIs, and AGV trajectories - is shown in Figure A12.1. Further details are provided in Appendix A12.

The technical performance of this implementation was evaluated in a virtual mock-up environment that simulated the shop floor's OT gateway and connected assets, allowing for precise measurement of latencies, data volumes, and service interaction frequencies over the entire lifecycle.

Subsequently, the economic performance was assessed by simulating the 48-month scenario and comparing the DCF of the reconfigurable system against static benchmark systems designed for medium and high capacity.

For the economic evaluation, two classes of production systems were compared: a *reconfigurable system* controlled by the closed-loop PPC architecture, and several *static benchmark systems* that represent current industrial practice. The benchmark systems use the same available resource types as the reconfigurable system but remain fixed throughout the entire 48-month period, i.e., no machines are added, removed, or relocated over time. Following industrial conventions, two benchmark capacities were defined: a *mid-capacity* system sized for 75% of the maximum peak load (observed in Q7), and a *high-capacity* line sized for 100%

of the peak load. Each benchmark was evaluated under two optimization strategies, maximizing either throughput (TP) or a weighted fitness function (Fit), identical to the configuration used in the planning services.

In contrast, the *reconfigurable system* dynamically adapts to quarterly forecasts by adding or removing process modules, rearranging the layout, and modifying AGV routing infrastructure. This flexibility enables the system to reduce capital expenditure in low-demand phases and to increase throughput during peak periods, which-together with automated closed-loop PPC-forms the basis of the economic advantage observed in the case study.

The economic evaluation follows the methodology of Wiendahl & ElMaraghy et al. (2007) and applies a DCF analysis (Newnan & Lavelle et al. 2019) over the 48-month production horizon. All revenues originate from internally priced product output, while costs consist of operational expenditures (material and administrative costs) and capital expenditures for acquiring new resources. If the system divests machines during low-demand periods, a 50% resale value of the original asset cost is assumed, following conservative industry estimates validated with domain experts. All cash flows are discounted at 10% to ensure comparability of alternative planning and reconfiguration strategies. These assumptions are identical for the reconfigurable and static benchmark systems, ensuring a consistent evaluation basis.

The objective of this case study is threefold. First, it aims to demonstrate the *functional correctness* of the proposed closed-loop PPC system in a realistic, industrial-scale RMS. Second, it evaluates the *scalability and robustness* of the task-agnostic OE, the underlying IT architecture, and the service-oriented PPC framework by comparing the demo and extended use cases. Third, it quantifies the *economic impact* of reconfigurable manufacturing under automated PPC through a long-term financial analysis against realistic static benchmark systems. Together, these objectives establish both the technical and economic feasibility of the software-defined PPC framework.

7.6.2 Results and Findings

The results of this comprehensive case study validate the framework's feasibility, scalability, and economic benefits in a real-world context.

Technical Performance and Scalability. The framework successfully managed the automated execution of the complex PPC process models over the entire simulated period. The communication logs (Figure 7.28 and Figure 7.29) confirm the concurrent and coordinated execution of all integrated services without manual intervention for a fully automated PPC of an RMS, demonstrating the OE's robustness and the automation potential of service-oriented PPC.

The observations reveal the system's graceful scalability and the effectiveness of the dual process model approach. As detailed in Table 7.8, the median time for a full quarterly replan in the complex extended use case was approximately 14.5 hours. While significant, this is an acceptable timeframe for a detailed, strategic planning process that is fully automated. In stark contrast, the median time for troubleshooting a disruption was only 123 seconds. This sub-two-minute response time for tactical replanning is a critical achievement, demonstrating that the system can absorb heavy strategic planning workloads without compromising its ability to react swiftly to operational disruptions. This capability is a direct result of the task-agnostic architecture, which allows for the flexible deployment of services with different fidelity levels depending on the context.

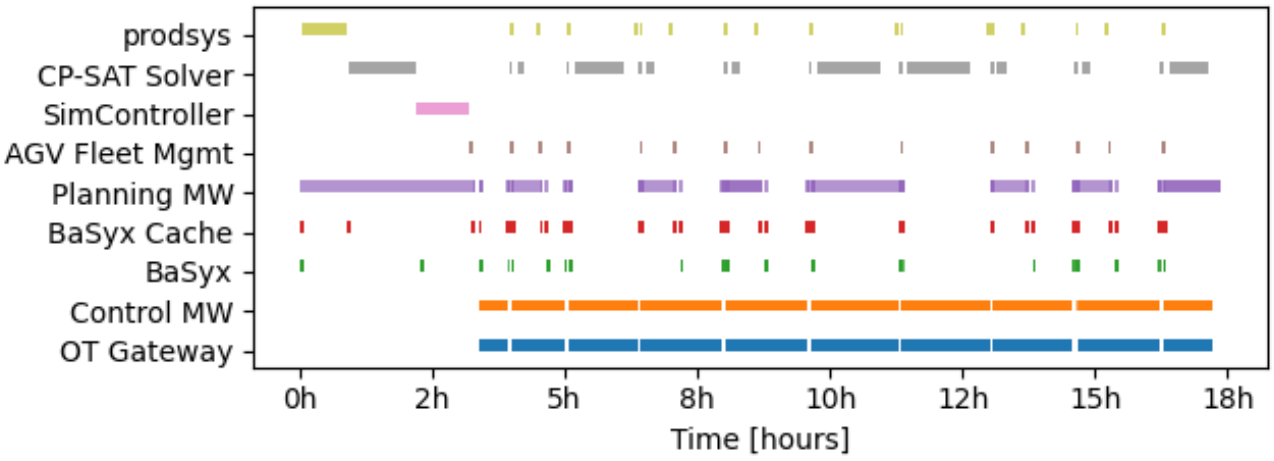

Figure 7.28: Communication activity of integrated services over 25 hours during the execution of the demo use case.

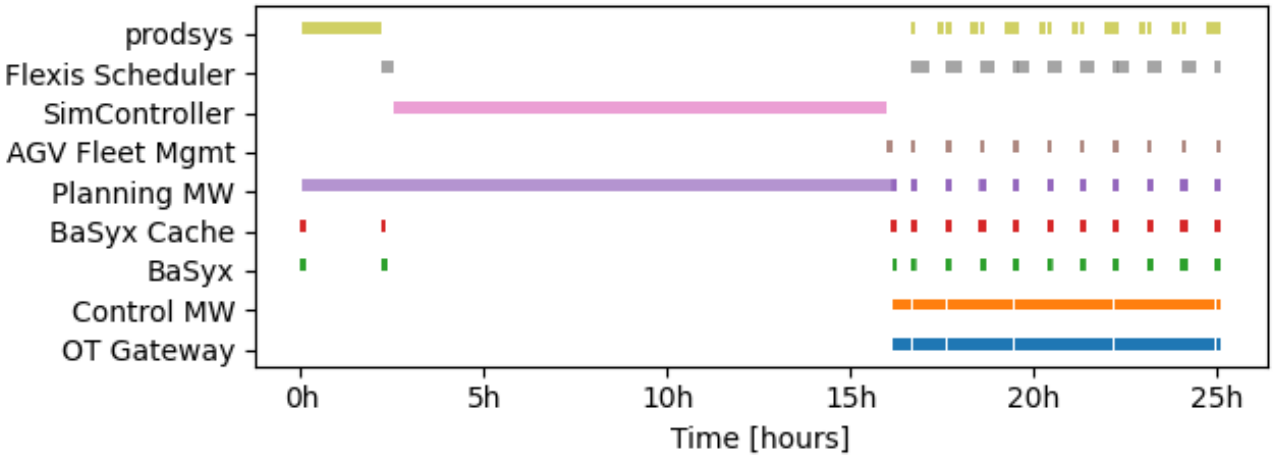

Figure 7.29: Communication activity of integrated services over 25 hours during the execution of the extended use case.

Table 7.8: Minimum, median, and maximum execution times for Planning vs. Troubleshooting System process model in both the demo and extended use cases.

Stage	Use Case	Min (s)	Median (s)	Max (s)
Planning	Demo	6 006.76	11 271.49	13 936.29
Planning	Extended	50 829.81	53 434.17	53 937.63
Troubleshooting	Demo	56.80	67.87	3 963.41
Troubleshooting	Extended	110.94	123.09	201.46

The analysis of service interactions confirmed the different characteristics of IT and OT communication seen before in the other case studies (see Appendix A12 for more details). Planning services (IT) involved high-latency, high-volume interactions, while shop-floor control services (OT) have low-latency, low-volume, high-frequency communication. This validated the architectural decision to decouple planning and control Middleware, as it allows each component to be optimized for its specific requirements. While challenges such as handling gigabyte-scale planning payloads and ensuring low latency for AAS access were encountered and addressed with caching and streaming, the overall technical evaluation was successful.

Economic Viability. The economic evaluation demonstrates that the adaptive capabilities enabled by the framework translate directly into tangible financial advantages. As shown in Figure 7.30 and Figure 7.31, the reconfigurable system, managed by the automated PPC logic, delivered a 10% higher DCF for both the demo and extended use case over the 48-month period when compared to the best-performing static benchmark system.

This superior performance stems from its ability to dynamically adapt its capacity to market demand. Static systems inevitably face a trade-off: the high-capacity benchmark suffered from excessive capital expenditure (CapEx) during periods of low demand, while the mid-capacity benchmark lost significant revenue by being unable to meet peak demand. The reconfigurable system navigated this volatility by divesting unneeded resources during downturns and acquiring new capacity to meet upturns, thereby optimizing both CapEx and revenue generation. Over the entire lifecycle, it achieved near-equivalent performance (99.94% of the highest benchmark throughput) while requiring substantially lower capital investment.

7.6.3 Interim Conclusion

This case study demonstrates that the proposed software-defined, closed-loop PPC framework operates reliably and efficiently in a complex, industrial-scale RMS. The dual-process model orchestration approach that combines quarterly strategic planning with rapid, event-driven troubleshooting proved essential for maintaining both long-term optimization quality and real-time responsiveness. The technical evaluation confirmed that the task-agnostic OE

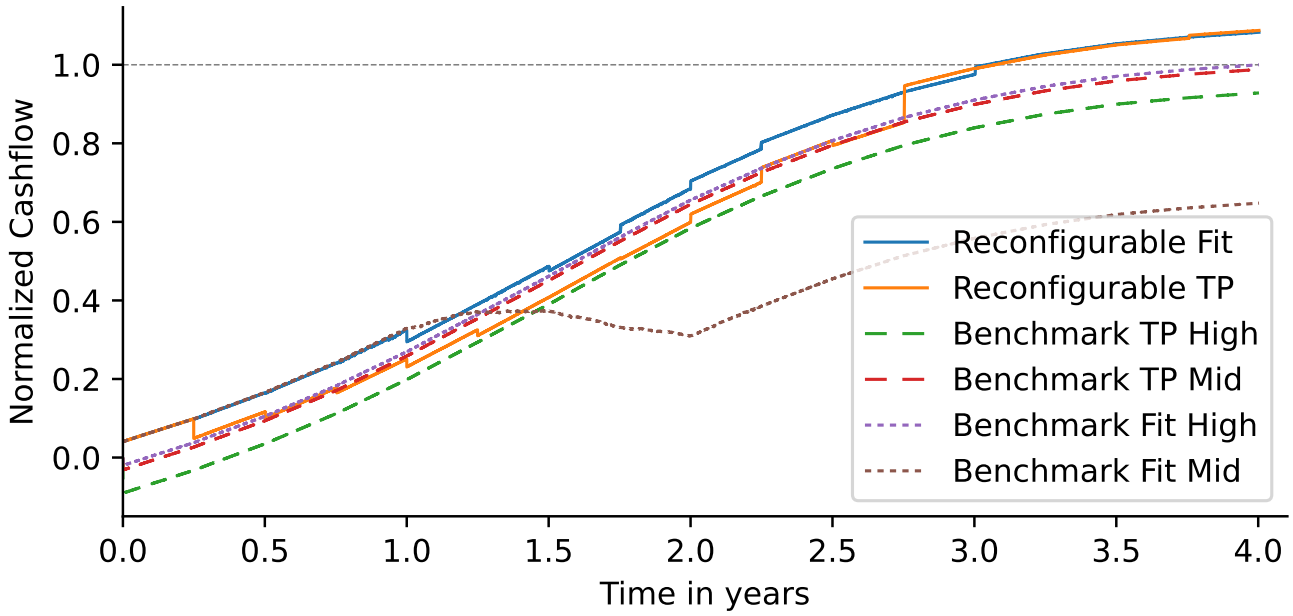

Figure 7.30: Normalized DCF over time for the demo use case, comparing the reconfigurable system against static mid- and high-capacity benchmarks.

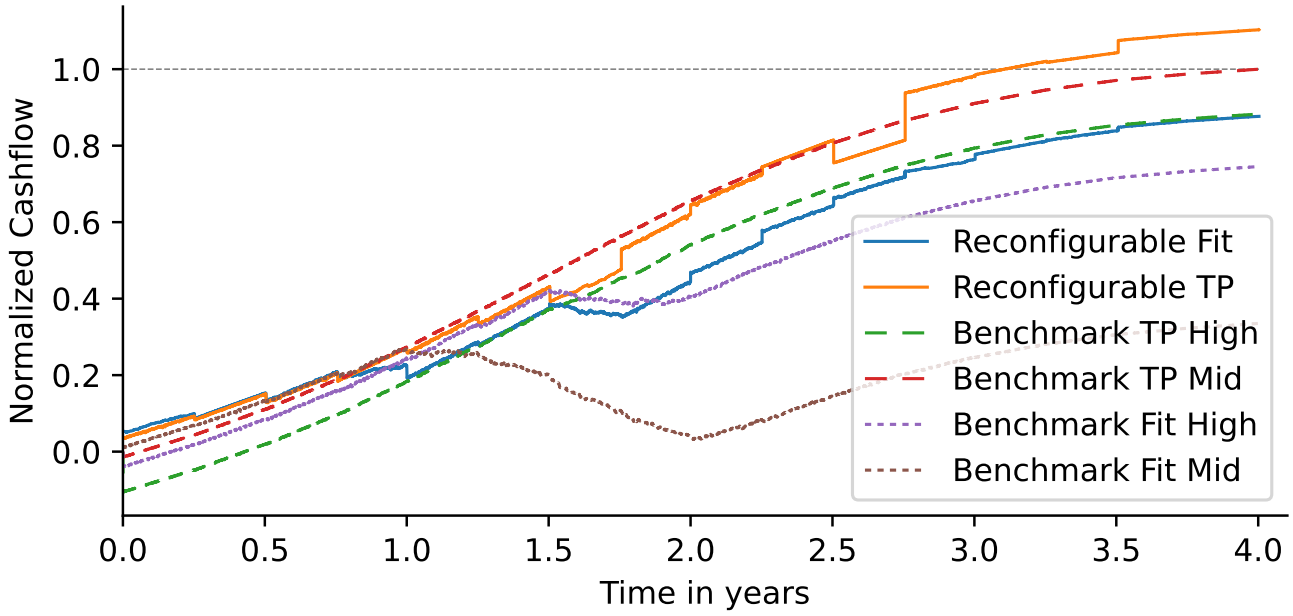

Figure 7.31: Normalized DCF over time for the extended use case, comparing the reconfigurable system against static mid- and high-capacity benchmarks.

scales gracefully, coordinating heterogeneous planning and validation services even under high load in the extended use case.

Economically, the reconfigurable system controlled by the automated PPC architecture outperformed static benchmark systems by dynamically adjusting its capacity to fluctuating demand.

This capability avoided the inefficiencies of both over- and under-provisioned static lines, resulting in a substantially higher DCF over the 48-month horizon.

Overall, Case Study 3 validates that the framework is technically robust, scalable, and economically advantageous, providing strong evidence of its suitability for real-world, service-oriented PPC in reconfigurable manufacturing under realistic industrial conditions. Taken together, the results demonstrate that service-oriented, task-agnostic PPC is not only architecturally elegant, but also operationally viable and economically justified at industrial scale.

8 Discussion and Outlook

This thesis aimed to resolve the architectural and functional dissonance between rigid, monolithic PPC systems and the dynamic requirements of RMS. By synthesizing SOA, SDM, LLM-driven semantic integration, and formal Petri net orchestration, the presented research provides a holistic framework for service-oriented PPC. This chapter discusses the core insights derived from the design and evaluation phases, explicitly answers the guiding research questions, critically analyzes limitations, and outlines future research trajectories.

8.1 Synthesis of Findings and Research Questions

The research followed a DSR methodology, focusing on the design, instantiation, and evaluation of a software artifact addressing architectural consistency, automated integration, and task-agnostic orchestration in PPC. The following subsections synthesize the empirical findings from the component benchmarks and the three system-level case studies to answer the three guiding research questions.

RQ1: A Holistic, Service-oriented Framework for Industrial Contexts. This research question is answered by the successful realization and validation of a registry-backed, containerized architecture that decouples the integration kernel, i.e. the Middleware, from the orchestration logic.

The design, rooted in SDM with the separation of concerns between Infrastructure, Integration, and Application layers, proved robust across diverse case studies. In the modular disassembly station scenario (Case Study 1), the framework demonstrated true plug-and-play capability: median OPC UA interaction latencies below 9 ms enabled tight synchronization at the OT boundary, while higher-latency IT services ran concurrently without blocking execution.

Scalability was confirmed in the distributed Learning Factory environment (Case Study 2), where the decentralized Middleware pattern effectively bridged on-premise, and cloud segments via EDC. The architecture accommodated mixed communication profiles, ranging from millisecond-scale event triggers to megabyte-sized simulation models, without structural changes. By implementing a "service-per-capability" design pattern supported by dynamic discovery, the framework delivers the modularity, location transparency, and loose coupling required to transition from static automation pyramids to flexible, software-defined ecosystems.

RQ2: Automating Semantic Integration. This research question is addressed by the implementation and evaluation of an integration method where LLMs automate the schema

matching, mapping, and data transformation process, bridging the gap between heterogeneous schemas.

The component evaluation demonstrated that this approach radically simplifies semantic integration. Across 41 heterogeneous mapping tasks - spanning standards covering layout data, simulation models, and legacy MES formats - the LLM-based service generated executable Python mapping functions that effectively solved complex semantic alignment challenges.

Quantitative analysis revealed that prompt engineering is decisive: while schema-only prompts yielded a modest mean F1-score of 0.42, enriched prompts containing descriptions, examples, and auxiliary rules achieved a mean F1-score of 0.89. Although performance naturally degraded with increasing schema complexity, the inclusion of specific context mitigated this effect, enabling the generation of complex mapping functions exceeding 300 lines of code. This reduces the role of the integration expert from writing low-level boilerplate code to high-level supervision and validation, significantly lowering the barrier for achieving semantic interoperability in brownfield environments. Moreover, the studies demonstrated the economic potential of this approach, with cost and time reductions of up to 80% for complex mapping cases in comparison to manual integration.

RQ3: Principles for Task-Agnostic Orchestration. Orchestration is targeted in this framework by the formalization and implementation of SAPN, which enable the dynamic composition of services based on abstract task requirements rather than hard-coded endpoints.

The component evaluation proved the performance and capabilities of the approach. At first, it was demonstrated how dynamic bindings to services, automatic fallback mechanisms, and automated data integration are utilized in the orchestration for reliable and scalable SAPN execution. With a benchmark study, it was shown that the SAPN-based OE provides superior performance in process model orchestration.

Moreover, the industrial case study (Case Study 3) validated that this orchestration core preserves agility even under heavy computational load in PPC environments. The system successfully managed a dual-process model strategy: a long-horizon strategic planning cycle (taking approximately 14.8 hours) and a rapid disruption troubleshooting cycle (median reaction time of 123 seconds).

The separation of the process definition (Petri net) from the service execution (late binding via Registry) allowed the system to switch between high-fidelity simulation services and heuristic solvers at runtime without process model modification. Economically, this reconfigurability translated into a 10.3% higher DCF compared to the best static benchmark system over a simulated 48-month lifecycle. This result directly ties the technical capability of task-agnostic

orchestration to tangible business value, proving that dynamic PPC architectures can optimize capital expenditure and throughput in volatile markets.

8.2 Contextualization and Practical Implications

The proposed framework advances the state of the art by moving beyond the protocol unification focus of early I4.0 efforts. While prior works often stopped at establishing static interfaces (e.g., wrapping a PLC in OPC UA), this research demonstrates a functional upper layer that handles semantic alignment and process logic dynamically.

Hybrid Integration Strategy. A key insight from the distributed system evaluation (Case Study 2) is the necessity of a hybrid integration strategy. Purely virtualized (stateless) integration proved effective for event-driven flows like real-time dashboards, minimizing latency and storage overhead. Conversely, materialized integration (caching data in the Middleware's in-memory model backed by AAS) was essential for heavy, pull-based, or multi-source fusion scenarios to prevent readout bottlenecks on the persistence layer. This practical differentiation provides a blueprint for architects: there is no generally suitable integration pattern; rather, the Middleware must flexibly toggle between virtualized and materialized states based on the requirements of the use case.

Scalability and Generalizability. The scalability of the approach was demonstrated along three axes: service volume (Case Study 2), deployment topology (spanning security domains via EDC in Case Study 2), and planning complexity (Case Study 3). While the specific services used (e.g., *flexis*, *prodsys*) are domain-specific, the architectural patterns are transferable to other domains such as logistics or energy management. The strongest leverage for generalization exists where assets offer standardized self-descriptions (with AAS or similar formats), allowing the OE to bootstrap its internal models automatically.

Adoption Path. For practitioners, the findings suggest a phased adoption path. Companies should not attempt a complete replacement of legacy PPC systems. Instead, high-pain integration points should be addressed first using the Middleware and LLM mapping to establish immediate value. Subsequently, a Service Registry can be introduced to standardize interfaces, followed by the formal orchestrator once a critical mass of services exists. This lean, incremental approach aligns investment with visible return on investment.

8.3 Limitations and Mitigation Strategies

Despite the demonstrated success, several limitations emerged during the development and evaluation phases which must be addressed for production-grade deployment.

Reliability of Generative Models. A primary bottleneck remains the non-deterministic nature of LLMs. While F1-scores were high, the mapping accuracy is highly sensitive to prompt quality and the specific model used. In safety-critical contexts, hallucinations in generated code could lead to data corruption. To mitigate this, the framework implements a strict validation harness. Generated code executes within a sandboxed environment and is subjected to automated unit tests derived from example data. Furthermore, the methodology enforces a human-in-the-loop review step for the initial deployment of new mappings. Future iterations could employ ensemble generation (querying multiple models) to statistically surface the most robust mapping candidates.

Interoperability Performance Bottlenecks. While the Middleware minimizes latency, reliance on external standards can introduce overhead. Specifically, AAS servers can become read bottlenecks under high-frequency access if not protected by caching layers. Additionally, the negotiation protocols of EDC introduce non-trivial latency, making them unsuitable for synchronous real-time control loops. Mitigation strategies involve the strict application of the hybrid integration patterns identified in Section 7.5. High-frequency loops must utilize virtualized push/event streams (e.g., Webhooks/SSE), reserving AAS interactions for asynchronous lifecycle updates. For distributed setups, decoupling EDC exchanges via asynchronous message queues is recommended to hide negotiation latency from the operational process.

Governance and Complexity. The flexibility of a microservice architecture introduces governance risks. Without strict discipline, the landscape can degrade into an unmanageable mesh of ad-hoc integrations ("spaghetti architecture"). Furthermore, the introduction of formal verification and distributed deployments adds cognitive overhead for operators compared to monolithic systems. To counter this, the system relies on the Service Registry as the single source of truth. Governance must be enforced by mandating that all service interactions occur via the registry and standardized Middleware paths. A dedicated, graphical UI that visualizes the Petri net state was implemented to make the complexity manageable for process engineers.

8.4 Outlook and Future Research

The foundation established by this thesis opens several promising avenues for future research.

First, the orchestration logic should be extended to support self-optimizing control loops. Currently, the selection of services is based on availability and static attributes. Future work could integrate Reinforcement Learning agents that learn optimal service compositions over time based on process outcomes (e.g., selecting a faster but less accurate scheduler during

high-load periods). Moreover, LLM-based agents could serve as a natural language interface to planners to easily adjust PPC logic defined in process models.

Second, robustness at scale requires further investigation into fault-containment domains. While the current engine supports retries and fallback mechanisms, self-healing compositions that can automatically re-negotiate service contracts or spawn redundant instances in cloud environments would enhance resilience.

Third, the LLM integration pipeline can be refined for domain-specificity. Fine-tuning smaller, open-source models on specific industrial data standards (like AML or specific AAS sub-models) could reduce inference costs and improve privacy compared to general-purpose API-based models. Additionally, automated configuration mining - extracting service contracts directly from legacy logs and documentation - could further accelerate the bootstrapping of the Service Registry.

Finally, future work should explore the formal synthesis of recovery strategies. By leveraging the mathematical properties of Petri nets, it may be possible to mathematically guarantee reaction times in mixed IT/OT loops, providing the certification capabilities necessary for safety-critical production environments. The industrial scenario in this work suggests that combining long-horizon, high-fidelity planning with short-cycle, low-fidelity troubleshooting is operationally advantageous; generalizing this dual-fidelity principle to additional PPC stages represents a significant opportunity for efficiency gains.

9 Conclusions

This thesis demonstrated that a service-oriented, schema-aware PPC architecture can fundamentally improve the flexibility, interoperability, and economic performance of PPC systems in volatile reconfigurable manufacturing environments. By systematically decoupling integration, orchestration, and application logic, the presented framework resolves core limitations of monolithic PPC systems while remaining operationally viable in brownfield industrial settings.

From an architectural perspective, the proposed framework for service-oriented PPC consisting of Middleware, Service Registry, SIS, and OE proved robust and scalable across heterogeneous deployment scenarios. Quantitative evaluations showed that low-latency OT synchronization (median OPC UA interaction times below 9 ms) can coexist with computationally intensive IT services without blocking control execution. Moreover, the evaluations demonstrated that the Middleware's capabilities enable the AAS to serve as a persistent semantic data store for PPC. Case studies further confirmed that the architecture supports mixed communication paradigms and cross-domain deployments without structural changes, validating its suitability for SDM ecosystems.

A central contribution of this work is the automation of semantic integration through LLM-driven schema mapping. Component-level benchmarks across 41 heterogeneous integration tasks demonstrated that enriched, context-aware prompting enables the generation of executable mapping functions with a mean F1-score of 0.89. These results show that LLMs can reliably externalize expert knowledge into reusable integration logic, reducing integration effort in terms of time and cost by up to 80%. Qualitatively, this shifts the role of the integration engineer from low-level implementation toward validation and governance, significantly lowering the barrier to interoperability in evolving production landscapes.

Task-agnostic orchestration was realized through the formalization and execution of SAPN. Benchmark studies confirmed that the SAPN-based OE achieves high execution throughput while enabling dynamic binding, fallback mechanisms, and automated schema mediation. The industrial case study validated these capabilities under realistic load conditions, where the system successfully coordinated long-horizon strategic planning cycles (approximately 14.8 hours) alongside rapid disruption handling, with automated replanning achieving a median reaction time of 123 seconds. Crucially, the task-agnostic separation of PPC process models from service execution allowed runtime switching between high-fidelity simulation services and heuristic solvers without modifying the process model itself.

Beyond technical feasibility, the results demonstrate tangible economic impact. In a simulated 48-month lifecycle evaluation, the dynamically reconfigurable PPC system achieved a 10.3% higher discounted cash flow compared to the best static benchmark configuration. This directly links architectural flexibility and task-agnostic orchestration to improved capital efficiency and throughput robustness in volatile production environments with RMS.

In summary, this thesis contributes:

1. a formally grounded, service-oriented reference architecture for PPC,
2. an LLM-based semantic integration method with quantified accuracy and efficiency gains, and
3. an executable, task-agnostic orchestration model that preserves agility under industrial-scale load.

Taken together, these contributions advance PPC from static, tightly coupled systems toward adaptive, software-defined control loops. The presented framework provides both a theoretical foundation and a validated engineering blueprint for future PPC systems that must continuously evolve alongside products, processes, and organizational structures.

References

References according to the scheme (A_<last name> <year>) refer to Bachelor and Master Theses at the wbk Institute of Production Science, which were supervised by the author of this dissertation.

A_Bartenbach 2022
Bartenbach, J. (2022), „Konzeptionierung und Implementierung einer adaptiven Planungssoftware für rekonfigurierbare Produktionssysteme“. Masterarbeit, Karlsruher Institut für Technologie (KIT), wbk Institut für Produktionstechnik, Karlsruhe.

A_Bulach 2025
Bulach, S. (2025), „Development of a Modular Framework for Automating Schema-Mapping Processes in Production Planning an Supply Chain Management“. Masterarbeit, Karlsruher Institut für Technologie (KIT), wbk Institut für Produktionstechnik, Karlsruhe.

A_Neßler 2025
Neßler, J. (2025), „Realisierung dynamischer 3D-Visualisierungen für Digitale Zwillinge von Produktionssystemen“. Masterarbeit, Karlsruher Institut für Technologie (KIT), wbk Institut für Produktionstechnik, Karlsruhe.

A_Zirbs 2025
Zirbs, R. (2025), „Konzeptentwicklung und -integration für einen sicheren und souveränen Datenaustausch in Datenökosystemen für digitale Produktpässe unter Verwendung des Eclipse Dataspace Connector“. Masterarbeit, Karlsruher Institut für Technologie (KIT), wbk Institut für Produktionstechnik, Karlsruhe.

Abiteboul & Hull et al. 1996
Abiteboul, S.; Hull, R. & Vianu, V. (1996), *Foundations of Databases*. Reprinted with corr. Reading, Mass.: Addison-Wesley. 685 pp.

Acker 2011
Acker, I. J. (2011), *Methoden zur mehrstufigen Ablaufplanung in der Halbleiterindustrie*. Wiesbaden: Gabler. DOI: 10.1007/978-3-8349-6731-2.

Ajdinović & Strljic et al. 2024
Ajdinović, S.; Strljic, M.; Lechler, A. & Riedel, O. (2024), „Interoperable Digital Product Passports: An Event-Based Approach to Aggregate Production Data to Improve Sustainability and Transparency in the Manufacturing Industry“. *2024 IEEE/SICE International Symposium on System Integration (SII)*, pp. 729–734. DOI: 10.1109/SII58957.2024.10417487.

Alexopoulos & Kalogeras et al. 2025
Alexopoulos, A.; Kalogeras, G.; Koutras, K. & Kalogeras, A. (2025), „Why Asset Administration Shells: A Survey on Uses and Challenges“, *IEEE Access* 13, pp. 126582–126609. DOI: 10.1109/ACCESS.2025.3589931.

Alizadeh & Shahrezaei et al. 2019
Alizadeh, M.; Shahrezaei, M. H. & Tahernezhad-Javazm, F. (2019), *Ontology Based Information Integration: A Survey*. DOI: 10.48550/arXiv.1909.13762. URL: http://arxiv.org/abs/1909.13762 (visited on 10/16/2025).

Alonso & Casati et al. 2004
Alonso, G.; Casati, F.; Kuno, H. & Machiraju, V. (2004), *Web Services: Concepts, Architectures and Applications*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-662-10876-5.

Babel 2024
Babel, W. (2024), *Systemintegration in Industrie 4.0 und IoT: Vom Ethernet bis hin zum Internet und OPC UA*. Wiesbaden: Springer Vieweg. 606 pp. DOI: 10.1007/978-3-658-42987-4.

Baheti & Gill 2011
Baheti, R. & Gill, H. (2011), „Cyber-Physical Systems“, *The impact of control technology* 12.1, pp. 161–166.

Bastos & Sguario Coelho De Andrade et al. 2021
Bastos, A.; Sguario Coelho De Andrade, M. L.; Yoshino, R. T. & Santos, M. M. D. (2021), „Industry 4.0 Readiness Assessment Method Based on RAMI 4.0 Standards“, *IEEE Access* 9, pp. 119778–119799. DOI: 10.1109/ACCESS.2021.3105456.

Batini & Scannapieco 2006
Batini, C. & Scannapieco, M. (2006), *Data Quality: Concepts, Methodologies and Techniques*. Springer Berlin Heidelberg. DOI: 10.1007/3-540-33173-5.

Bauer 2017
Bauer, J. (2017), *Produktionscontrolling Und -Management Mit SAP® ERP*. IT-Professional. Wiesbaden: Springer Fachmedien. DOI: 10.1007/978-3-658-18366-0.

Bauernhansl 2023
Bauernhansl, T., ed. (2023), *Handbuch Industrie 4.0: Band 1: Produktion*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-662-58532-0.

Bauernhansl & Krüger et al. 2016
Bauernhansl, T.; Krüger, J.; Reinhart, G. & Schuh, G. (2016), „WGP-Standpunkt Industrie 4.0“, DOI: 10.24406/publica-fhg-297876.

Behrendt & Ungen et al. 2023
Behrendt, S.; Ungen, M.; Fisel, J.; Hung, K.-C.; May, M.-C.; Leberle, U. & Lanza, G. (2023), „Improving Production System Flexibility and Changeability Through Software-Defined Manufacturing“. *Production at the Leading Edge of Technology*. Ed. by M. Liewald; A. Verl; T. Bauernhansl & H.-C. Möhring. Lecture Notes in Production Engineering. Cham: Springer International Publishing, pp. 705–716. DOI: 10.1007/978-3-031-18318-8_70.

Behrendt & Altenmüller et al. 2025

Behrendt, S.; Altenmüller, T.; May, M. C.; Kuhnle, A. & Lanza, G. (2025), „Real-to-Sim: Automatic Simulation Model Generation for a Digital Twin in Semiconductor Manufacturing“, *Journal of Intelligent Manufacturing*. DOI: 10.1007/s10845-025-02572-x.

Behrendt & Bail et al. 2025

Behrendt, S.; Bail, F.; Benfer, M. & Lanza, G. (2025), „Aas-Middleware: Enabling Interoperable, Real-Time Integration for Smart Manufacturing“. *Proceedings of the 5th Modeling, Estimation and Control Conference (MECC 2025)*. Vol. 59. 30. Pittsburgh, Pennsylvania, USA: IFAC-PapersOnLine, pp. 671–676. DOI: 10.1016/j.ifacol.2025.12.315.

Behrendt & Danz et al. 2025

Behrendt, S.; Danz, J.; Kirner, N.; Rudzinski-Rudno, J. S. von; Holtzwart, T.; Zhao, K. & Offermanns, T. (June 2025), *sdm4fzi/prodsys: Version 0.9.10*. Version 0.9.10. DOI: 10.5281/zenodo.15709909.

Behrendt & Enke et al. 2026

Behrendt, S.; Enke, C.; Rüdt, M.; Bussmann, M.; Tutsch, H.; Spieckermann, S. & Lanza, G. (2026), „Closing the Loop: A Software-Defined Approach to Agile Production Planning and Control“, *Journal of Manufacturing Systems* 86, pp. 1007–1021. DOI: 10.1016/j.jmsy.2026.03.025.

Behrendt & Stamer et al. 2023

Behrendt, S.; Stamer, F.; May, M. C. & Lanza, G. (2023), „Towards a Service-Oriented Architecture for Production Planning and Control: A Comprehensive Review and Novel Approach“. *Proceedings of the 5th Conference on Production Systems and Logistics (CPSL 2023)*. Ed. by D. Herberger. Stellenbosch, South Africa: publish-Ing., pp. 255–267. DOI: 10.15488/15271.

Behrendt & Wurster et al. 2023

Behrendt, S.; Wurster, M.; Strljic, M.; Klein, J.-F.; May, M. C. & Lanza, G. (2023), „Interoperable Architecture for Logical Reconfigurations of Modular Production Systems“. *Proceedings of the 5th Conference on Production Systems and Logistics (CPSL 2023)*. Ed. by D. Herberger. Stellenbosch, South Africa: publish-Ing., p. 622. DOI: 10.15488/15270.

Bernstein & Haas 2008

Bernstein, P. & Haas, L. (2008), „Information Integration in the Enterprise“, *Commun. ACM* 51, pp. 72–79. DOI: 10.1145/1378727.1378745.

Bianco & Kotermanski et al. 2007

Bianco, P.; Kotermanski, R. & Merson, P. (2007), *Evaluating a Service-Oriented Architecture*. CMU/SEI-2007-TR-015. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Biesinger & Meike et al. 2019
Biesinger, F.; Meike, D.; Kraß, B. & Weyrich, M. (2019), „A Digital Twin for Production Planning Based on Cyber-Physical Systems: A Case Study for a Cyber-Physical System-Based Creation of a Digital Twin“, *Procedia CIRP*. 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering 79, pp. 355–360. DOI: 10.1016/j.procir.2019.02.087.

Bortolini & Galizia et al. 2018
Bortolini, M.; Galizia, F. G. & Mora, C. (2018), „Reconfigurable Manufacturing Systems: Literature Review and Research Trend“, *Journal of Manufacturing Systems* 49, pp. 93–106. DOI: 10.1016/j.jmsy.2018.09.005.

Boyes & Hallaq et al. 2018
Boyes, H.; Hallaq, B.; Cunningham, J. & Watson, T. (2018), „The Industrial Internet of Things (IIoT): An Analysis Framework“, *Computers in Industry* 101, pp. 1–12. DOI: 10.1016/j.compind.2018.04.015.

Bozkurt & Schulz 2023
Bozkurt, A. & Schulz, R. (Oct. 2023), „Asset Administration Shell and Knowledge Graph-based Production Logistics Planning in Fluid Manufacturing Systems“. *Logistics Journal*. Vol. 2023. DOI: 10.2195/lj_proc_bozkurt_en_202310_01.

Breje & Gyorödi et al. 2018
Breje, A.-R.; Gyorödi, R.; Gyorödi, C.; Zmaranda, D. & Pecherle, G. (2018), „Comparative Study of Data Sending Methods for XML and JSON Models“, *International Journal of Advanced Computer Science and Applications* 9.12. DOI: 10.14569/IJACSA.2018.091229.

Brovkina & Riedel 2021
Brovkina, D. & Riedel, O. (2021), „Assembly Process Model for Automated Assembly Line Design“. *2021 IEEE 3rd Eurasia Conference on IOT, Communication and Engineering (ECICE)*, pp. 588–594. DOI: 10.1109/ECICE52819.2021.9645604.

Brown & Mann et al. 2020
Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I. & Amodei, D. (2020), *Language Models Are Few-Shot Learners*. DOI: 10.48550/arXiv.2005.14165. URL: http://arxiv.org/abs/2005.14165 (visited on 11/06/2025).

Bueno & Godinho Filho et al. 2020
Bueno, A.; Godinho Filho, M. & Frank, A. G. (2020), „Smart Production Planning and Control in the Industry 4.0 Context: A Systematic Literature Review“, *Computers & Industrial Engineering* 149, p. 106774. DOI: 10.1016/j.cie.2020.106774.

Buss & Safari et al. 2025

Buss, C.; Safari, M.; Termehchy, A.; Lee, S. & Maier, D. (2025), *Towards Scalable Schema Mapping Using Large Language Models*. DOI: 10.48550/arXiv.2505.24716. URL: http://arxiv.org/abs/2505.24716 (visited on 07/15/2025).

Cánovas Izquierdo & Cabot 2013

Cánovas Izquierdo, J. L. & Cabot, J. (2013), „Discovering Implicit Schemas in JSON Data“. *Web Engineering*. Ed. by F. Daniel; P. Dolog & Q. Li. Berlin, Heidelberg: Springer, pp. 68–83. DOI: 10.1007/978-3-642-39200-9_8.

Cassano & Gouwar et al. 2024

Cassano, F.; Gouwar, J.; Lucchetti, F.; Schlesinger, C.; Freeman, A.; Anderson, C. J.; Feldman, M. Q.; Greenberg, M.; Jangda, A. & Guha, A. (2024), *Knowledge Transfer from High-Resource to Low-Resource Programming Languages for Code LLMs*. DOI: 10.48550/arXiv.2308.09895. URL: http://arxiv.org/abs/2308.09895 (visited on 11/06/2025).

Christen 2012

Christen, P. (2012), *Data Matching. Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-642-31164-2.

Cimini & Lagorio et al. 2022

Cimini, C.; Lagorio, A.; Cavalieri, S.; Riedel, O.; Pereira, C. E. & Wang, J. (2022), „Human-Technology Integration in Smart Manufacturing and Logistics: Current Trends and Future Research Directions“, *Computers & Industrial Engineering* 169, p. 108261. DOI: 10.1016/j.cie.2022.108261.

Cimino & Grazia Gnoni et al. 2023

Cimino, A.; Grazia Gnoni, M.; Longo, F. & Solina, V. (2023), „Integrating Multiple Industry 4.0 Approaches and Tools in an Interoperable Platform for Manufacturing SMEs“, *Computers & Industrial Engineering* 186, p. 109732. DOI: 10.1016/j.cie.2023.109732.

Cutrona & Bonomi et al. 2024

Cutrona, V.; Bonomi, N.; Montini, E.; Ruppert, T.; Delinavelli, G. & Pedrazzoli, P. (2024), „Extending Factory Digital Twins through Human Characterisation in Asset Administration Shell“, *International Journal of Computer Integrated Manufacturing* 37.10-11, pp. 1214–1231. DOI: 10.1080/0951192X.2023.2278108.

Dam & Klausner et al. 2024

Dam, T.; Klausner, L. D.; Neumaier, S. & Priebe, T. (2024), *A Survey of Dataspace Connector Implementations*. DOI: 10.48550/arXiv.2309.11282. URL: http://arxiv.org/abs/2309.11282 (visited on 11/29/2025).

De Lauretis 2019

De Lauretis, L. (2019), „From Monolithic Architecture to Microservices Architecture“. *2019*

IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 93–96. DOI: 10.1109/ISSREW.2019.00050.

Dmitry & Manfred 2014

Dmitry, N. & Manfred, S.-S. (2014), „On Micro-Services Architecture“, *International Journal of Open Information Technologies* 2.9, pp. 24–27.

Doan & Halevy et al. 2012

Doan, A.; Halevy, A. & Ives, Z. (2012), *Principles of Data Integration*. Burlington, MA: Morgan Kaufmann.

Drath & Luder et al. 2008

Drath, R.; Luder, A.; Peschke, J. & Hundt, L. (2008), „AutomationML - the Glue for Seamless Automation Engineering“. *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 616–623. DOI: 10.1109/ETFA.2008.4638461.

Drath & Mosch et al. 2023

Drath, R.; Mosch, C.; Hoppe, S.; Faath, A.; Barnstedt, E.; Fiebiger, B. & Schlögl, W. (2023), *Interoperabilität Mit Der Verwaltungsschale, OPC UA Und AutomationML: Zielbild Und Handlungsempfehlungen Für Industrielle Interoperabilität*. Diskussionspapier. AutomationML e.V., IDTA, OPC Foundation, VDMA, Microsoft, KUKA, Siemens.

Dumitrescu & Albers et al. 2021

Dumitrescu, R.; Albers, A.; Riedel, O.; Stark, R. & Gausemeier, J. (2021), *Advanced Systems Engineering – Value Creation in Transition: Engineering in Germany – Status Quo in Business and Science*. Paderborn, Germany: Fraunhofer IEM.

Dyckhoff & Spengler 2007

Dyckhoff, H. & Spengler, T. S. (2007), *Produktionswirtschaft. Eine Einführung für Wirtschaftsingenieure*. 2nd ed. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-540-72218-2.

Ebert & Gallardo et al. 2016

Ebert, C.; Gallardo, G.; Hernantes, J. & Serrano, N. (2016), „DevOps“, *IEEE Software* 33.3, pp. 94–100. DOI: 10.1109/MS.2016.68.

ECLASS Standard: The Cross-Industry Reference Data Standard for Products and Services 2024

ECLASS Standard: The Cross-Industry Reference Data Standard for Products and Services (2024). Cologne, Germany: ECLASS e. V.

Eclipse BaSyx – Industry 4.0 Operating System 2025

Eclipse BaSyx – Industry 4.0 Operating System (2025). https://github.com/eclipse-basyx/basyx-java-server-sdk. Open-source project, accessed 2025-10-22.

Ellwein & Neumann et al. 2023

Ellwein, C.; Neumann, R. & Verl, A. (2023), „Software-Defined Manufacturing: Data Repre-

sentation“, *Procedia CIRP*. 16th CIRP Conference on Intelligent Computation in Manufacturing Engineering 118, pp. 360–365. DOI: 10.1016/j.procir.2023.06.062.

ElMaraghy & Monostori et al. 2021
ElMaraghy, H.; Monostori, L.; Schuh, G. & ElMaraghy, W. (2021), „Evolution and Future of Manufacturing Systems“, *CIRP Annals* 70.2, pp. 635–658. DOI: 10.1016/j.cirp.2021.05.008.

ElMaraghy 2005
ElMaraghy, H. A. (2005), „Flexible and Reconfigurable Manufacturing Systems Paradigms“, *International Journal of Flexible Manufacturing Systems* 17.4, pp. 261–276. DOI: 10.1007/s10696-006-9028-7.

ElMaraghy 2009
ElMaraghy, H. A., ed. (2009), *Changeable and Reconfigurable Manufacturing Systems*. Springer Series in Advanced Manufacturing. London: Springer. DOI: 10.1007/978-1-84882-067-8.

ElMaraghy & ElMaraghy et al. 2012
ElMaraghy, W.; ElMaraghy, H.; Tomiyama, T. & Monostori, L. (2012), „Complexity in Engineering Design and Manufacturing“, *CIRP Annals* 61.2, pp. 793–814. DOI: 10.1016/j.cirp.2012.05.001.

Elnadi & Abdallah 2024
Elnadi, M. & Abdallah, Y. O. (2024), „Industry 4.0: Critical Investigations and Synthesis of Key Findings“, *Management Review Quarterly* 74.2, pp. 711–744. DOI: 10.1007/s11301-022-00314-4.

Engelsberger & Greiner 2018
Engelsberger, M. & Greiner, T. (2018), „Dynamic Reconfiguration of Service-Oriented Resources in Cyber–Physical Production Systems by a Process-Independent Approach with Multiple Criteria and Multiple Resource Management Operations“, *Future Generation Computer Systems* 88, pp. 424–441. DOI: 10.1016/j.future.2018.06.002.

Erl 2016
Erl, T. (2016), *Service-Oriented Architecture: Analysis and Design for Services and Microservices*. Upper Saddle River, NJ: Prentice Hall Press.

Eugster & Felber et al. 2003
Eugster, P. T.; Felber, P. A.; Guerraoui, R. & Kermarrec, A.-M. (2003), „The Many Faces of Publish/Subscribe“, *ACM Comput. Surv.* 35.2, pp. 114–131. DOI: 10.1145/857076.857078.

Eversheim 2002
Eversheim, W. (2002), *Organisation in der Produktionstechnik 3*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-642-56336-2.

Fallah & Wolny et al. 2016
Fallah, S. M.; Wolny, S. & Wimmer, M. (2016), „Towards Model-Integrated Service-Oriented

Manufacturing Execution System“. *2016 1st International Workshop on Cyber-Physical Production Systems (CPPS)*, pp. 1–5. DOI: 10.1109/CPPS.2016.7483917.

Faller & Höftmann 2018

Faller, C. & Höftmann, M. (2018), „Service-Oriented Communication Model for Cyber-Physical-Production-Systems“, *Procedia CIRP*. 11th CIRP Conference on Intelligent Computation in Manufacturing Engineering 67, pp. 156–161. DOI: 10.1016/j.procir.2017.12.192.

Fernandez & Elmore et al. 2023

Fernandez, R. C.; Elmore, A. J.; Franklin, M. J.; Krishnan, S. & Tan, C. (2023), „How Large Language Models Will Disrupt Data Management“, *Proc. VLDB Endow.* 16.11, pp. 3302–3309. DOI: 10.14778/3611479.3611527.

Fielding 2000

Fielding, R. T. (2000), „Architectural Styles and the Design of Network-Based Software Architectures“. Dissertation. Irvine, CA: University of California, Irvine.

Frick & Ellwein et al. 2024

Frick, F.; Ellwein, C.; Lechler, A.; Neubauer, M. & Verl, A. (2024), „Software-Defined Manufacturing: Reference Architecture“. *2024 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, pp. 1289–1295. DOI: 10.1109/SPEEDAM61530.2024.10609236.

Gleich & Behrendt et al. 2024

Gleich, K.; Behrendt, S.; Hörger, M.; Benfer, M. & Lanza, G. (2024), „An Asset Administration Shell-Based Digital Product Passport as a Gaia-X Service“, *Procedia CIRP* 127, p. 224. DOI: 10.1016/j.procir.2024.07.039.

Göppert & Grahn et al. 2023

Göppert, A.; Grahn, L.; Rachner, J.; Grunert, D.; Hort, S. & Schmitt, R. H. (2023), „Pipeline for Ontology-Based Modeling and Automated Deployment of Digital Twins for Planning and Control of Manufacturing Systems“, *Journal of Intelligent Manufacturing* 34.5, pp. 2133–2152. DOI: 10.1007/s10845-021-01860-6.

Grassi & Guizzi et al. 2020

Grassi, A.; Guizzi, G.; Santillo, L. C. & Vespoli, S. (2020), „A Semi-Heterarchical Production Control Architecture for Industry 4.0-Based Manufacturing Systems“, *Manufacturing Letters* 24, pp. 43–46. DOI: 10.1016/j.mfglet.2020.03.007.

Grüner & Hoernicke et al. 2023

Grüner, S.; Hoernicke, M.; Stark, K.; Schoch, N.; Eskandani, N. & Pretlove, J. (2023), „Towards Asset Administration Shell-Based Continuous Engineering in Process Industries“, *at - Automatisierungstechnik* 71.8, pp. 689–708. DOI: 10.1515/auto-2023-0012.

Gutenschwager & Rabe et al. 2017

Gutenschwager, K.; Rabe, M.; Spieckermann, S. & Wenzel, S. (2017), *Simulation in Produktion und Logistik*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-662-55745-7.

Hackstein 1984

Hackstein, R. (1984), *Produktionsplanung und -steuerung (PPS). Ein Handbuch für die Betriebspraxis*. Düsseldorf: VDI-Verlag.

Hansen & Madnick et al. 2003

Hansen, M.; Madnick, S. & Siegel, M. (2003), „Data Integration Using Web Services". *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web*. Ed. by S. Bressan; M. L. Lee; A. B. Chaudhri; J. X. Yu & Z. Lacroix. Berlin, Heidelberg: Springer, pp. 165–182. DOI: 10.1007/3-540-36556-7_15.

Hartig & Pérez 2018

Hartig, O. & Pérez, J. (2018), „Semantics and Complexity of GraphQL". *Proceedings of the 2018 World Wide Web Conference*. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, pp. 1155–1164. DOI: 10.1145/3178876.3186014.

Heidel & Hoffmeister et al. 2017

Heidel, R.; Hoffmeister, M.; Hankel, M. & Döbrich, U. (2017), *Industrie4.0 Basiswissen RAMI4.0: Referenzarchitekturmodell mit Industrie4.0-Komponente*. Ed. by Deutsches Institut für Normung. 1. Auflage. DIN. Berlin, Wien, Zürich: Beuth Verlag GmbH. 142 pp.

Hevner & March et al. 2004

Hevner, A. R.; March, S. T.; Park, J. & Ram, S. (2004), „Design Science in Information Systems Research", *MIS Quarterly* 28.1, pp. 75–105. DOI: 10.2307/25148625.

Hull 1997

Hull, R. (1997), „Managing Semantic Heterogeneity in Databases: A Theoretical Prospective". *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. Tucson Arizona USA: ACM, pp. 51–61. DOI: 10.1145/263661.263668.

Human & Basson et al. 2021

Human, C.; Basson, A. H. & Kruger, K. (2021), „Digital Twin Data Pipeline Using MQTT in SLADTA". *Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future*. Ed. by T. Borangiu; D. Trentesaux; P. Leitão; O. Cardin & S. Lamouri. Studies in Computational Intelligence. Cham: Springer International Publishing, pp. 111–122. DOI: 10.1007/978-3-030-69373-2_7.

Hunkeler & Truong et al. 2008

Hunkeler, U.; Truong, H. L. & Stanford-Clark, A. (2008), „MQTT-S — A Publish/Subscribe Protocol for Wireless Sensor Networks". *2008 3rd International Conference on Communi-*

cation Systems Software and Middleware and Workshops (COMSWARE '08), pp. 791–798. DOI: 10.1109/COMSWA.2008.4554519.

IEC 62264 / ISA-95: Enterprise-Control System Integration 2013

IEC 62264 / ISA-95: Enterprise-Control System Integration (2013). Geneva, Switzerland: International Electrotechnical Commission (IEC) and International Society of Automation (ISA).

IEC Common Data Dictionary (CDD) 2024

IEC Common Data Dictionary (CDD) (2024). Geneva, Switzerland: International Electrotechnical Commission (IEC).

Industrial Digital Twin Association 2023a

Industrial Digital Twin Association (2023a), *Specification of the Asset Administration Shell, Part 1: Metamodel*. IDTA Number 01001-3-0. Frankfurt am Main, Germany: Industrial Digital Twin Association (IDTA).

Industrial Digital Twin Association 2023b

Industrial Digital Twin Association (2023b), *Specification of the Asset Administration Shell, Part 2: Application Programming Interfaces*. IDTA Number 01002-3-0. Frankfurt am Main, Germany: Industrial Digital Twin Association (IDTA).

Iñigo & Legaristi et al. 2022

Iñigo, M. A.; Legaristi, J.; Larrinaga, F.; Perez, A.; Cuenca, J.; Kremer, B.; Montejo, E. & Porto, A. (2022), „Towards Standardized Manufacturing as a Service through Asset Administration Shell and International Data Spaces Connectors". *IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1–6. DOI: 10.1109/IECON49645.2022.9968592.

IO-Link: The Standardized Communication Interface for Sensors and Actuators 2023

IO-Link: The Standardized Communication Interface for Sensors and Actuators (2023). Karlsruhe, Germany: IO-Link Consortium.

Jensen & Kristensen 2009

Jensen, K. & Kristensen, L. M. (2009), *Coloured Petri Nets. Modelling and Validation of Concurrent Systems*. Berlin, Heidelberg: Springer. DOI: 10.1007/b95112.

Jin & Sahni et al. 2018

Jin, B.; Sahni, S. & Shevat, A. (2018), *Designing Web APIs*. Sebastopol, CA: O'Reilly Media.

Kagermann & Wahlster et al. 2013

Kagermann, H.; Wahlster, W. & Helbig, J. (2013), *Umsetzungsempfehlungen Für Das Zukunftsprojekt Industrie 4.0. Deutschlands Zukunft Als Produktionsstandort Sichern. Abschlussbericht Des Arbeitskreises Industrie 4.0.* online. Berlin: Forschungsunion Wirtschaft - Wissenschaft; Deutsche Akademie der Technikwissenschaften.

Kaib 2002

Kaib, M. (2002), *Enterprise Application Integration - Grundlagen, Integrationsprodukte, Anwendungsbeispiele*. Wiesbaden: Deutscher Universitätsverlag. DOI: 10.1007/978-3-663-07913-2.

Kang & Krishnaswamy et al. 2020

Kang, Y.-B.; Krishnaswamy, S.; Sawangphol, W.; Gao, L. & Li, Y.-F. (2020), „Understanding and Improving Ontology Reasoning Efficiency through Learning and Ranking“, *Information Systems* 87, p. 101412. DOI: 10.1016/j.is.2019.07.002.

Kayali & Wenz et al. 2024

Kayali, M.; Wenz, F.; Tatbul, N. & Demiralp, Ç. (2024), *Mind the Data Gap: Bridging LLMs to Enterprise Data Integration*. DOI: 10.48550/arXiv.2412.20331. URL: http://arxiv.org/abs/2412.20331 (visited on 07/15/2025).

Kellner & Lienland et al. 2020

Kellner, F.; Lienland, B. & Lukesch, M. (2020), *Produktionswirtschaft: Planung, Steuerung und Industrie 4.0*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-662-61446-4.

Koren & Heisel et al. 1999

Koren, Y.; Heisel, U.; Jovane, F.; Moriwaki, T.; Pritschow, G.; Ulsoy, G. & Van Brussel, H. (1999), „Reconfigurable Manufacturing Systems“, *CIRP Annals* 48.2, pp. 527–540. DOI: 10.1016/S0007-8506(07)63232-6.

Koren & Gu et al. 2018

Koren, Y.; Gu, X. & Guo, W. (2018), „Reconfigurable Manufacturing Systems: Principles, Design, and Future Trends“, *Frontiers of Mechanical Engineering* 13.2, pp. 121–136. DOI: 10.1007/s11465-018-0483-0.

Krafzig & Banke et al. 2004

Krafzig, D.; Banke, K. & Slama, D. (2004), *Enterprise SOA. Service-Oriented Architecture Best Practices*. Upper Saddle River, NJ: Prentice Hall PTR.

Krenczyk & Paprocka 2023

Krenczyk, D. & Paprocka, I. (2023), „Integration of Discrete Simulation, Prediction, and Optimization Methods for a Production Line Digital Twin Design“, *Materials* 16. DOI: 10.3390/ma16062339.

Kritzinger & Karner et al. 2018

Kritzinger, W.; Karner, M.; Traar, G.; Henjes, J. & Sihn, W. (2018), „Digital Twin in Manufacturing: A Categorical Literature Review and Classification“, *IFAC-PapersOnLine*. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018 51.11, pp. 1016–1022. DOI: 10.1016/j.ifacol.2018.08.474.

Kuhnle & Kaiser et al. 2021

Kuhnle, A.; Kaiser, J.-P.; Theiß, F.; Stricker, N. & Lanza, G. (2021), „Designing an Adaptive

Production Control System Using Reinforcement Learning“, *Journal of Intelligent Manufacturing* 32.3, pp. 855–876. DOI: 10.1007/s10845-020-01612-y.

Kusiak 2018

Kusiak, A. (2018), „Smart Manufacturing“, *International Journal of Production Research* 56.1-2, pp. 508–517. DOI: 10.1080/00207543.2017.1351644.

Lanza & Moser et al. 2015

Lanza, G.; Moser, E.; Stoll, J. & Haefner, B. (2015), „Learning Factory on Global Production“, *Procedia CIRP* 32, pp. 120–125. DOI: 10.1016/j.procir.2015.02.081.

Lechler & Schlechtendahl 2023

Lechler, A. & Schlechtendahl, J. (2023), „Steuerung Aus Der Cloud“. *Handbuch Industrie 4.0: Band 1: Produktion*. Ed. by T. Bauernhansl. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 13–25. DOI: 10.1007/978-3-662-58532-0_27.

Lechler & Verl 2017

Lechler, A. & Verl, A. (2017), „Software Defined Manufacturing Extends Cloud-Based Control“. *Proceedings of the 12th International Manufacturing Science and Engineering Conference (MSEC 2017)*. American Society of Mechanical Engineers. DOI: 10.1115/MSEC2017-2656.

Lemos & Daniel et al. 2015

Lemos, A. L.; Daniel, F. & Benatallah, B. (2015), „Web Service Composition: A Survey of Techniques and Tools“, *ACM Comput. Surv.* 48.3, 33:1–33:41. DOI: 10.1145/2831270.

Lenzerini 2002

Lenzerini, M. (2002), „Data Integration: A Theoretical Perspective“. *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. Madison Wisconsin: ACM, pp. 233–246. DOI: 10.1145/543613.543644.

Leser & Naumann 2007

Leser, U. & Naumann, F. (2007), *Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. 1. Auflage. Heidelberg: dpunkt.verlag.

Liu & Jiang et al. 2020

Liu, C.; Jiang, P. & Jiang, W. (2020), „Web-Based Digital Twin Modeling and Remote Control of Cyber-Physical Production Systems“, *Robotics and Computer-Integrated Manufacturing* 64, p. 101956. DOI: 10.1016/j.rcim.2020.101956.

Lödding 2016

Lödding, H. (2016), *Verfahren der Fertigungssteuerung*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-662-48459-3.

Lopez & Shao et al. 2018

Lopez, F.; Shao, Y.; Mao, Z. M.; Moyne, J.; Barton, K. & Tilbury, D. (2018), „A Software-

Defined Framework for the Integrated Management of Smart Manufacturing Systems“, *Manufacturing Letters* 15, pp. 18–21. DOI: 10.1016/j.mfglet.2017.12.015.

Luo & Thevenin et al. 2023

Luo, D.; Thevenin, S. & Dolgui, A. (2023), „A State-of-the-Art on Production Planning in Industry 4.0“, *International Journal of Production Research*. DOI: https://doi.org/10.1080/00207543.2022.2122622.

Madhavan & Bernstein et al. 2001

Madhavan, J.; Bernstein, P. A. & Rahm, E. (2001), „Generic Schema Matching with Cupid“. *Proceedings of the 27th International Conference on Very Large Data Bases*. VLDB ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 49–58.

Mahnke & Leitner et al. 2009

Mahnke, W.; Leitner, S.-H. & Damm, M. (2009), *OPC Unified Architecture*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-540-68899-0.

Malik & Sharma et al. 2021

Malik, P. K.; Sharma, R.; Singh, R.; Gehlot, A.; Satapathy, S. C.; Alnumay, W. S.; Pelusi, D.; Ghosh, U. & Nayak, J. (2021), „Industrial Internet of Things and Its Applications in Industry 4.0: State of The Art“, *Computer Communications* 166, pp. 125–139. DOI: 10.1016/j.comcom.2020.11.016.

Mehrabi & Ulsoy et al. 2000

Mehrabi, M. G.; Ulsoy, A. G. & Koren, Y. (2000), „Reconfigurable Manufacturing Systems: Key to Future Manufacturing“, *Journal of Intelligent Manufacturing* 11.4, pp. 403–419. DOI: 10.1023/A:1008930403506.

Metović & Maisch et al. 2026

Metović, A.; Maisch, N.; Ajdinović, S.; Lechler, A.; Wortmann, A. & Riedel, O. (2026), *Industrial Semantics-Aware Digital Twins: A Hybrid Graph Matching Approach for Asset Administration Shells*. DOI: 10.48550/arXiv.2601.06613. (Visited on 01/21/2026).

Meudt & Wonnemann et al. 2017

Meudt, T.; Wonnemann, A. & Metternich, J. (2017), *Produktionsplanung und -steuerung (PPS): Ein Überblick der Literatur der unterschiedlichen Einteilung von PPS-Konzepten*. TUprints publication (CC BY-NC-ND 4.0). Darmstadt: Technische Universität Darmstadt, Institut für Produktionsmanagement, Technologie und Werkzeugmaschinen (PTW). DOI: 10.26083/tuprints-00006654.

Mönch & Fowler et al. 2013

Mönch, L.; Fowler, J. W. & Mason, S. J. (2013), *Production Planning and Control for Semiconductor Wafer Fabrication Facilities: Modeling, Analysis, and Systems*. Vol. 52. Operations Research/Computer Science Interfaces Series. New York, NY: Springer. DOI: 10.1007/978-1-4614-4472-5.

Monostori 2014
Monostori, L. (2014), „Cyber-Physical Production Systems: Roots, Expectations and R&D Challenges“, *Procedia CIRP*. Variety Management in Manufacturing 17, pp. 9–13. DOI: 10.1016/j.procir.2014.03.115.

MTConnect Standard: Open, Extensible Communication Protocol for Machine Tools and Manufacturing Equipment 2023
MTConnect Standard: Open, Extensible Communication Protocol for Machine Tools and Manufacturing Equipment (2023). McLean, VA, USA: MTConnect Institute.

Newnan & Lavelle et al. 2019
Newnan, D.; Lavelle, J. & Newnan Te, D. (2019), *Engineering Economic Analysis*. 14th ed. Oxford University Press.

Niknejad & Ismail et al. 2020
Niknejad, N.; Ismail, W.; Ghani, I.; Nazari, B.; Bahari, M. & Hussin, A. R. B. C. (2020), „Understanding Service-Oriented Architecture (SOA): A Systematic Literature Review and Directions for Further Investigation“, *Information Systems* 91, p. 101491. DOI: 10.1016/j.is.2020.101491.

Nowshin & Mohamed et al. 2025
Nowshin, I.; Mohamed, G.; Schöttke, D.; Schäfer, S. & Zielstorff, A. (2025), „An Automated Approach to Compliance Testing of Standardized Submodel Templates“. *2025 IEEE 8th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pp. 1–7. DOI: 10.1109/ICPS65515.2025.11087912.

Nyhuis & Hübner et al. 2017
Nyhuis, P.; Hübner, M.; Quirico, M.; Schäfers, P. & Schmidt, M. (2017), „Veränderung in Der Produktionsplanung Und -Steuerung“. *Handbuch Industrie 4.0*. Carl Hanser Verlag GmbH & Co. KG, pp. 31–50. DOI: 10.3139/9783446449893.002.

Oluyisola & Bhalla et al. 2022
Oluyisola, O. E.; Bhalla, S.; Sgarbossa, F. & Strandhagen, J. O. (2022), „Designing and Developing Smart Production Planning and Control Systems in the Industry 4.0 Era: A Methodology and Case Study“, *Journal of Intelligent Manufacturing* 33.1, pp. 311–332. DOI: 10.1007/s10845-021-01808-w.

Overbeck & Graves et al. 2024
Overbeck, L.; Graves, S. C. & Lanza, G. (2024), „Development and Analysis of Digital Twins of Production Systems“, *International Journal of Production Research* 62.2024, pp. 3544–3558. DOI: 10.1080/00207543.2023.2242525.

Papazoglou & Van Den Heuvel 2006
Papazoglou, M. P. & Van Den Heuvel, W.-J. (2006), „Service-Oriented Design and De-

velopment Methodology“, *International Journal of Web Engineering and Technology* 2.4, pp. 412–442. DOI: 10.1504/IJWET.2006.010423.

Parciak & Vandevoort et al. 2024

Parciak, M.; Vandevoort, B.; Neven, F.; Peeters, L. M. & Vansummeren, S. (2024), *Schema Matching with Large Language Models: An Experimental Study*. DOI: 10.48550/arXiv.2407.11852. URL: http://arxiv.org/abs/2407.11852 (visited on 11/26/2024).

Pedro Lopes & Ibrahim et al. 2024

Pedro Lopes, R.; Ibrahim, A.; Barbosa, J. & Leitao, P. (2024), „Microservices Architecture to Enable an Open Platform for Realizing Zero Defects in Cyber-Physical Manufacturing“, *Logic Journal of the IGPL*, jzae112. DOI: 10.1093/jigpal/jzae112.

Peffers & Tuunanen et al. 2007

Peffers, K.; Tuunanen, T.; Rothenberger, M. A. & Chatterjee, S. (Dec. 2007), „A Design Science Research Methodology for Information Systems Research“, *Journal of Management Information Systems* 24.3, pp. 45–77. DOI: 10.2753/MIS0742-1222240302.

Pereira & Cainelli et al. 2023

Pereira, P. H. M.; Cainelli, G.; Pereira, C. E. & de Freitas, E. P. (2023), „Interoperability Middleware for IIoT Gateways Based on Standard Ontologies and AAS“, *IFAC-PapersOnLine*. 22nd IFAC World Congress 56.2, pp. 9831–9836. DOI: 10.1016/j.ifacol.2023.10.403.

Pérez & Irisarri et al. 2015

Pérez, F.; Irisarri, E.; Orive, D.; Marcos, M. & Estevez, E. (2015), „A CPPS Architecture Approach for Industry 4.0“. *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–4. DOI: 10.1109/ETFA.2015.7301606.

Petrasch & Petrasch 2022

Petrasch, R. J. & Petrasch, R. R. (2022), „Data Integration and Interoperability: Towards a Model-Driven and Pattern-Oriented Approach“, *Modelling* 3.1, pp. 105–126. DOI: 10.3390/modelling3010008.

Pfrommer & Stogl et al. 2015

Pfrommer, J.; Stogl, D.; Aleksandrov, K.; Navarro, S. E.; Hein, B. & Beyerer, J. (2015), „Plug & Produce by Modelling Skills and Service-Oriented Orchestration of Reconfigurable Manufacturing Systems“, *at - Automatisierungstechnik* 63.10, pp. 790–800. DOI: 10.1515/auto-2014-1157.

Qin & Liu et al. 2016

Qin, J.; Liu, Y. & Grosvenor, R. (2016), „A Categorical Framework of Manufacturing for Industry 4.0 and Beyond“, *Procedia CIRP*. The Sixth International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2016) 52, pp. 173–178. DOI: 10.1016/j.procir.2016.08.005.

Qiu 2007

Qiu, R. G. (2007), „A Service-Oriented Integration Framework for Semiconductor Manufacturing Systems“, *International Journal of Manufacturing Technology and Management* 10.2-3, pp. 177–191. DOI: 10.1504/IJMTM.2007.011848.

Rahm & Bernstein 2001

Rahm, E. & Bernstein, P. A. (2001), „A Survey of Approaches to Automatic Schema Matching“, *The VLDB Journal* 10.4, pp. 334–350. DOI: 10.1007/s007780100057.

Ray & Ligatti 2012

Ray, D. & Ligatti, J. (2012), „Defining Code-Injection Attacks“, *SIGPLAN Not.* 47.1, pp. 179–190. DOI: 10.1145/2103621.2103678.

Redelinghuys & Basson et al. 2020

Redelinghuys, A. J. H.; Basson, A. H. & Kruger, K. (2020), „A Six-Layer Architecture for the Digital Twin: A Manufacturing Case Study Implementation“, *Journal of Intelligent Manufacturing* 31.6, pp. 1383–1402. DOI: 10.1007/s10845-019-01516-6.

Redelinghuys & Kruger et al. 2020

Redelinghuys, A. J. H.; Kruger, K. & Basson, A. (2020), „A Six-Layer Architecture for Digital Twins with Aggregation“. *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future*. Ed. by T. Borangiu; D. Trentesaux; P. Leitão; A. Giret Boggino & V. Botti. Studies in Computational Intelligence. Cham: Springer International Publishing, pp. 171–182. DOI: 10.1007/978-3-030-27477-1_13.

Reisig 2011

Reisig, W. (2011), „Petri Nets“. *Modeling in Systems Biology: The Petri Net Approach*. Ed. by I. Koch; W. Reisig & F. Schreiber. London: Springer, pp. 37–56. DOI: 10.1007/978-1-84996-474-6_3.

Ren & Zhang et al. 2017

Ren, L.; Zhang, L.; Wang, L.; Tao, F. & Chai, X. (2017), „Cloud Manufacturing: Key Characteristics and Applications“, *International Journal of Computer Integrated Manufacturing* 30.6, pp. 501–515. DOI: 10.1080/0951192X.2014.902105. eprint: https://doi.org/10.1080/0951192X.2014.902105.

Richards 2015

Richards, M. (2015), *Microservices vs. Service-Oriented Architecture*. Sebastopol, CA: O'Reilly Media.

Richardson & Ruby 2007

Richardson, L. & Ruby, S. (2007), *RESTful Web Services*. Sebastopol, CA: O'Reilly Media.

Safi & Murad et al. 2011

Safi, F.; Murad, M.; Sulaiman, m. nasir & Udzir, N. (2011), „Adaptable Decentralized

Service Oriented Architecture“, *Journal of Systems and Software* 84, pp. 1591–1617. DOI: 10.1016/j.jss.2011.03.031.

Saqlain & Piao et al. 2019

Saqlain, M.; Piao, M.; Shim, Y. & Lee, J. Y. (2019), „Framework of an IoT-based Industrial Data Management for Smart Manufacturing“, *Journal of Sensor and Actuator Networks* 8.2 (2), p. 25. DOI: 10.3390/jsan8020025.

Scholten 2007

Scholten, B. (2007), *The Road to Integration: A Guide to Applying the ISA-95 Standard in Manufacturing*. Softcover. Research Triangle Park, NC: International Society of Automation.

Schubert & Kuehner et al. 2023

Schubert, V.; Kuehner, S.; Krauss, T.; Trat, M. & Bender, J. (2023), „Towards a B2B Integration Framework for Smart Services in Industry 4.0“, *Procedia Computer Science*. 4th International Conference on Industry 4.0 and Smart Manufacturing 217, pp. 1649–1659. DOI: 10.1016/j.procs.2022.12.365.

Schuh 2006

Schuh, G., ed. (2006), *Produktionsplanung und -steuerung: Grundlagen, Gestaltung und Konzepte*. Berlin, Heidelberg: Springer. DOI: 10.1007/3-540-33855-1.

Siatras & Bakopoulos et al. 2023

Siatras, V.; Bakopoulos, E.; Mavrothalassitis, P.; Nikolakis, N. & Alexopoulos, K. (2023), „On the Use of Asset Administration Shell for Modeling and Deploying Production Scheduling Agents within a Multi-Agent System“, *Applied Sciences* 13.17 (17), p. 9540. DOI: 10.3390/app13179540.

Skouradaki & Ferme et al. 2016

Skouradaki, M.; Ferme, V.; Pautasso, C.; Leymann, F. & Hoorn, A. van (2016), „Micro-Benchmarking BPMN 2.0 Workflow Management Systems with Workflow Patterns“. *Advanced Information Systems Engineering: 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13–17, 2016, Proceedings*. Ed. by S. Nurcan; P. Soffer; M. Bajec & J. Eder. Cham: Springer International Publishing, pp. 67–82. DOI: 10.1007/978-3-319-39696-5_5.

Słodziak & Nowak 2016

Słodziak, W. & Nowak, Z. (2016), „Performance Analysis of Web Systems Based on XMLHttpRequest, Server-Sent Events and WebSocket“. *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II*. Ed. by A. Grzech; L. Borzemski; J. Świątek & Z. Wilimowska. Cham: Springer International Publishing, pp. 71–83. DOI: 10.1007/978-3-319-28561-0_6.

Stark & Fresemann et al. 2019

Stark, R.; Fresemann, C. & Lindow, K. (2019), „Development and Operation of Digital Twins for Technical Systems and Services“, *CIRP Annals* 68.1, pp. 129–132. DOI: 10.1016/j.cirp.2019.04.024.

Stricker & Lanza 2014

Stricker, N. & Lanza, G. (2014), „The Concept of Robustness in Production Systems and Its Correlation to Disturbances“, *Procedia CIRP* 19, pp. 87–92. DOI: 10.1016/j.procir.2014.04.078.

Tao & Qi et al. 2018

Tao, F.; Qi, Q.; Liu, A. & Kusiak, A. (2018), „Data-Driven Smart Manufacturing“, *Journal of Manufacturing Systems*. Special Issue on Smart Manufacturing 48, pp. 157–169. DOI: 10.1016/j.jmsy.2018.01.006.

Thames & Schaefer 2016

Thames, L. & Schaefer, D. (2016), „Software-Defined Cloud Manufacturing for Industry 4.0“, *Procedia CIRP*. The Sixth International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2016) 52, pp. 12–17. DOI: 10.1016/j.procir.2016.07.041.

Trunzer & Calà et al. 2019

Trunzer, E.; Calà, A.; Leitão, P.; Gepp, M.; Kinghorst, J.; Lüder, A.; Schauerte, H.; Reifferscheid, M. & Vogel-Heuser, B. (2019), „System Architectures for Industrie 4.0 Applications“, *Production Engineering* 13.3, pp. 247–257. DOI: 10.1007/s11740-019-00902-6.

Tzavaras & Mainas et al. 2023

Tzavaras, A.; Mainas, N. & Petrakis, E. G. M. (2023), „OpenAPI Framework for the Web of Things“, *Internet of Things* 21, p. 100675. DOI: 10.1016/j.iot.2022.100675.

Valipour & Amirzafari et al. 2009

Valipour, M. H.; Amirzafari, B.; Maleki, K. N. & Daneshpour, N. (2009), „A Brief Survey of Software Architecture Concepts and Service Oriented Architecture“. *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pp. 34–38. DOI: 10.1109/ICCSIT.2009.5235004.

Van Der Aalst 1998

Van Der Aalst, W. M. P. (1998), „The Application of Petri Nets to Workflow Management“, *Journal of Circuits, Systems and Computers* 08.01, pp. 21–66. DOI: 10.1142/S0218126698000043.

Vaswani & Shazeer et al. 2017

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł. ukasz & Polosukhin, I. (2017), „Attention Is All You Need“. *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc. DOI: 10.48550/arXiv.1706.03762.

Verband Deutscher Maschinen- und Anlagenbau e. V. (VDMA) 2024
Verband Deutscher Maschinen- und Anlagenbau e. V. (VDMA) (Mar. 2024), *LIF - Layout Interchange Format: Definition of a format of track layouts for exchange between the integrator of the driverless transport vehicles and a (third-party) master control system*. Tech. rep. Version 1.0.0. Accessed: 2025-06-30. Frankfurt am Main, Germany: Verband Deutscher Maschinen- und Anlagenbau e. V. (VDMA).

Vyskočil & Douda et al. 2023
Vyskočil, J.; Douda, P.; Novák, P. & Wally, B. (2023), „A Digital Twin-Based Distributed Manufacturing Execution System for Industry 4.0 with AI-Powered On-The-Fly Replanning Capabilities“, *Sustainability* 15. DOI: 10.3390/su15076251.

Wang & Keivanloo et al. 2014
Wang, S.; Keivanloo, I. & Zou, Y. (2014), „How Do Developers React to RESTful API Evolution?“ *Service-Oriented Computing*. Ed. by X. Franch; A. K. Ghose; G. A. Lewis & S. Bhiri. Berlin, Heidelberg: Springer, pp. 245–259. DOI: 10.1007/978-3-662-45391-9_17.

Wiendahl & ElMaraghy et al. 2007
Wiendahl, H.-P.; ElMaraghy, H. A.; Nyhuis, P.; Zäh, M. F.; Wiendahl, H.-H.; Duffie, N. & Brieke, M. (2007), „Changeable Manufacturing - Classification, Design and Operation“, *CIRP Annals* 56.2, pp. 783–809. DOI: 10.1016/j.cirp.2007.10.003.

Wiendahl & Reichardt et al. 2023
Wiendahl, H.-H.; Reichardt, J. & Nyhuis, P. (2023), *Handbuch Fabrikplanung*. München: Carl Hanser Verlag GmbH & Co. KG. DOI: 10.3139/9783446473607.

Wilhelm & Niermann et al. 2024
Wilhelm, J.; Niermann, D.; Keiser, D. & Freitag, M. (2024), „Towards Holistic Interoperability of Cyber-Physical Production Systems within RAMI 4.0“, *Procedia Computer Science*. 5th International Conference on Industry 4.0 and Smart Manufacturing (ISM 2023) 232, pp. 946–955. DOI: 10.1016/j.procs.2024.01.094.

Ye & Hong et al. 2021
Ye, X.; Hong, S. H.; Song, W. S.; Kim, Y. C. & Zhang, X. (2021), „An Industry 4.0 Asset Administration Shell-Enabled Digital Solution for Robot-Based Manufacturing Systems“, *IEEE Access* 9, pp. 154448–154459. DOI: 10.1109/ACCESS.2021.3128580.

Ye & Xu et al. 2023
Ye, X.; Xu, W.; Liu, J.; Zhong, Y.; Liu, Q.; Zhou, Z.; Song, W. S. & Hong, S. H. (2023), „Implementing Digital Twin and Asset Administration Shell Models for a Simulated Sorting Production System“, *IFAC-PapersOnLine*. 22nd IFAC World Congress 56.2, pp. 11880–11887. DOI: 10.1016/j.ifacol.2023.10.600.

Yin 2018
Yin, R. K. (2018), *Case study research and applications*. Vol. 6. Sage Thousand Oaks, CA.

Zelewski & Hohmann et al. 2010

Zelewski, S.; Hohmann, S. & Hügens, T. (2010), *Produktionsplanungs- und -steuerungssysteme. Konzepte und exemplarische Implementierungen mithilfe von SAP R/3*. Oldenbourg Wissenschaftsverlag. DOI: 10.1524/9783486599862.

Zhao & Zhou et al. 2025

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; Du, Y.; Yang, C.; Chen, Y.; Chen, Z.; Jiang, J.; Ren, R.; Li, Y.; Tang, X.; Liu, Z.; Liu, P.; Nie, J.-Y. & Wen, J.-R. (2025), *A Survey of Large Language Models*. DOI: 10.48550/arXiv.2303.18223. arXiv: 2303.18223 [cs]. URL: http://arxiv.org/abs/2303.18223 (visited on 08/11/2025). Pre-published.

Zhong & Xu et al. 2017

Zhong, R. Y.; Xu, X.; Klotz, E. & Newman, S. T. (2017), „Intelligent Manufacturing in the Context of Industry 4.0: A Review“, *Engineering* 3.5, pp. 616–630. DOI: 10.1016/J.ENG.2017.05.015.

List of Figures

List of Tables

List of own publications

Behrendt, S.; Ungen, M.; Fisel, J.; Hung, K.-C.; May, M.-C.; Leberle, U. & Lanza, G. (2023), „Improving Production System Flexibility and Changeability Through Software-Defined Manufacturing“. *Production at the Leading Edge of Technology*. Ed. by M. Liewald; A. Verl; T. Bauernhansl & H.-C. Möhring. Lecture Notes in Production Engineering. Cham: Springer International Publishing, pp. 705–716. DOI: 10.1007/978-3-031-18318-8_70.

Behrendt, S.; Altenmüller, T.; May, M. C.; Kuhnle, A. & Lanza, G. (2025), „Real-to-Sim: Automatic Simulation Model Generation for a Digital Twin in Semiconductor Manufacturing“, *Journal of Intelligent Manufacturing*. DOI: 10.1007/s10845-025-02572-x.

Behrendt, S.; Bail, F.; Benfer, M. & Lanza, G. (2025), „Aas-Middleware: Enabling Interoperable, Real-Time Integration for Smart Manufacturing“. *Proceedings of the 5th Modeling, Estimation and Control Conference (MECC 2025)*. Vol. 59. 30. Pittsburgh, Pennsylvania, USA: IFAC-PapersOnLine, pp. 671–676. DOI: 10.1016/j.ifacol.2025.12.315.

Behrendt, S.; Enke, C.; Rüdt, M.; Bussmann, M.; Tutsch, H.; Spieckermann, S. & Lanza, G. (2026), „Closing the Loop: A Software-Defined Approach to Agile Production Planning and Control“, *Journal of Manufacturing Systems* 86, pp. 1007–1021. DOI: 10.1016/j.jmsy.2026.03.025.

Behrendt, S.; Martin, M.; Puchta, A.; Ströbel, R.; Fisel, J.; May, M. C.; Gönnheimer, P.; Fleischer, J. & Lanza, G. (2022), „Software-Defined Manufacturing for the Entire Life Cycle at Different Levels of Production“. *Advances in Automotive Production Technology – Towards Software-Defined Manufacturing and Resilient Supply Chains*. Ed. by N. Kiefl; F. Wulle; C. Ackermann & D. Holder. Cham: Springer International Publishing, pp. 25–34. DOI: 10.1007/978-3-031-27933-1_3.

Behrendt, S.; Stamer, F.; May, M. C. & Lanza, G. (2023), „Towards a Service-Oriented Architecture for Production Planning and Control: A Comprehensive Review and Novel Approach“. *Proceedings of the 5th Conference on Production Systems and Logistics (CPSL 2023)*. Ed. by D. Herberger. Stellenbosch, South Africa: publish-Ing., pp. 255–267. DOI: 10.15488/15271.

Behrendt, S.; Wurster, M.; May, M. C. & Lanza, G. (2023a), „Extended Production Planning of Reconfigurable Manufacturing Systems by Means of Simulation-Based Optimization“. *Proceedings of the 4th Conference on Production Systems and Logistics (CPSL 2023)*. Ed. by D. Herberger; M. Hübner & V. Stich. Santiago de Querétaro, Mexico: publish-Ing., pp. 210–220. DOI: 10.15488/13440.

Behrendt, S.; Wurster, M.; Strljic, M.; Klein, J.-F.; May, M. C. & Lanza, G. (2023b), „Interoperable Architecture for Logical Reconfigurations of Modular Production Systems“.

Proceedings of the 5th Conference on Production Systems and Logistics (CPSL 2023). Ed. by D. Herberger. Stellenbosch, South Africa: publish-Ing., p. 622. DOI: 10.15488/15270.

Bilen, A.; Stamer, F.; Behrendt, S. & Lanza, G. (2024), „A Quality Data Model Based on Asset Administration Shell Technology to Enable Autonomous Quality Control Loops“. *Production at the Leading Edge of Technology : Proceedings of the 13th Congress of the German Academic Association for Production Technology (WGP), Freudenstadt, Germany. Ed.: T. Bauernhansl*, p. 195. DOI: 10.1007/978-3-031-47394-4_20.

Bott, A.; Behrendt, S.; Gleich, K.; Karimi, E.; Fleischer, J. & Lanza, G. (2025), „Challenges in Implementing Gaia-X for Industrial Applications – Insights from Three Diverse Use Cases“, *Zeitschrift für wirtschaftlichen Fabrikbetrieb* 120.3, p. 163. DOI: 10.1515/zwf-2025-1023.

Erb, Y.; Teigeler, H.; Lins, S.; Sunyaev, A.; Gleich, K.; Behrendt, S.; Karimi, E.; Bott, A.; Lanza, G.; Schulze, V.; Fleischer, J.; Leander-Knoll, M.; Jaganathan, K.; Scheibenberger, K.; Frick, F.; Bubeck, W.; Neher, P.; Neumann, R.; Lechler, A.; Verl, A.; Riedel, O.; Andriushchenko, O.; Shcherbakov, O.; Hoppe, D. & Brinkmeier, D. (2024), *A Gaia-X-based Ecosystem for Manufacturing: Lessons Learned from the Project Gaia-X4ICM*. Research Report. Karlsruhe Institute of Technology. DOI: 10.5445/IR/1000174929.

Gleich, K.; Behrendt, S.; Hörger, M.; Benfer, M. & Lanza, G. (2024), „An Asset Administration Shell-Based Digital Product Passport as a Gaia-X Service“, *Procedia CIRP* 127, p. 224. DOI: 10.1016/j.procir.2024.07.039.

Martin, M.; Behrendt, S.; Enke, C.; Tutsch, H.; Peukert, S. & Lanza, G. (2023), „Flexibility and Changeability for Software-Defined Manufacturing“. *Production Processes and Product Evolution in the Age of Disruption*. Ed. by F. G. Galizia & M. Bortolini. Lecture Notes in Mechanical Engineering. Cham: Springer International Publishing, pp. 373–380. DOI: 10.1007/978-3-031-34821-1_41.

Neubauer, M.; Ellwein, C.; Frick, F.; Fisel, J.; Kampert, D.; Leberle, U.; May, M. C.; Behrendt, S.; Esslinger, E.; Pfeiffer, D. & Zahn, P. (2022), „Kontinuität – Das Neue Paradigma“, *Computer&Automation* 2022.4/2022, pp. 28–31.

Rechkemmer, D.; Korth, M.; Behrendt, S.; May, M. C.; Benfer, M. & Lanza, G. (2026), „Optimizing Learning Pathways for Digital Twin Education in Manufacturing Systems“. *Advancing Learning Factories: Enabling Future-Ready Skills*. Ed. by L. Louw; V. Hummel; I. de Kock & K. von Leipzig. Cham: Springer Nature Switzerland, pp. 62–69. DOI: 10.1007/978-3-031-98880-6_8.

Wurster, M.; Bail, F.; Behrendt, S. & Lanza, G. (2023), „Adaptive Multi-Priority Rule Approach to Control Agile Disassembly Systems in Remanufacturing“. *Proceedings of the 5th Conference on Production Systems and Logistics (CPSL 2023)*. Ed. by D. Herberger. Stellenbosch, South Africa: publish-Ing., p. 782. DOI: 10.15488/15278.

Appendix

A1 Automation Pyramid

The automation pyramid , displayed in Figure 2.6, is a widely used conceptual model that illustrates the hierarchical structure of industrial automation and control systems. It classifies the various IT and OT systems within a manufacturing enterprise into distinct layers based on their functionality, time horizons, and hardware requirements. This hierarchy is closely aligned with the international standard ISA-95 IEC 62264 (*IEC 62264 / ISA-95: Enterprise-Control System Integration* 2013), which defines the interface between enterprise and control systems.

A1.1 Functional Layers

The pyramid is typically divided into five distinct levels, ranging from the physical production process at the bottom to enterprise-wide planning at the top.

- **Level 0: The Field Level (Production Process).** The base of the pyramid consists of the physical production environment. This includes the sensors that collect data (e.g., temperature, pressure, RFID) and actuators that perform physical actions (e.g., motors, valves, grippers). The primary function here is the direct interaction with the physical product. code Code

- **Level 1: The Control Level (Sensing and Manipulation).** Located directly above the field level, this layer contains the hardware responsible for controlling the physical devices. This typically involves Programmable Logic Controllers (PLCs) and Proportional-Integral-Derivative (PID) controllers. These systems operate in hard real-time, processing signals from Level 0 and generating control commands within milliseconds or microseconds.

- **Level 2: The Supervisory Level (Monitoring).** This layer is responsible for the visualization and supervision of the production process. Systems such as Supervisory Control and Data Acquisition (SCADA) and Human-Machine Interfaces (HMI) allow operators to monitor process states, handle alarms, and manually intervene if necessary.

- **Level 3: The Operations Level (Manufacturing Operations Management).** This layer marks the transition from controlling equipment to managing production workflows. MES reside here. They are responsible for detailed production scheduling, resource allocation, quality management, and tracking Key Performance Indicators (KPIs) like Overall Equipment Effectiveness (OEE). The time horizon shifts from seconds to shifts, days, or batches.

- **Level 4: The Enterprise Level (Business Planning and Logistics).** The apex of the pyramid is occupied by ERP (ERP) systems. This layer manages the broader business functions, including finance, human resources, procurement, and customer order management. The focus is on long-term planning, spanning weeks, months, or years.

A1.2 Hierarchical Isolation and Communication Rigidity

The defining characteristic of the automation pyramid is its strict separation of concerns and rigid communication pathways. Historically, this isolation was a necessary design choice driven by technological constraints and the need for system stability.

Information Aggregation and Filtering. Communication in the pyramid flows vertically. Data generated at the field level (Level 0) is high-frequency and raw. As it moves up the pyramid, it is aggregated and filtered. For example, a PLC (Level 1) might read a sensor every 10 milliseconds, but the MES (Level 3) only requires a "job complete" signal once per hour, and the ERP (Level 4) only needs a daily production report. This aggregation prevents the higher-level business systems, which are not designed for real-time processing, from being overwhelmed by the massive volume of raw sensor data.

Time Horizon Divergence. The layers are isolated because they operate in fundamentally different time domains. Level 1 requires deterministic, low-latency responses to ensure safety and quality, while Level 4 operates on transactional time (waiting for database commits or user inputs). Direct communication between an ERP system and a motor controller is structurally prohibited to prevent business-layer latency from causing physical crashes on the shop floor.

The Information Silo Effect. While this structure ensures stability, it creates significant latency and "information silos." Information must traverse every intermediate layer to move from the shop floor to the top floor. Consequently, semantic context is often lost during aggregation, and business decisions are made based on data that is no longer real-time. This structural rigidity is the primary motivation for the shift toward the decentralized, service-oriented architectures discussed in the main text.

A2 Web Service API and Schema Specifications

Modern industrial software systems rely heavily on standardized interfaces to exchange data. To ensure that independent systems can interpret the structure and semantics of the data being exchanged, formal specification languages are employed. This chapter provides a technical overview of the three most prevalent standards relevant to the data integration challenges discussed in this thesis: the OpenAPI Specification, JSON Schema, and GraphQL.

A2.1 The OpenAPI Specification

The OpenAPI Specification (OAS), formerly known as Swagger, is the de facto standard for defining and documenting RESTful interfaces. It provides a programming language-agnostic interface description language that allows both humans and computers to discover and understand the capabilities of a service without access to source code or additional documentation.

A2.1.1 Mechanism and Structure

An OpenAPI document defines an API's contract. It is typically written in YAML or JSON format and describes the available endpoints (paths), the operations they support (e.g., GET, POST, PUT, DELETE), and the expected format of input parameters and response bodies.

The core value of OAS lies in its machine-readability. Tools can parse the specification to automatically generate client libraries (SDKs) in various programming languages, server stubs, and interactive documentation (e.g., Swagger UI). A standard OAS document consists of several key sections, as visualized in Figure A2.1:

- **Info:** Continues name, description and general information about the service.
- **Server & Security:** Information about the server location of the service and secutiry information how to access it.
- **Paths:** Defines the available endpoints (e.g., `/orders/id`).
- **Operations:** Describes the HTTP methods valid for a path and the expected behavior.
- **Tags & external docs**: Define tags for endpoints to group them and allows to append addidiotnal documentation of them.
- **Components:** A reusable section for storing security schemes, parameters, and most importantly, data schemas (defined using JSON Schema).

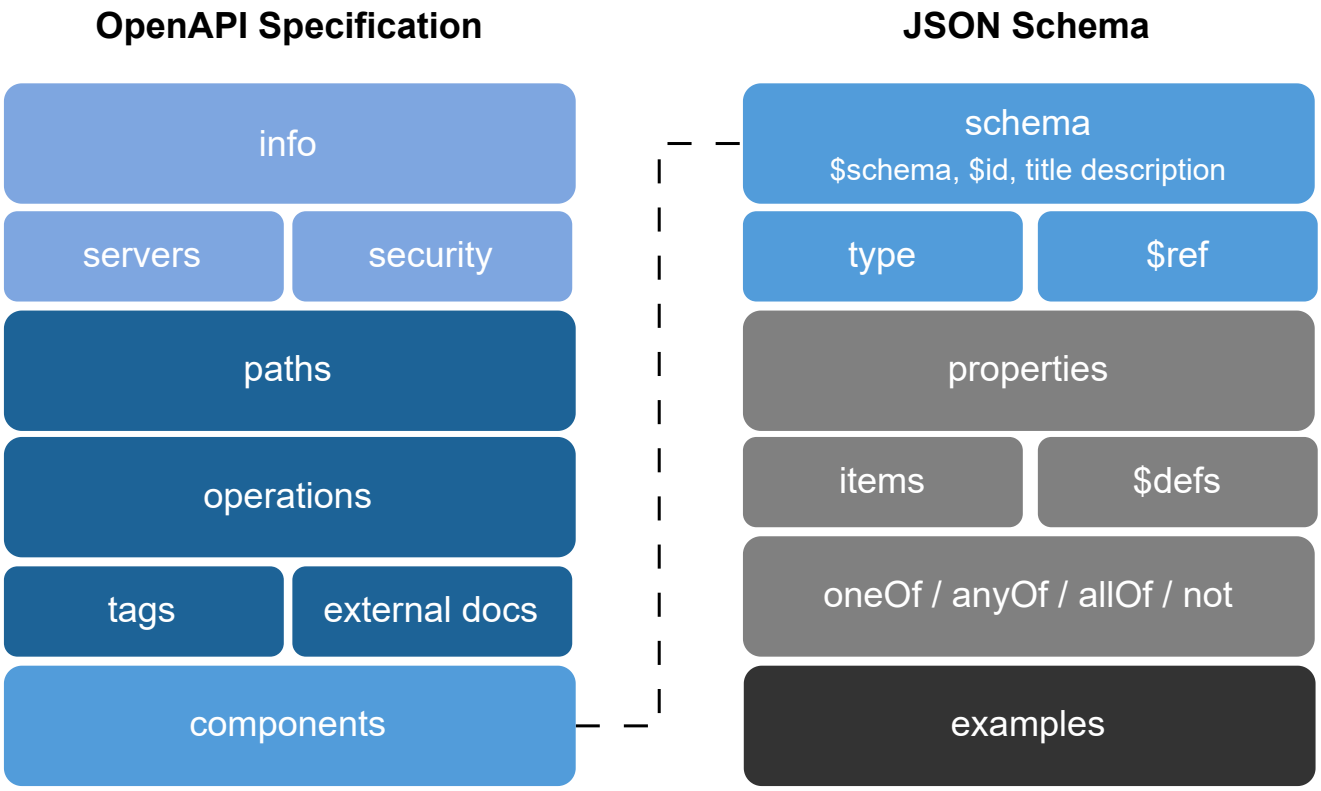

Figure A2.1: Conceptual overview of the structure of OAS and JSON Schema.

By strictly defining the input and output formats, OpenAPI reduces syntactic heterogeneity, although it cannot strictly enforce semantic consistency across different developers.

A2.2 JSON Schema

While OpenAPI describes the *behavior* of an API, JSON Schema provides the vocabulary to validate the *structure* of the data transmitted. It is a declarative language that allows for the annotation and validation of JSON documents.

A2.2.1 JSON Schema Components

JSON Schema provides a declarative vocabulary for describing the structure and constraints of JSON data. Figure A2.1 highlights the core building blocks:

Schema Metadata. Keywords such as `$schema`, `$id`, `title`, and `description` define meta-information about the schema itself. They specify the schema dialect, assign a global identifier, and provide human-readable documentation.

Type and References. The `type` keyword declares the expected JSON data type (e.g., `object`, `string`, `array`, `number`). The `$ref` keyword allows reusing or linking to other schema definitions, enabling modularity and avoiding duplication.

Properties. When `type=object`, the `properties` section declares the object's fields and their respective schemas. Additional constraints such as `required` can refine which fields must be present.

Items and `$defs`. For arrays, the `items` keyword specifies the schema of each element. The `$defs` section provides local reusable sub-schemas, which can be referenced via `$ref`.

Combinators. The keywords `oneOf`, `anyOf`, `allOf`, and `not` enable logical composition of schemas. They allow expressing alternatives, intersections, inheritance-like structures, or exclusions.

Examples. The `examples` keyword provides illustrative sample instances of valid data. Although optional and not used for validation, it improves documentation and helps developers understand the expected structure.

The following minimal JSON Schema demonstrates these components:

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://example.com/user.schema.json",
  "title": "User",
  "description": "A basic user profile",

  "type": "object",
  "properties": {
    "name": { "type": "string" },
    "age":  { "type": "integer" },
    "role": { "$ref": "#/$defs/roleEnum" }
  },
  "required": ["name"],

  "items": {},
  "$defs": {
    "roleEnum": {
      "oneOf": [
        { "const": "admin" },
        { "const": "user" }
      ]
    }
```

```
  },

  "examples": [
    { "name": "Alice", "age": 30, "role": "admin" }
  ]
}
```

A2.2.2 Validation and Constraints

JSON Schema enables the definition of strict rules for data formats. It verifies that a JSON object satisfies specific criteria before it is processed by an application. These criteria include:

- **Data Types:** Enforcing that a field is a string, number, boolean, array, or object.
- **Required Fields:** Specifying which properties must be present.
- **Constraints:** Defining patterns (Regular Expressions) for strings, minimum/maximum values for numbers, or item counts for arrays.

A2.2.3 Reusability and Composition

A powerful feature of JSON Schema is its ability to reference other schemas using the `$ref` keyword. This allows complex data models to be composed of smaller, modular definitions. For example, an `Address` schema can be defined once and referenced within both a `Customer` schema and a `Supplier` schema. This modularity is essential for managing complexity in large industrial systems but can also introduce challenges in integration when references are nested deeply or resolved externally.

A2.3 GraphQL

GraphQL represents a shift away from the fixed-endpoint structure of REST (and OpenAPI) toward a flexible, query-based approach. Originally developed by Facebook, it serves as both a query language for APIs and a runtime for fulfilling those queries with existing data.

A2.3.1 The GraphQL Schema Definition Language

Unlike REST, where the server dictates the structure of the response, GraphQL allows the client to request exactly the data it needs. This is governed by a strongly typed schema defined in the GraphQL Schema Definition Language (SDL). The SDL defines the "Graph" of the API:

- **Types:** Custom object types that represent the resources (e.g., `Machine`, `Order`).

- **Fields:** The specific attributes available on those types.
- **Queries and Mutations:** The entry points for fetching (Query) or modifying (Mutation) data.

A2.3.2 Introspection and Fetching

A key feature of GraphQL is *introspection*. A client can query the GraphQL schema itself to discover the available types and operations. In a traditional REST/OpenAPI scenario, fetching a user and their orders might require multiple round-trip requests (fetching the user, then fetching orders based on the user ID). In GraphQL, this is handled via a single request to a single endpoint. The client constructs a hierarchical query matching the structure of the desired response, effectively solving the problems of "over-fetching" (receiving too much data) and "under-fetching" (needing multiple requests) common in rigid REST architectures.

A3 Example for LLM-based schema integration

This appendix provides a representative example of an LLM prompt for automatic semantic integration, an automatically generated mapping description, the corresponding Python mapping function, and its unit test. The example illustrates how location messages from the Kinexon real-time tracking system are transformed into the location schema used by the Morpheus MES of the LFGP, as used in the integration benchmarking (Section 7.2). Table A3.1 summarizes the semantic correspondence between the two schemas.

Morpheus	**Kinexon**	
`id`	—	
`resourceId`	`kinexon_data.resource_id`	Resource identifier from the Kinexon payload
`resourceName`	`kinexon_data.object_name`	Human-readable name of the tracked resource
`gameRoundId`	`kinexon_data.game_round_id`	Associated game round identifier
`resourceType`	`kinexon_data.resource_type`	Type/category of the tracked resource
`validFrom`	`kinexon_data.kinexon_position.timestamp`	Timestamp at which the position was recorded
`validTo`	—	Always `None` (no mapping defined)
`x`	`kinexon_data.kinexon_position.x`	X-coordinate of the position
`y`	`kinexon_data.kinexon_position.y`	Y-coordinate of the position
`z`	`kinexon_data.kinexon_position.z`	Z-coordinate of the position
`latitude`	`kinexon_data.kinexon_position.latitude`	Geographical latitude (if available)
`longitude`	`kinexon_data.kinexon_position.longitude`	Geographical longitude (if available)

Table A3.1: Mapping of `input_instance` fields to the unified output structure.

A3.1 Instantiated Mapping prompt

Below is for the described mapping task the instantiated structured prompt for the LLM, using the prompt structure presented in Section 5.3.2l. The prompt is divided into its individual components and some minimal formatting has been applied to improve readability here. The previous function prompt component is not specified, since no previous function code was considered for this example.

Role:

```
You are a schema mapping scientist.
```

Task:

```
Generate a Python function that maps data from an input model to an output
    model. Please only return the mapping_function and nothing else, make
   sure the returned value can be parsed in python interpreter without
   syntax errors.

IMPORTANT: All necessary import statements must be placed INSIDE the
   function, not outside. This ensures the function is self-contained and
   can be executed in any context.

Use the following template:

def mapping_function(input_instance: dict) -> dict:
    """
    Maps an input instance to an output instance based on the specified
       rules.

    Parameters:
        input_instance (dict): Input data conforming to the input model
           schema.

    Returns:
        dict: Output data conforming to the output model schema.
    """
    # Place ALL necessary imports here at the beginning of the function
    # Example imports (only include what you actually need):
    # from datetime import datetime, date
    # import json
    # import re
    # from typing import Any, Optional
    # from decimal import Decimal

    output_instance = dict()
    { mapping_logic }
    return output_instance
```

```
Both models are defined as JSON Schema objects. Write the mapping function
    for instances of these schemas.
```

Schemas (with contained documentation strings and examples in the schema definition):

```
Both models are defined as JSON Schema objects. Write the mapping function
    for instances of these schemas.

Input model schema:
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Union_KinexonData",
    "type": "object",
    "properties": {
       "kinexon_data": {
          "examples": [
             {
                "game_round_id": "68415666436a3e760c41c492",
                "kinexon_position": {
                   "latitude": 48.1508,
                   "longitude": 11.5748,
                   "object_id": "tag_1008",
                   "timestamp": "2025-02-19T10:37:39.479Z",
                   "x": 24.995,
                   "y": 35.7971,
                   "z": 1.6
                },
                "object_name": "Worker8",
                "resource_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
                "resource_type": "WORKER"
             }
          ],
          "properties": {
             "object_name": {
                "description": "The name of the object.",
                "example": "Worker8",
                "title": "Object Name",
```

```
                "type": "string"
            },
            "resource_id": {
                "description": "The unique identifier of the resource.",
                "example": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
                "title": "Resource Id",
                "type": "string"
            },
            "resource_type": {
                "$ref": "#/$defs/LocationResourceType",
                "description": "The type of the resource.",
                "example": "WORKER"
            },
            "game_round_id": {
                "description": "The unique identifier of the game round
                    .",
                "example": "68415666436a3e760c41c492",
                "title": "Game Round Id",
                "type": "string"
            },
            "kinexon_position": {
                "$ref": "#/$defs/KinexonPosition",
                "description": "The position data from Kinexon."
            }
        },
        "required": [
            "object_name",
            "resource_id",
            "resource_type",
            "game_round_id",
            "kinexon_position"
        ],
        "title": "KinexonData",
        "type": "object"
    }
},
"required": [
```

```
        "kinexon_data"
    ],
    "$defs": {
        "KinexonPosition": {
            "description": "Models a single position update from the
               Kinexon system.",
            "examples": [
                {
                    "latitude": 48.1508,
                    "longitude": 11.5748,
                    "object_id": "tag_1008",
                    "timestamp": "2025-02-19T10:37:39.479Z",
                    "x": 24.995,
                    "y": 35.7971,
                    "z": 1.6
                }
            ],
            "properties": {
                "object_id": {
                    "description": "The unique identifier of the tracked
                       object in Kinexon.",
                    "example": "tag_1008",
                    "title": "Object Id",
                    "type": "string"
                },
                "x": {
                    "description": "The x-coordinate of the object.",
                    "example": 24.995,
                    "title": "X",
                    "type": "number"
                },
                "y": {
                    "description": "The y-coordinate of the object.",
                    "example": 35.7971,
                    "title": "Y",
                    "type": "number"
                },
```

```
        "z": {
            "description": "The z-coordinate of the object.",
            "example": 1.6,
            "title": "Z",
            "type": "number"
        },
        "latitude": {
            "description": "The latitude of the object.",
            "example": 48.1508,
            "title": "Latitude",
            "type": "number"
        },
        "longitude": {
            "description": "The longitude of the object.",
            "example": 11.5748,
            "title": "Longitude",
            "type": "number"
        },
        "timestamp": {
            "description": "The ISO 8601 formatted timestamp of the
               position update.",
            "example": "2025-02-19T10:37:39.479Z",
            "title": "Timestamp",
            "type": "string"
        }
    },
    "required": [
        "object_id",
        "x",
        "y",
        "z",
        "latitude",
        "longitude",
        "timestamp"
    ],
    "title": "KinexonPosition",
    "type": "object"
```

```
        },
        "LocationResourceType": {
            "enum": [
                "SHELF",
                "WORKER",
                "WORKSTATION"
            ],
            "title": "LocationResourceType",
            "type": "string"
        }
    }
}

Output model schema:

{
    "$defs": {
        "LocationResourceType": {
            "enum": [
                "SHELF",
                "WORKER",
                "WORKSTATION"
            ],
            "title": "LocationResourceType",
            "type": "string"
        }
    },
    "description": "Represents a location in the system.
        Args:
            id (Optional[str]): Unique identifier for the location.
            resourceId (Optional[str]): Identifier for the resource
               associated with this location.
            resourceName (Optional[str]): Name of the resource associated
               with this location.
            gameRoundId (Optional[str]): Identifier for the game round
               associated with this location.
```

```
        validFrom (Optional[datetime]): Start time for the validity of
           this location.
        validTo (Optional[datetime]): End time for the validity of this
            location.
        x (Optional[float]): X-coordinate of the location in a 3D space.

        y (Optional[float]): Y-coordinate of the location in a 3D space.

        z (Optional[float]): Z-coordinate of the location in a 3D space.

        latitude (Optional[float]): Latitude of the location, used for
           geographical positioning.
        longitude (Optional[float]): Longitude of the location, used
           for geographical positioning.",
"examples": [
    {
        "gameRoundId": "685fa6c3685a274874411c5e",
        "id": "685fa6d3685a274874412788",
        "latitude": 48.1508,
        "longitude": 11.5746,
        "resourceId": "67a32880a68e602891fc17cf",
        "resourceName": "Joiner1 (Manual)",
        "resourceType": "WORKSTATION",
        "validFrom": "2025-06-28T08:23:54.460000Z",
        "validTo": "2025-06-28T08:24:39.881000Z",
        "x": 13.8189,
        "y": 45.3753,
        "z": 1.6
    },
    {
        "gameRoundId": "685fa6c3685a274874411c5e",
        "id": "685fa6d3685a274874412789",
        "latitude": 48.1508,
        "longitude": 11.5746,
        "resourceId": "67a32880a68e602891fc17c7",
        "resourceName": "Press1 (Manual)",
        "resourceType": "WORKSTATION",
```

```
            "validFrom": "2025-06-28T08:24:23.775000Z",
            "validTo": "2025-06-28T08:25:09.263000Z",
            "x": 9.1023,
            "y": 45.5366,
            "z": 1.6
        }
    ],
    "properties": {
        "id": {
            "anyOf": [
                {
                    "type": "string"
                },
                {
                    "type": "null"
                }
            ],
            "default": null,
            "title": "Id"
        },
        "resourceId": {
            "anyOf": [
                {
                    "type": "string"
                },
                {
                    "type": "null"
                }
            ],
            "default": null,
            "title": "Resourceid"
        },
        "resourceName": {
            "anyOf": [
                {
                    "type": "string"
                },
```

```
            {
                "type": "null"
            }
        ],
        "default": null,
        "title": "Resourcename"
    },
    "gameRoundId": {
        "anyOf": [
            {
                "type": "string"
            },
            {
                "type": "null"
            }
        ],
        "default": null,
        "title": "Gameroundid"
    },
    "resourceType": {
        "anyOf": [
            {
                "$ref": "#/$defs/LocationResourceType"
            },
            {
                "type": "null"
            }
        ],
        "default": null
    },
    "validFrom": {
        "anyOf": [
            {
                "format": "date-time",
                "type": "string"
            },
            {
```

```
            "type": "null"
        }
    ],
    "default": null,
    "title": "Validfrom"
},
"validTo": {
    "anyOf": [
        {
            "format": "date-time",
            "type": "string"
        },
        {
            "type": "null"
        }
    ],
    "default": null,
    "title": "Validto"
},
"x": {
    "anyOf": [
        {
            "type": "number"
        },
        {
            "type": "null"
        }
    ],
    "default": null,
    "title": "X"
},
"y": {
    "anyOf": [
        {
            "type": "number"
        },
        {
```

```
            "type": "null"
        }
    ],
    "default": null,
    "title": "Y"
},
"z": {
    "anyOf": [
        {
            "type": "number"
        },
        {
            "type": "null"
        }
    ],
    "default": null,
    "title": "Z"
},
"latitude": {
    "anyOf": [
        {
            "type": "number"
        },
        {
            "type": "null"
        }
    ],
    "default": null,
    "title": "Latitude"
},
"longitude": {
    "anyOf": [
        {
            "type": "number"
        },
        {
            "type": "null"
```

```
                }
            ],
            "default": null,
            "title": "Longitude"
        }
    },
    "title": "Location",
    "type": "object"
}
```

Additional Rules:

```
For the mapping, consider the following additional rules in the mapping:
- "Map 'x', 'y', 'z', 'latitude', 'longitude' directly from Kinexon
   position."
- "Map Kinexon 'timestamp' to Morpheus 'validFrom'."
- "The 'resourceName' is derived from the Kinexon object name."
- "The 'resource_id' is derived from a mapping of workstation names to IDs
    or by formatting the resource name."
- "The 'resourceType' is determined based on the Kinexon resource type."
- "The 'gameRoundId' is derived from the specified game round ID."
```

System Policies:

```
Remember:
1. All imports must be inside the function
2. Only import what you actually use
3. The function must be completely self-contained
4. Test that your mapping logic handles edge cases (None values, missing
   fields, etc.).
5. Avoid TypeError: 'NoneType' object is not iterable or accessing a None
   Type value with a key.
6. Leave ids unchanged. If fields like id_short or similar exist besides
   id fields, make the value similar to that of the id field.
7. Set all optional field's values (also optionals) to None if they are
   not present in the input instance.
```

A3.2 Generated Mapping Function

Below is the LLM-generated mapping function of the Kinexon to Morpheus Location mapping that implements the semantic integration, as depicted in Table A3.1.

```
def mapping_function(input_instance: dict) -> dict:
    """
␣␣␣␣Maps␣an␣input␣instance␣to␣an␣output␣instance␣based␣on␣the␣specified
␣␣␣␣rules.

␣␣␣␣Parameters:
␣␣␣␣␣␣␣␣input_instance␣(dict):␣Input␣data␣conforming␣to␣the␣input
␣␣␣␣␣␣␣␣␣␣␣␣model␣schema.

␣␣␣␣Returns:
␣␣␣␣␣␣␣␣dict:␣Output␣data␣conforming␣to␣the␣output␣model␣schema.
␣␣␣␣"""
    # Place ALL necessary imports here at the beginning of
    # the function
    No imports are needed for this mapping.

    output_instance = {
        'id': None,
        'resourceId': None,
        'resourceName': None,
        'gameRoundId': None,
        'resourceType': None,
        'validFrom': None,
        'validTo': None,
        'x': None,
        'y': None,
        'z': None,
        'latitude': None,
        'longitude': None
    }

    if not isinstance(input_instance, dict):
        return output_instance
```

```
    kinexon_data = input_instance.get('kinexon_data')

    if isinstance(kinexon_data, dict):
        # Map fields from the kinexon_data object
        output_instance['resourceName'] = kinexon_data.get(
            'object_name')
        output_instance['resourceId'] = kinexon_data.get(
            'resource_id')
        output_instance['resourceType'] = kinexon_data.get(
            'resource_type')
        output_instance['gameRoundId'] = kinexon_data.get(
            'game_round_id')

        kinexon_position = kinexon_data.get('kinexon_position')
        if isinstance(kinexon_position, dict):
            # Map fields from the nested kinexon_position object
            output_instance['x'] = kinexon_position.get('x')
            output_instance['y'] = kinexon_position.get('y')
            output_instance['z'] = kinexon_position.get('z')
            output_instance['latitude'] = kinexon_position.get(
                'latitude')
            output_instance['longitude'] = kinexon_position.get(
                'longitude')
            output_instance['validFrom'] = kinexon_position.get(
            'timestamp')

    return output_instance
```

A3.3 Example for generated mapping description

Below is the generated human readable markdown documentation of the mapping function, which can be used during review of the mapping result for experts to get a faster overview.

```
# Purpose

The mapping function takes a single input dictionary containing Kinexon
tracking data and converts it into a standardized resource location object.

It extracts identification, type, game round information, and position
```

```
(both Cartesian and geographic) and writes them into a fixed output schema.


## Input Expectations

- The function expects ‘input_instance‘ to be a Python dictionary.
- It looks for a key called ‘kinexon_data‘ inside this dictionary.
- If ‘input_instance‘ is not a dictionary, or if ‘kinexon_data‘ is missing
  or not a dictionary, the function returns an output object where all
  fields are set to ‘None‘.

## Output Structure

The function always returns a dictionary with the following keys:

- ‘id‘
- ‘resourceId‘
- ‘resourceName‘
- ‘gameRoundId‘
- ‘resourceType‘
- ‘validFrom‘
- ‘validTo‘
- ‘x‘
- ‘y‘
- ‘z‘
- ‘latitude‘
- ‘longitude‘

All fields are initialized to ‘None‘ and then selectively filled from the
input data.

## Field Mapping Logic

### Top-level Kinexon metadata

From ‘input_instance["kinexon_data"]‘, the following mappings are applied:
```

```
- ‘resourceName‘ <- ‘kinexon_data["object_name"]‘
- ‘resourceId‘ <- ‘kinexon_data["resource_id"]‘
- ‘resourceType‘ <- ‘kinexon_data["resource_type"]‘
- ‘gameRoundId‘ <- ‘kinexon_data["game_round_id"]‘

### Position and coordinates

If ‘kinexon_data["kinexon_position"]‘ exists and is a dictionary, then:

- ‘x‘ <- ‘kinexon_position["x"]‘
- ‘y‘ <- ‘kinexon_position["y"]‘
- ‘z‘ <- ‘kinexon_position["z"]‘
- ‘latitude‘ <- ‘kinexon_position["latitude"]‘
- ‘longitude‘ <- ‘kinexon_position["longitude"]‘
- ‘validFrom‘ <- ‘kinexon_position["timestamp"]‘

The field ‘validTo‘ is not filled and remains ‘None‘.

## Error Handling and Robustness

- If the input is not a dictionary, the function returns an output object
  with all fields set to ‘None‘.
- Missing keys are handled via ‘.get(...)‘, leaving the corresponding
   fields
  as ‘None‘.
- No exceptions are raised; the function always returns a valid output
  structure.

## Summary

The mapping function:

- Normalizes Kinexon tracking events into a fixed resource location schema.

- Extracts identifiers and type information.
- Extracts spatial coordinates and timestamps.
- Handles incomplete or malformed input safely.
```

A3.4 Example for generated mapping function unit test

Below is the automatically generated unit test for the mapping function that validates correct mappings. In this example, no example instances were provided but the LLM generated synthetic test data.

```
import pytest
from mapping import mapping_function


def test_mapping_with_full_kinexon_data():
    input_instance = {
        "kinexon_data": {
            "object_name": "Worker A",
            "resource_id": "res-123",
            "resource_type": "worker",
            "game_round_id": "round-1",
            "kinexon_position": {
                "x": 1.23,
                "y": 4.56,
                "z": 7.89,
                "latitude": 48.12345,
                "longitude": 8.54321,
                "timestamp": "2025-11-29T12:34:56Z",
            },
        }
    }

    result = mapping_function(input_instance)

    assert result["resourceName"] == "Worker A"
    assert result["resourceId"] == "res-123"
    assert result["resourceType"] == "worker"
    assert result["gameRoundId"] == "round-1"

    assert result["x"] == 1.23
    assert result["y"] == 4.56
    assert result["z"] == 7.89
    assert result["latitude"] == 48.12345
```

```
assert result["longitude"] == 8.54321
assert result["validFrom"] == "2025-11-29T12:34:56Z"

# Fields that are not set by the function should stay None
assert result["id"] is None
assert result["validTo"] is None
```

A4 Middleware API interfaces

The Middleware relies on three standardized interfaces that clearly separate concerns within the Integration Layer, as visualized in Figure A4.1.

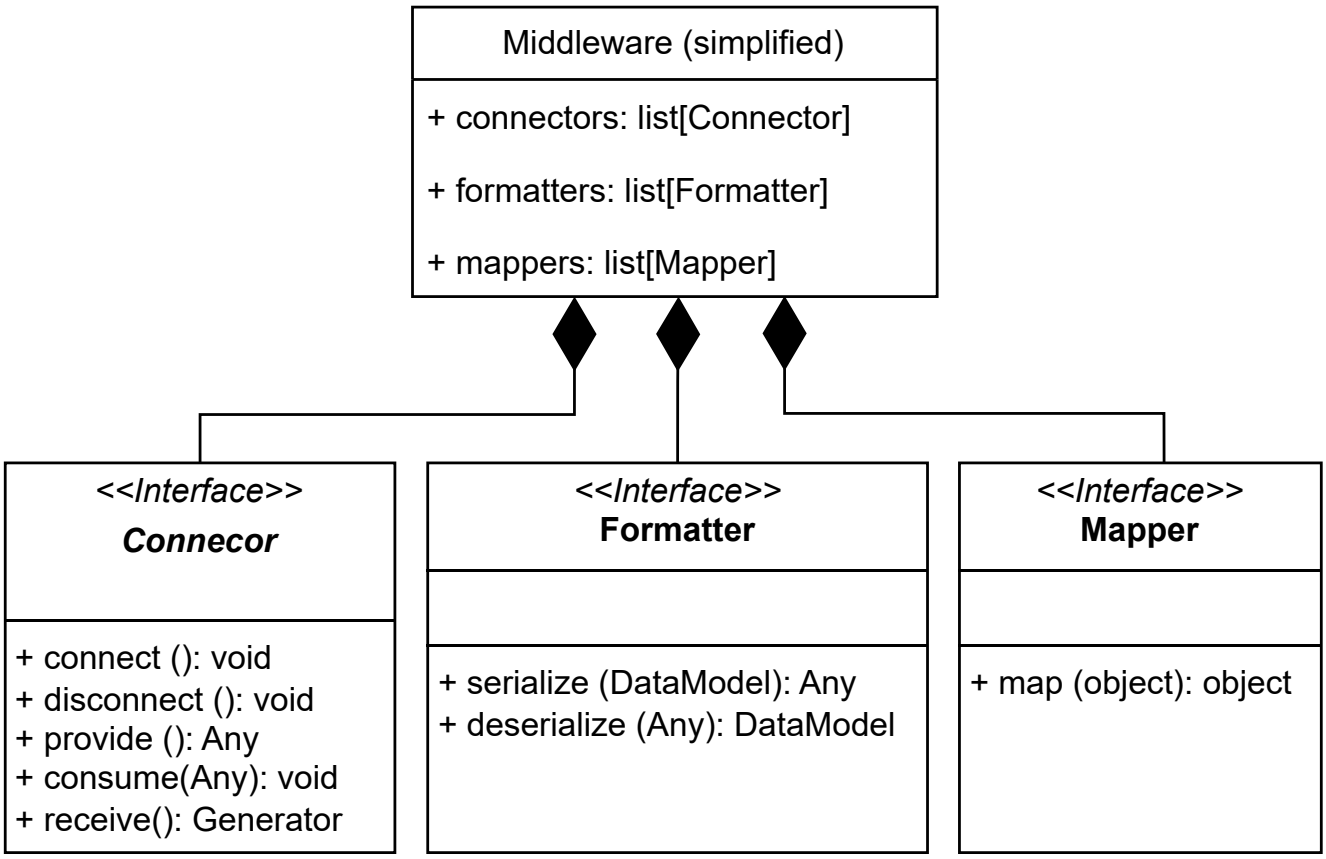

Figure A4.1: UML class diagram depicting the standardized interfaces of Connectors, Formatters and Mappers in the Middleware.

Connectors abstract communication with external systems by providing a uniform API for establishing and terminating connections, sending data (`provide()`), consuming incoming data (`consume()`), and iterating over event streams (`receive()`). Note, that only one of the three functions must be provided, depending on the use of the Connector.

Formatters resolve syntactic heterogeneity by defining methods for serializing internal Datamodels into external representations and deserializing arbitrary incoming data into structured DataModel objects.

Mappers address semantic heterogeneity by transforming objects between different schema definitions through a single `map()` operation. Input and Output of the function can be simple key value stores but also DataModels.

By enforcing these minimal yet expressive interfaces, the Middleware achieves a high degree of modularity: components can be substituted, extended, or auto-generated while guaranteeing consistent behavior across all services.

A5 Schema Integration Service User Interface

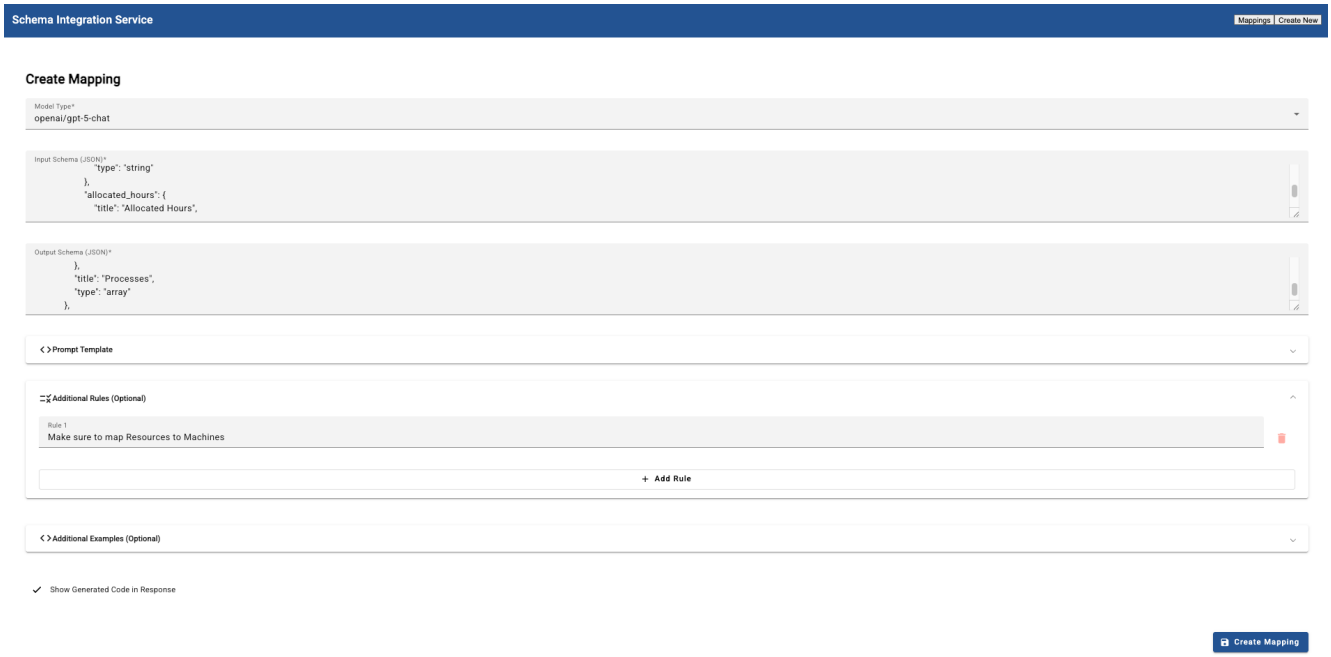

Figure A5.1: Creation view of a mapping request in the SIS user interface, where the components of the structured prompt are specified.

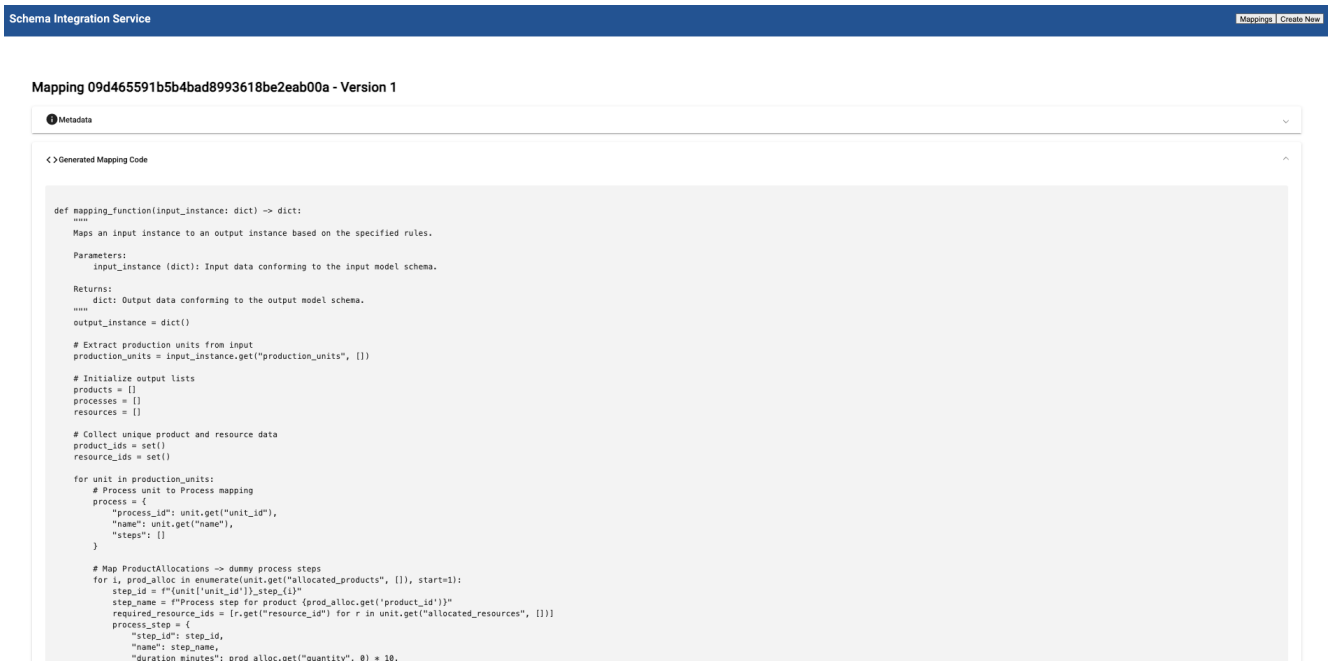

Figure A5.2: Result view of a mapping request in the SIS user interface, showing the generated mapping function and its executable code.

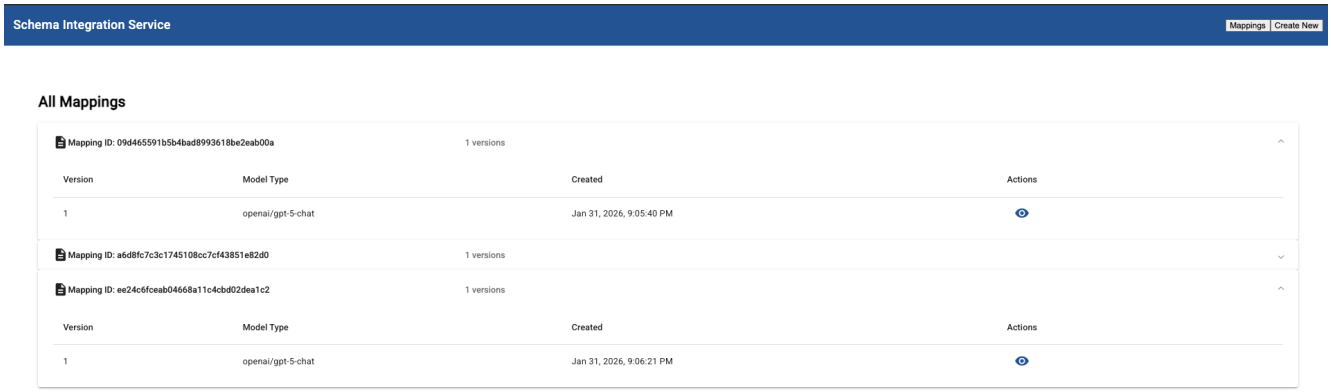

Figure A5.3: Overview view of the SIS user interface, listing created mapping functions together with their corresponding versions.

A6 Orchestration Engine User Interface

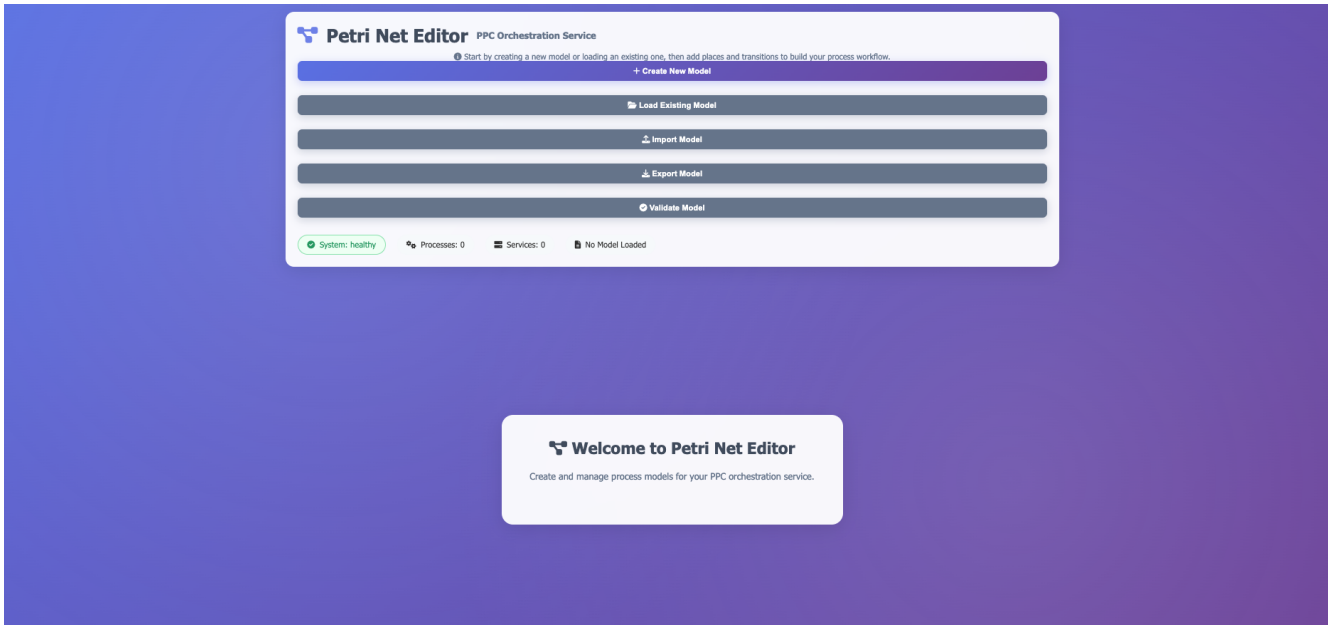

Figure A6.1: Overview page of the OE UI, providing access to core functions including SAPN model creation, loading, import/export, and well-typedness validation.

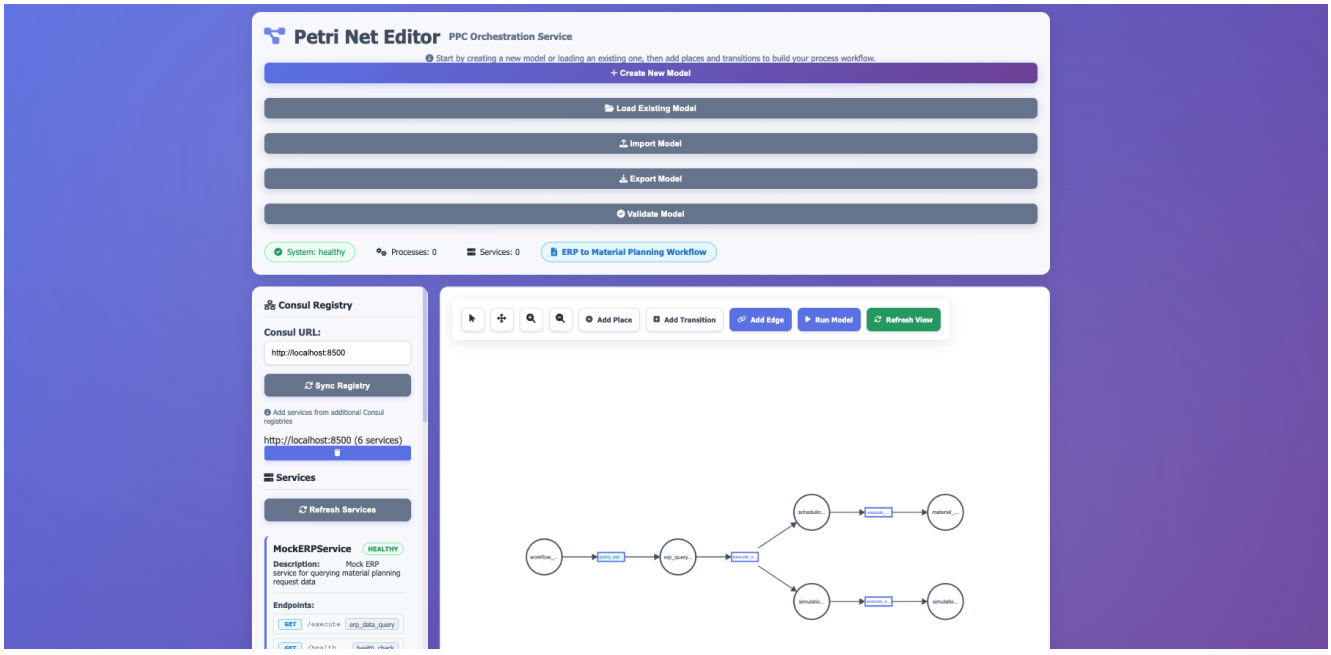

Figure A6.2: Graph-based visualization of a loaded SAPN model in the OE UI, allowing interactive adjustment of places and transitions; available services retrieved from the Service Registry are shown in the left sidebar.

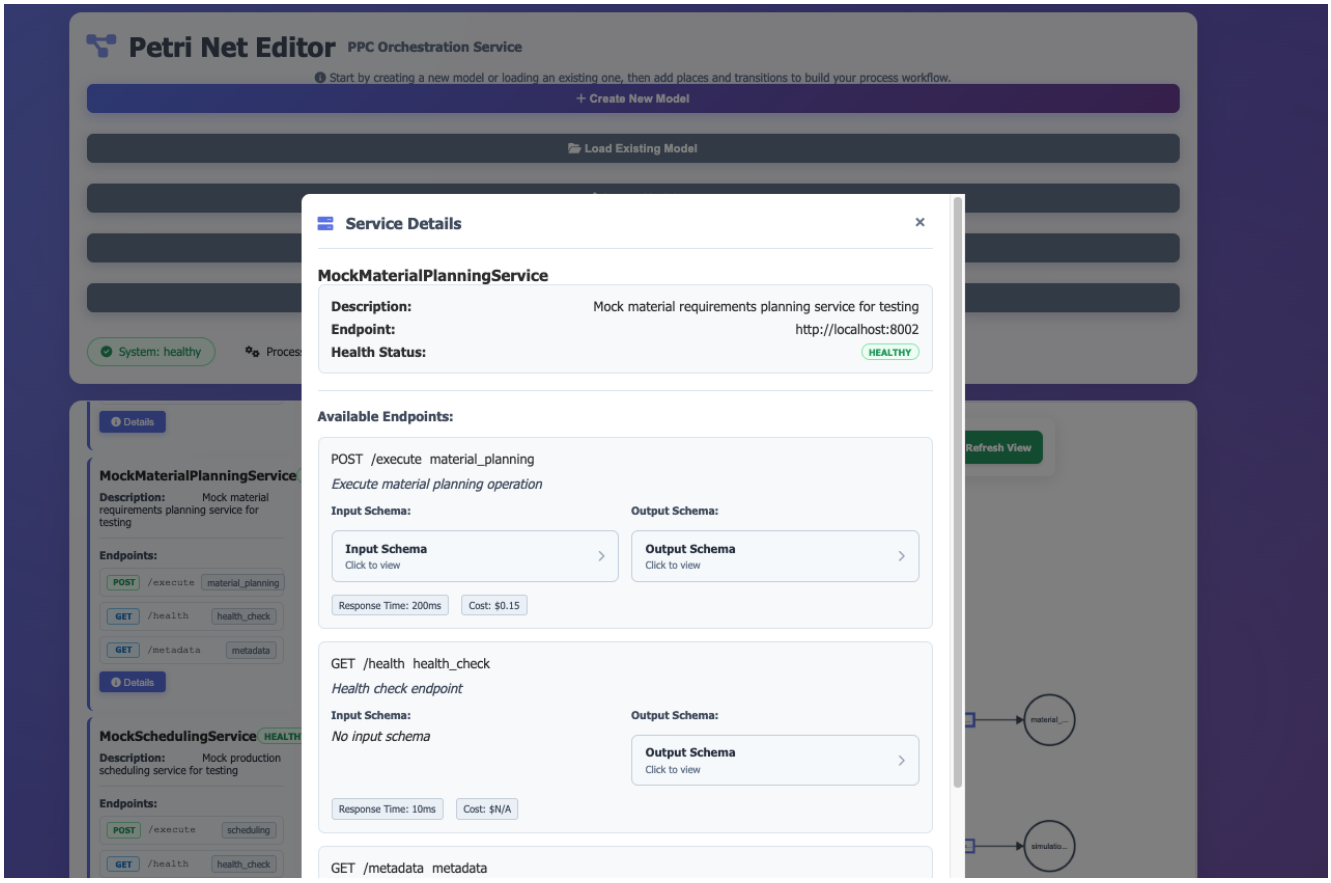

Figure A6.3: Service detail view in the OE UI, displaying metadata retrieved from the Service Registry, including service description, capabilities, input/output schemas, health status, and deployment location.

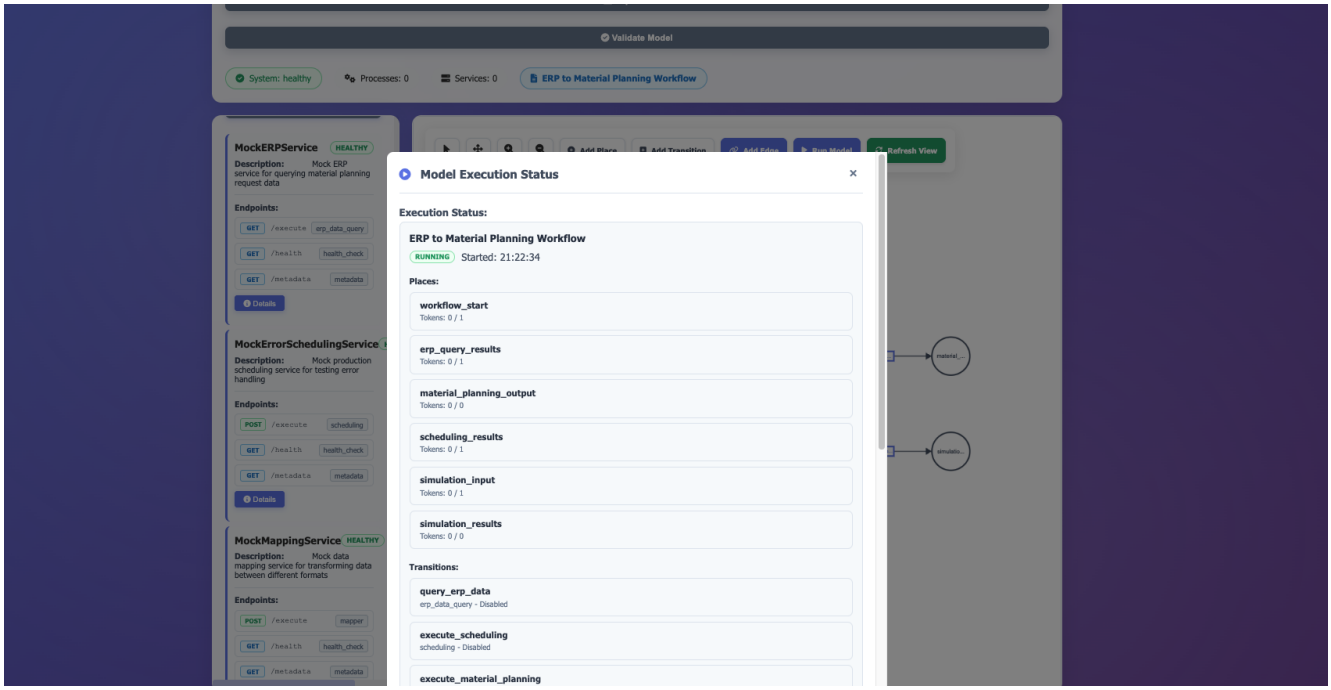

Figure A6.4: Execution view of an SAPN process model in the OE UI, showing live execution status and current markings of places and transitions.

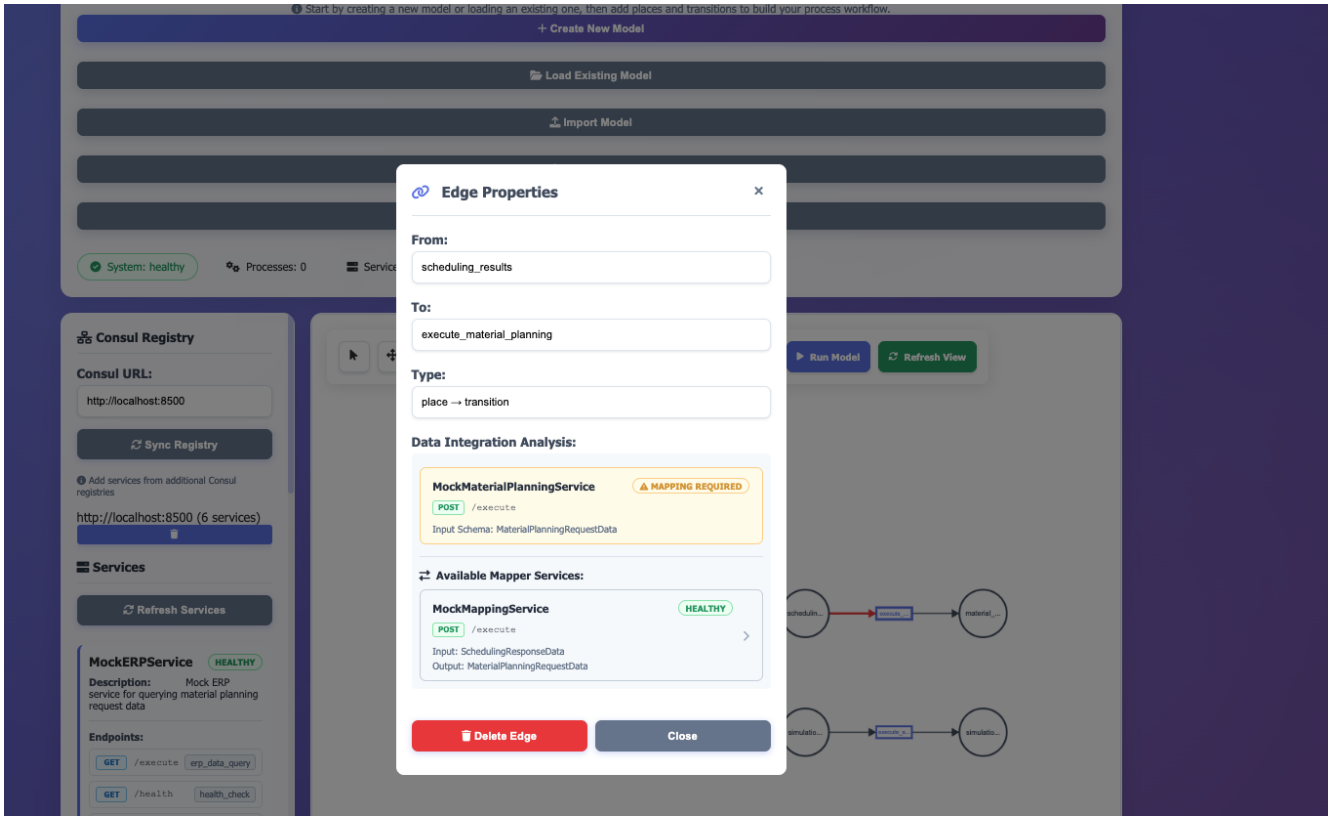

Figure A6.5: Semantic compatibility view in the OE UI, indicating whether required semantic mappings for available services are necessary and available.

A7 Data Integration Benchmark Results

A7.1 Data Integration Benchmark Schemas

This section provides a structured and detailed description of the data schemas used in the automated data integration benchmark introduced in Section 7.2. The purpose of this appendix is to document the semantic scope, structural characteristics, and intended usage context of each schema, thereby establishing a transparent foundation for the definition and evaluation of the concrete mapping tasks described subsequently.

The selected schemas originate from heterogeneous production IT systems and services used in industrial practice and applied research. Together, they cover a broad spectrum of Production Planning and Control (PPC) concerns, including production execution, simulation and optimization, layout and intralogistics planning, real-time location tracking, operator guidance, and production analytics. This diversity ensures that the benchmark reflects realistic integration challenges encountered in service-oriented PPC environments.

A7.1.1 Overview of Benchmark Schemas

The benchmark comprises seven schemas, which are briefly summarized in Table A7.1. A detailed description of each schema is provided in the following subsections. Rather than providing exhaustive schema specifications, the following descriptions highlight those fields and structural elements that are particularly relevant for semantic mapping and automated integration tasks.

A7.1.2 SDM Reference Model

The SDM Reference Model (Ellwein; Neumann et al. 2023) serves as the semantic backbone of the benchmark and is designed as a system-independent, holistic production data model. It captures core PPC concepts and provides stable identifiers and abstractions for cross-system integration.

Integration-relevant entities and fields include:

- **Resource**: uniquely identified via semantic identifiers (e.g., `id_short`) and structured hierarchically using explicit parent-child relations. Relevant fields include `production_level`, `resource_type`, and `capabilities`.
- **Process**: abstract description of manufacturing steps, characterized by `capability` and `manufacturing_type`, enabling semantic matching across systems.

Table A7.1: Schemas Used in the Data Integration Benchmark

Schema	Scope and Role in the Benchmark
SDM Reference Model	Holistic, schema-driven production data model serving as a global integration schema and semantic anchor for cross-system mappings.
prodsys	Simulation-oriented production system model used for material flow simulation, optimization, and performance analysis.
Morpheus	Manufacturing Execution System (MES) schema used for production tracking, order management, and execution monitoring in the LFGP.
Intralogistic Map Format	Layout and facility description schema for intralogistics and autonomous transport route planning, based on the Layout Interchange Format (LIF).
Datenberg	Flattened association schema used for production analytics and dashboard visualization.
Kinexon	Real-time location tracking schema for workers, assets, and shopfloor equipment.
Tulip	Lightweight MES schema for Pick-by-Light order visualization and operator guidance.

- **Procedure**: executable realization of a process, including temporal execution models (e.g., stochastic or deterministic time distributions).
- **Product**: product definitions with references to required processes and optional bill-of-material structures.
- **Order** and **Event**: entities providing temporal context, order states, and execution traces.
- **Location**: spatial information associated with resources, enabling alignment with layout and tracking systems.

Due to its explicit separation of abstract processes and executable procedures, the SDM Reference Model is particularly suited as a pivot schema for both upstream (e.g., MES) and downstream (e.g., simulation or layout) integrations.

A7.1.3 prodsys Simulation Schema

The prodsys schema (Behrendt; Danz et al. 2025) is optimized for discrete-event simulation and production system optimization. Its structure emphasizes explicit modeling of system dynamics and stochastic behavior.

Key integration-relevant fields include:

- **ResourceData**: fields such as `process_ids`, `state_ids`, `input_queues`, and `output_queues`, defining resource capabilities and routing logic.
- **ProcessData**: characterized by `capability`, `time_model_id`, and optional failure parameters.
- **TimeModelData**: specification of temporal behavior via distribution type and parameters (e.g., mean, variance).
- **ProductData**: ordered lists of required processes, defining the production routing.
- **QueueData**: buffer capacities and sequencing policies.

Integration tasks involving prodsys frequently require aggregating execution-level data into simulation-ready abstractions and translating temporal information into compatible time model representations.

A7.1.4 Morpheus MES Schema

Morpheus is the Manufacturing Execution System used in the Learning Factory for Green Production (LFGP). Its schema captures operational production data, including workstations, work processes, production lines, orders, product variants, individual product instances, and execution events.

A distinguishing characteristic of the Morpheus schema is its event-centric design, which records detailed timestamps for production steps, resource assignments, and order states. Additionally, the schema incorporates contextual entities such as training sessions and game rounds, which are specific to the learning factory environment.

Within the benchmark, Morpheus serves as a representative MES schema. Mapping tasks involving Morpheus typically focus on transforming execution-level data into planning, simulation, analytics, or visualization-oriented schemas.

A7.1.5 Kinexon Real-Time Location Schema

The Kinexon schema represents data provided by a real-time location system (RTLS) deployed on the shopfloor. It describes tracked objects such as workers, workstations, or shelves, together with their spatial coordinates and timestamps.

The schema supports both local coordinate systems (factory-relative (x, y, z) and global coordinates (latitude and longitude). Location updates are time-stamped and associated with object identifiers and semantic resource types.

In the benchmark, the Kinexon schema is used to assess integration tasks involving dynamic, high-frequency sensor data and its alignment with static or event-based production data models, such as MES or reference schemas.

A7.1.6 Intralogistic Map Format

The Intralogistic Map Format is a specialized schema for describing factory layouts and intralogistics-relevant structures. It is based on the Layout Interchange Format (LIF) (Verband Deutscher Maschinen- und Anlagenbau e. V. (VDMA) 2024) and includes entities such as maps, stations, obstacles, trajectories, and geometric boundaries.

The schema is designed for applications such as autonomous guided vehicle (AGV) navigation, route planning, and layout-based simulation. Stations represent interaction points for transport processes, while obstacles and boundaries encode physical constraints of the shopfloor.

In the benchmark, this schema is used to evaluate mappings that derive spatial layout information from production system models, for example by transforming resource hierarchies and location data into a navigable facility map.

A7.1.7 Datenberg Dashboard Schema

The Datenberg schema is a flattened, association-based data model designed for dashboard visualization and analytics. Instead of deeply nested structures, it emphasizes explicit relationships between entities such as workstations, production lines, work processes, product variants, and events.

This denormalized structure reflects the requirements of business intelligence and visualization tools, which favor simple, joinable datasets over highly structured domain models.

Mapping tasks involving the Datenberg schema typically assess the ability to aggregate, flatten, and contextualize production data originating from MES or reference models for analytical consumption.

A7.1.8 Tulip Pick-by-Light Schema

The Tulip schema supports integration with an app-based MES used for Pick-by-Light applications. It consists of lightweight table-oriented data structures that encode order information relevant for operator guidance.

Field naming conventions and table layouts are tailored to the configuration requirements of the Tulip platform. As a result, mappings to this schema often require selective data extraction, renaming, and structural simplification.

In the benchmark, the Tulip schema represents downstream, human-facing systems with strict structural constraints and limited semantic scope.

A7.1.9 Summary

Together, the described schemas form a heterogeneous and representative benchmark set for evaluating automated data integration in PPC contexts. They differ significantly in abstraction level, structural complexity, semantic focus, and intended use, ranging from execution-oriented MES data and real-time sensor streams to simulation models, layout descriptions, and dashboard-oriented views.

This diversity enables a systematic assessment of the proposed LLM-supported integration approach across a wide spectrum of realistic integration scenarios. The concrete mapping tasks derived from these schemas are defined and analyzed in the subsequent sections of this appendix.

A7.2 Data Integration Benchmark Tasks

This section documents the complete set of mapping tasks used in the automated data integration benchmark described in Section 7.2. The benchmark comprises **41 individual mapping tasks** that evaluate the ability of LLMs to synthesize correct transformation functions between heterogeneous production data schemas. The tasks were derived from realistic integration needs in PPC environments, including execution-to-simulation transformations, MES-to-dashboard exports, layout derivation, and location tracking integration.

A7.2.1 Benchmark Summary and Task Distribution

Task counts and difficulty distribution. The benchmark includes a total of 41 tasks, grouped into three difficulty classes: **8 easy** (20%), **15 medium** (36%), and **18 hard** (44%). The difficulty assignment follows the qualitative criteria described in Section A7.2.12 and was guided by the manual effort required to implement the corresponding gold-standard mappers.

Source and target schema distributions. Most tasks originate from the MES-centered Morpheus schema (25 tasks), reflecting typical integration demands where execution data constitutes the primary source. Target schemas are dominated by the SDM Reference Model (22 tasks), which acts as the central interoperability schema for the benchmark. A summarized distribution is provided in Table A7.2.

Table A7.2: Summary of Benchmark Task Distributions

Source Schema		Target Schema	
MorpheusModel	25	ReferenceModel	22
ReferenceModel	10	prodsys	10
prodsysModel	4	DatenbergModel	5
KinexonModel	2	MorpheusModel	1
		TulipTableModel	1
		IntralogisticMapFormat	1

A7.2.2 Task Categories

To support structured analysis and interpretability, the 41 tasks are organized into **eight mapping categories** according to their source and target schema pairs:

1. Kinexon → Morpheus (1 task)
2. Kinexon → ReferenceModel (1 task)
3. Morpheus → Datenberg (5 tasks)
4. Morpheus → ReferenceModel (18 tasks)
5. Morpheus → Tulip (1 task)
6. prodsys → ReferenceModel (4 tasks)
7. ReferenceModel → IntralogisticMapFormat (1 task)
8. ReferenceModel → prodsys (10 tasks)

The following subsections describe each category and summarize the contained tasks, including their objective, input/output types, and characteristic transformation patterns.

A7.2.3 Kinexon → Morpheus (1 task)

Task 1: Kinexon to Morpheus Location Mapping (easy). This task maps real-time position updates from the Kinexon RTLS schema (`KinexonData`) to the Morpheus MES location representation (`Location`). It represents a direct integration of sensor data into MES context.

Key transformations:

- Identity and metadata: `object_name` → `resourceName`, `resource_id` → `resourceId`, `resource_type` → `resourceType`

- Context: `game_round_id` → `gameRoundId`
- Coordinates:
  - `kinexon_position.x` → `x`
  - `kinexon_position.y` → `y`
  - `kinexon_position.z` → `z`
  - `kinexon_position.latitude` → `latitude`
  - `kinexon_position.longitude` → `longitude`
- Time: `kinexon_position.timestamp` → `validFrom`

A7.2.4 Kinexon → ReferenceModel (1 task)

Task 2: Kinexon to ReferenceModel Location Mapping (easy). This task transforms `KinexonData` into a standardized `Location` entity of the ReferenceModel. In contrast to Task 1, it requires alignment with a canonical identifier style and the semantics of the reference schema.

Key transformations:

- Generation of a canonical identifier (e.g., `id_short`)
- Mapping of local and global coordinates and the timestamp field
- Derivation of resource identity attributes (name/id) from Kinexon metadata

A7.2.5 Morpheus → Datenberg (5 tasks)

The Morpheus-to-Datenberg tasks cover the transformation of execution-oriented MES data into flattened association structures for analytics and dashboard visualization. This category emphasizes **denormalization**, **explicit relationship extraction**, and **sequence preservation**.

Task 3: Workstation–Production Line Associations (medium). Transforms a `MorpheusModel` into `List[WorkstationProductionLineAssociation]` by iterating over production lines and emitting one association per workstation reference.

Task 4: Workprocess–Variant Associations (medium). Transforms a `MorpheusModel` into `List[WorkprocessVariantAssociation]` by iterating over variants and extracting ordered work process sequences. The `sequence_number` is derived from the index of each process in the variant-specific sequence list.

Task 5: Polehousing–Variant Association (easy). Maps a single `PoleHousing` instance to a `PolehousingVariantAssociation` by linking instance and variant identifiers.

Task 6: ProcessData to Event (easy). Transforms a `ProcessData` execution record into an `Event` representation used for analytics, including start time, resource/procedure references, an end time field, and a success indicator.

Task 7: Event–GameRound–Training Association (easy). Combines `ProcessData` with contextual entities (`Training` and `GameRound`) to emit an `EventGameRoundTraining` association record, enabling dashboard-level grouping of events by training sessions and scenarios.

A7.2.6 Morpheus → ReferenceModel (18 tasks)

This category constitutes the largest share of tasks and reflects the integration of execution-level MES data into a standardized production representation. It includes both **full-model transformations** and **targeted extraction tasks** that derive particular reference-model entity sets. Typical challenges include multi-entity joins, capability derivation, hierarchical resource modeling, and derivation of executable procedures.

Task 8: Complete MorpheusModel to ReferenceModel (hard). A full transformation from `MorpheusModel` to `ReferenceModel`, covering resources, processes, procedures, products, orders, and performance/event information. This task represents the end-to-end integration of an MES into the reference schema.

Entity-level transformations (Tasks 9–18). The following tasks transform individual Morpheus entities into their ReferenceModel counterparts:

- Task 9 (medium): `WorkStation` → `Resource` (capabilities derived from process assignments)

- Task 10 (hard): `ProductionLine` → `Resource` (hierarchical resource with sub-resources and aggregated capabilities)
- Task 11 (medium): `WorkProcess` → `Process`
- Task 12 (medium): `EditableVariant` → `Process` (sequence/production process representation)
- Task 13 (easy): `EditablePart` → `Product`
- Task 14 (medium): `EditableVariant` → `Product` (including BOM/sub-products and tracking data)
- Task 15 (hard): `WorkProcess` → `Procedure` (execution model derived from historical process data)
- Task 16 (medium): `Order` (Morpheus) → `Order` (ReferenceModel)
- Task 17 (easy): `ProcessData` → `Event`
- Task 18 (easy): `Location` (Morpheus) → `Location` (ReferenceModel)

Model extraction transformations (Tasks 19–25). These tasks derive lists of particular entity types from a full `MorpheusModel`. They represent common integration patterns where only partial standardized views are required:

- Task 19 (hard): `MorpheusModel` → `List[Process]`
- Task 20 (hard): `MorpheusModel` → `List[Product]`
- Task 21 (hard): `MorpheusModel` → `List[Procedure]`
- Task 22 (hard): `MorpheusModel` → `List[Resource]`
- Task 23 (medium): `MorpheusModel` → `List[Order]`
- Task 24 (medium): `MorpheusModel` → `List[Performance]`
- Task 25 (medium): `Order` (Morpheus) → `List[OrderedProduct]`

A7.2.7 Morpheus → Tulip (1 task)

Task 26: Morpheus Order to Tulip Table Entry (easy). This task creates a Pick-by-Light compatible table entry for the Tulip platform (`OrderReceivedTableEntry`) from a Morpheus `Order`. It represents a downstream integration into a constrained, table-oriented schema.

Key transformations:

- `id` → `id`
- timestamp conversion to `bnjsa_timestamp`
- `variantId` → `sehib_variant_name`

A7.2.8 prodsys → ReferenceModel (4 tasks)

These tasks transform simulation-centric production models into the standardized ReferenceModel representation, focusing on resources and procedures. This category contains non-trivial semantic alignment, particularly for state-based procedures (setup, breakdown, maintenance) and unit conversions.

Task 27: Complete prodsysModel to ReferenceModel (hard). Transforms a `ProdsysModel` into a `ReferenceModel`, focusing on resources and procedures. Setup, breakdown, and maintenance semantics are derived from state objects.

Characteristic rules: (i) only resources and procedures are transformed; (ii) procedure types are inferred from state categories; (iii) temporal conversions are applied (minutes → seconds).

Task 28: prodsysModel to List[Procedure] (hard). Extracts only procedure entities from `BreakDownStateData` and `SetupStateData`.

Task 29: prodsysModel to List[Resource] (hard). Transforms production and transport resources into station-level reference resources and introduces a plant-level resource.

Task 30: prodsys Resource to Resource (medium). Transforms a single prodsys resource entry into a `Resource`, including capability inference from `process_ids` and the inclusion of derived state-related procedures.

Task 31: SetupStateData to Procedure (medium). Creates a setup procedure with origin/-target capability attributes and converts the associated time model (minutes → seconds).

A7.2.9 ReferenceModel → IntralogisticMapFormat (1 task)

Task 32: ReferenceModel to IntralogisticMapFormat (hard). Transforms a ReferenceModel into an LIF-compliant intralogistics layout representation. The transformation maps station-level resources into either stations or obstacles based on their semantic type, applies fixed boundary sizes, and constructs a static outer boundary.

Key mapping rules:

- Only resources are transformed.
- Manufacturing/Source/Sink resources at station level → `Station` objects with a fixed boundary size (1.2 m × 1.2 m).
- Storage/Barrier resources at station level → `Obstacle` objects with a fixed boundary size (2.0 m × 2.0 m).
- Resources representing empty spots or material flow are excluded.
- Static map parameters are used (e.g., rotation radius and outer boundary).
- Positions are derived from the resource configuration (typically from sub-resources).

A7.2.10 ReferenceModel → prodsys (10 tasks)

The ReferenceModel-to-prodsys tasks represent the generation of simulation-ready data structures from a standardized production representation. They include time model conversion, procedure-to-process transformations, state derivations, and resource synthesis.

Time model tasks (Tasks 33–34).

- Task 33 (easy): `TimeModel` → `TIME_MODEL_DATA`, including handling of different time model types and conversion (seconds → minutes).
- Task 34 (medium): `ReferenceModel` → `List[TIME_MODEL_DATA]`, extracting all time models and deriving additional arrival models from orders where required.

Process and process module tasks (Tasks 35–36).

- Task 35 (hard): `ReferenceModel` → `List[PROCESS_DATA_UNION]`, converting procedures to prodsys processes, including required-capability and transport process representations.
- Task 36 (medium): `ReferenceModel` → `List[CompoundProcessData]`, generating compound processes at module level and applying deduplication rules.

Product and state tasks (Tasks 37–38).

- Task 37 (medium): `ReferenceModel` → `List[ProductData]`, converting product definitions and process references into prodsys format.
- Task 38 (hard): `ReferenceModel` → `List[STATE_DATA_UNION]`, deriving setup and breakdown state data from setup and maintenance/breakdown procedures.

Resource, source, and sink tasks (Tasks 39–41).

- Task 39 (hard): `ReferenceModel` → `List[RESOURCE_DATA_UNION]`, generating production and transport resources including process/state assignments and spatial placement rules.
- Task 40 (medium): `ReferenceModel` → `List[SourceData]`, generating source definitions from source-type resources and selecting base products.
- Task 41 (medium): `ReferenceModel` → `List[SinkData]`, generating sink definitions from sink-type resources and selecting base products.

A7.2.11 Standardized Task Definition Format

Each mapping task in the benchmark is specified using a standardized task descriptor containing schema references, JSON schema definitions, an expected output schema, and natural language transformation rules. Conceptually, each task is defined by:

- the source data model and input schema(s),
- the target data model and output schema,
- a difficulty label (`easy`, `medium`, `hard`),
- a gold-standard mapper function used as reference,

- and additional transformation constraints expressed in natural language.

A7.2.12 Difficulty Criteria

The three difficulty classes used for analysis in Section 7.2 are defined as follows:

Easy: Direct field mappings, simple renaming, basic type conversions, typically involving one or two entities.

Medium: Iteration over collections, lookup operations across model components, conditional logic, and transformations spanning multiple entities.

Hard: Multi-step aggregations, hierarchical synthesis, non-local inference rules (e.g., capability or procedure derivation), nested transformations, and multi-entity consistency constraints.

A7.3 Detailed results of Data Integration Benchmark

Model	Mapping Case	Eff.	F1		Precision		Recall	
			μ	σ	μ	σ	μ	σ
DeepSeek R1	S	0.49	0.34	0.42	0.37	0.45	0.36	0.41
	SED	0.34	0.25	0.38	0.25	0.38	0.27	0.42
	SEDR	0.22	0.19	0.37	0.20	0.38	0.19	0.37
Gemini 2.5 Flash	S	0.81	0.49	0.38	0.49	0.39	0.59	0.40
	SED	0.83	0.52	0.40	0.52	0.39	0.62	0.40
	SEDR	**0.95**	0.84	0.25	0.84	0.26	0.88	0.25
Gemini 2.5 Pro	S	0.78	0.55	0.39	0.52	0.38	0.62	0.41
	SED	0.78	0.54	0.40	0.52	0.39	0.67	0.42
	SEDR	**0.95**	**0.89**	**0.23**	**0.87**	**0.24**	**0.91**	**0.22**
GPT-5	S	0.71	0.45	0.40	0.48	0.43	0.53	0.41
	SED	0.78	0.51	0.41	0.50	0.41	0.60	0.43
	SEDR	0.93	0.85	0.27	0.85	0.28	0.86	0.26
Grok 4	S	0.68	0.44	0.40	0.41	0.40	0.56	0.43
	SED	0.61	0.43	0.43	0.41	0.41	0.52	0.47
	SEDR	0.73	0.66	0.42	0.64	0.41	0.68	0.43

Table A7.3: Benchmark results for LLMs across mapping cases showing effectiveness (eff.) and the mean (μ) and standard deviation (σ) of the accuracy metrics F1, precision, and recall.

A8 Micro Benchmarking Workflows

Figure A8.1 visualizes the 5 process model patterns used in the (Skouradaki; Ferme et al. 2016). In the original benchmark, these process models are realized with BPMN, but a translation to Petri nets was performed to evaluate SAPNs. Translations were performed under the principle to maintain to maintain the same logic for every individual process model pattern, as described by the BPMN models.

Each process model in the benchmark is deliberately minimal, avoiding external interactions and using mostly empty script tasks to ensure that observed effects stem from the process model semantics rather than application logic.

Sequence Flow (SEQ). This process model models a simple linear progression of two script tasks. It captures the baseline overhead of the process navigator and serves as the fundamental reference point for all other patterns. Its purpose is to measure the minimal execution cost of sequential task progression.

Parallel Split and Synchronization (PAR). This process model executes two branches concurrently and synchronizes them before completion. It isolates the effects of parallelism on process model execution, enabling the study of engine behavior under multi-threaded execution and join operations.

Exclusive Choice and Simple Merge (EXC). The EXC model adds a conditional branch controlled by a randomly generated value. Its objective is to quantify the cost of gateway evaluation and merging, revealing the overhead introduced by basic decision logic.

Explicit Termination (EXT). The EXT pattern creates two concurrent branches, where one completes quickly while the other performs a fixed waiting operation. According to BPMN 2.0 semantics, completion of the fast branch triggers termination of the slower one. This pattern assesses how engines handle termination propagation and interruption of running activities.

Arbitrary Cycle (CYC). The CYC process model implements a loop using nested exclusive gateways and a counter variable. Depending on the initial branch, the loop iterates either a short or long number of cycles. This pattern stresses the OE with repeated gateway evaluations and cyclic control flow, making it suitable for revealing bottlenecks in loop management and state transitions.

Mixed Pattern Workload (MIX). The MIX workload executes all above process model types concurrently in a uniform distribution. Its purpose is to evaluate whether interactions between heterogeneous control-flow constructs create performance side effects, and to analyze system stability under combined, realistic workloads.

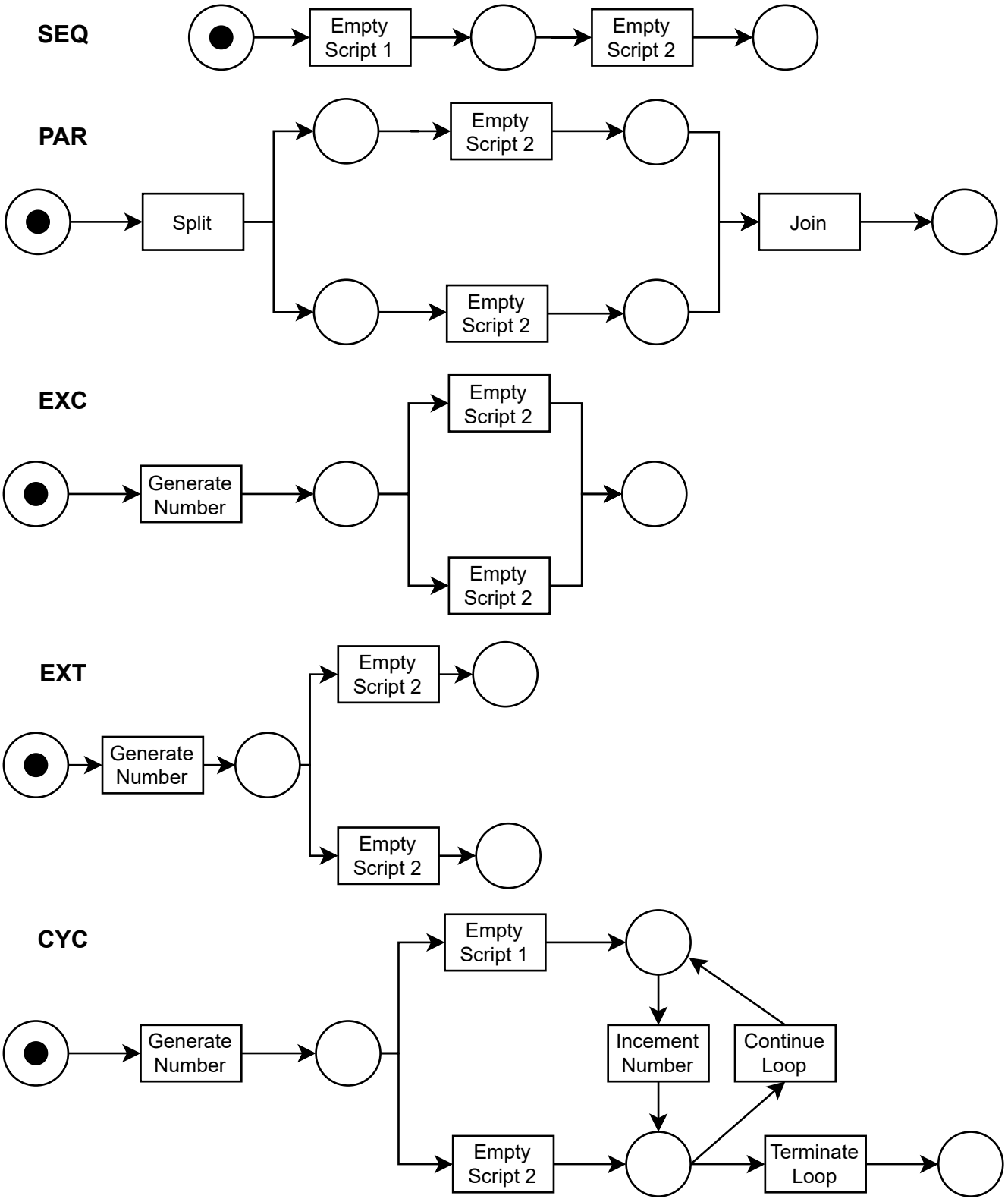

Figure A8.1: Visualization of the Workflow Patterns used in the micro benchmarking, based on (Skouradaki; Ferme et al. 2016).

Table A8.1: Summary of Mean Duration (Latency) and Throughput for Micro-Benchmark Experiments.

Pattern	System	Mean Duration (ms)	Median (ms)	Std Dev (ms)	Total Executions	Throughput (wi/s)
SEQ	WfMS A	0.39	0.00	1.70	781,736	1447.66
	WfMS B	6.39	6.00	1.21	35,516	63.31
	Camunda	0.74	1.00	2.29	786,664	1456.79
	SAPN	**0.52**	**0.31**	**0.70**	**889,061**	**1412.01**
EXC	WfMS A	0.48	0.00	2.07	775,455	1436.03
	WfMS B	9.30	9.00	2.11	27,805	49.04
	Camunda	0.85	1.00	2.51	765,274	1417.17
	SAPN	**0.51**	**0.32**	**0.99**	**889,349**	**1412.58**
EXT	WfMS A	14.10	11.00	13.45	770,229	1426.35
	WfMS B	2622.00	5012.00	2500.44	1,703	0.38
	Camunda	0.40	0.00	1.03	784,614	1455.68
	SAPN	**0.89**	**0.59**	**1.46**	**890,832**	**1414.78**
PAR	WfMS A	13.29	10.00	11.99	772,013	1429.65
	WfMS B	10.06	10.00	2.22	27,718	48.89
	Camunda	0.70	1.00	2.10	773,883	1433.12
	SAPN	**0.62**	**0.40**	**0.77**	**889,322**	**1412.68**
CYC	WfMS A	6.23	2.00	18.68	347,770	644.02
	WfMS B	39.36	43.00	9.52	8,695	13.46
	Camunda	3.06	2.00	4.43	177,770	327.99
	SAPN	**2.10**	**1.35**	**2.21**	**888,174**	**1411.10**
MIX	WfMS A	8.16	2.00	14.65	758,659	1402.33
	WfMS B	540.02	12.00	1525.27	2,392	1.78
	Camunda	1.22	1.00	4.21	575,210	1061.27
	SAPN	**0.70**	**0.40**	**1.07**	**852,475**	**1356.03**

A9 Case Study 1

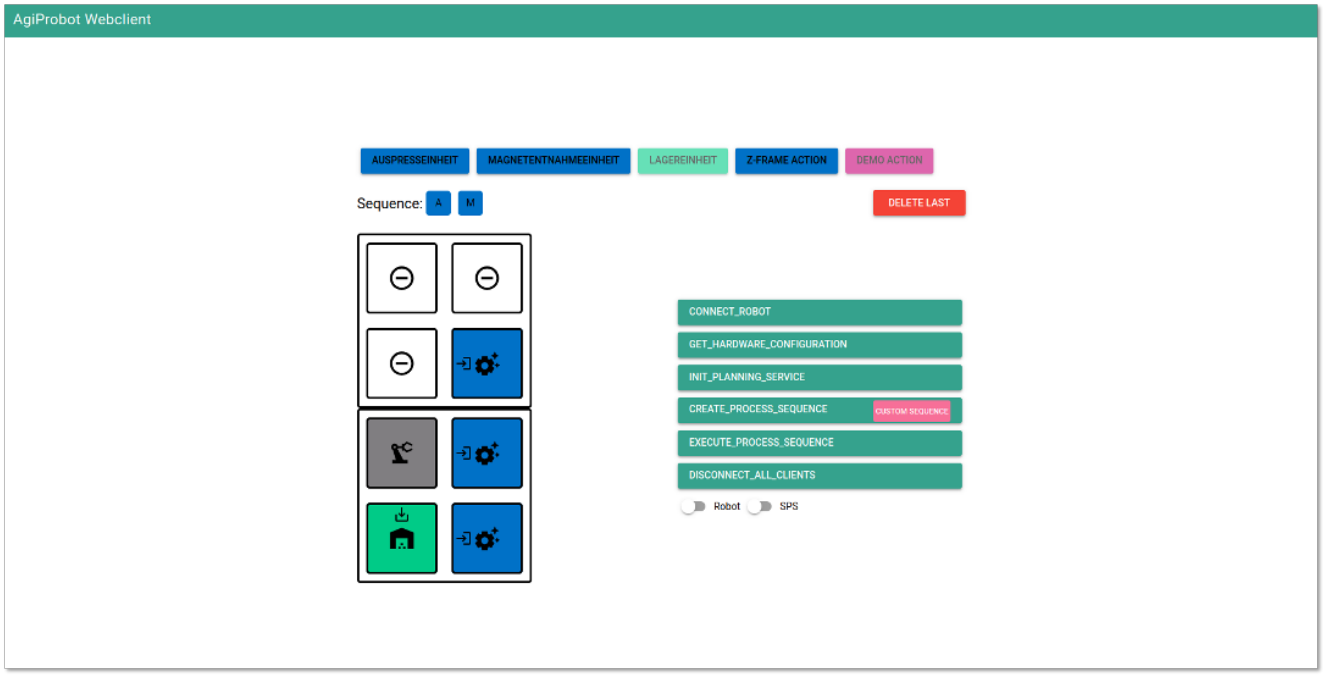

Figure A9.1: Screenshot of the web-based UI of the modular station illustrating station setup, the graphical configuration of process sequences, and the execution control interface.

A10 Eclipse Data Space Connector

The DPP demonstrator used in case study 2 (Section 7.5) builds on the concept proposed by Gleich; Behrendt et al. (2024) and operationalizes it through an EDC-based cross-company data exchange Workflow. The EDC forms the technological backbone for secure, sovereign, and policy-controlled data sharing within federated data spaces such as Gaia-X (Dam; Klausner et al. 2024). In essence, the EDC acts as a contract- and policy-enforcement gateway that mediates data exchange between organizations without ever requiring centralization of sensitive information. Each participating entity deploys an EDC runtime, which exposes standardized REST endpoints for negotiating access policies, creating usage contracts, requesting assets, and initiating data transfers. During a data exchange, the EDC verifies the requesting party's identity, evaluates data-usage policies, ensures that only the contractually permitted subset of information is released, and orchestrates the transfer channel. No raw data is exchanged before contract negotiation is completed and policies are validated, ensuring compliance with European data sovereignty principles and maintaining local control over all data assets.

In the implemented DPP scenario, the Provider Middleware registers the SDM-formatted DPP dataset as an EDC asset and associates it with fine-grained usage policies (e.g., read-only access, limited retention, purpose-bound processing). The Consumer Middleware initiates a contract negotiation via its EDC instance, which results in a mutually agreed data usage contract. Once the contract is established, the EDC triggers the secure transfer of the DPP payload from the provider to the consumer. Importantly, the EDC enforces a pull-based, contract-governed data exchange, avoiding direct exposure of backend systems. This mechanism enables the hybrid architecture described in the case study: on-premise MES events are transformed into SDM-compliant DPP data, exposed through the cloud-based provider EDC, and finally materialized into an AAS instance by the consumer Middleware. The result is a decentralized yet interoperable architecture in which sensitive production data remains local, and only standardized, policy-controlled DPP information traverses company boundaries - a key principle emphasized in recent literature on Gaia-X - aligned DPP implementations.

A11 Case Study 2

The analysis of latency, depicted in the box plots of Figure A11.1, reveals critical performance characteristics of the different integration patterns. Simple, virtualized integrations that perform in-memory mapping, such as the Consumer Middleware transforming data to the AAS format, exhibit extremely low latencies, often in the microsecond range. This highlights the efficiency of the core framework for stateless transformations. In contrast, components involved in more complex operations or external communication show higher latency. The Provider EDC and Consumer EDC Middleware instances show latency in the tens of milliseconds, attributable to the overhead of the EDC protocol negotiation and network communication. The prodsys and SimController Middleware show the highest latencies, measured in seconds, which is expected as they involve synchronous calls to external, computationally intensive simulation engines.

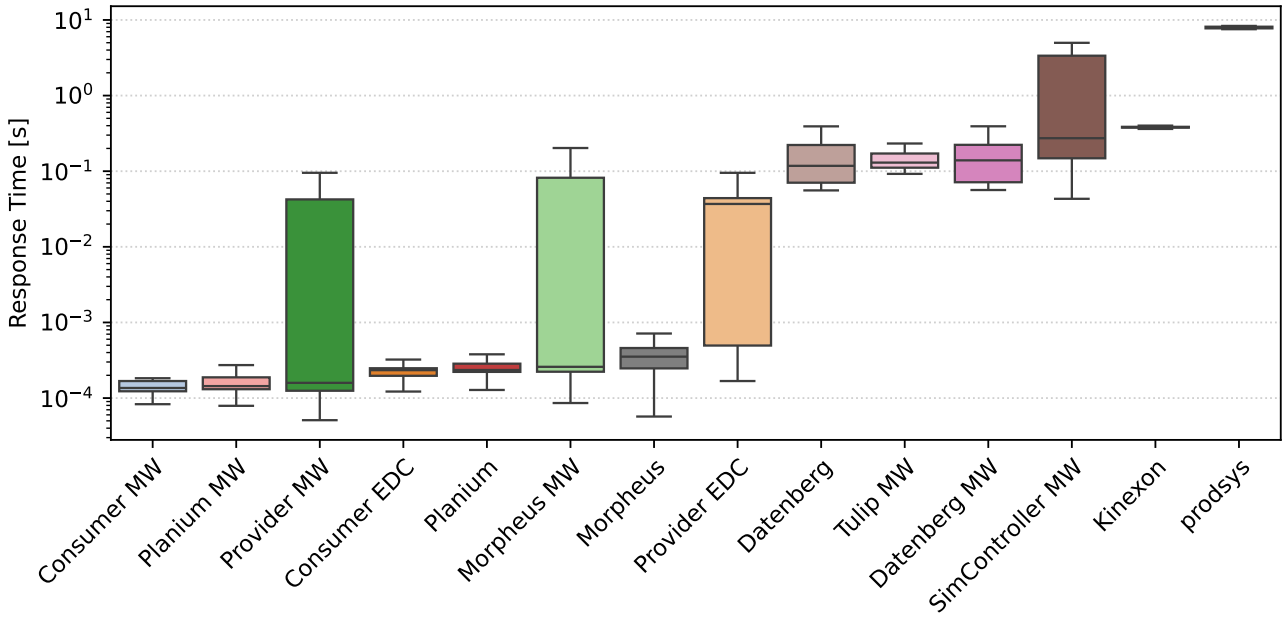

Figure A11.1: Box plot analysis of communication latency for each service.

The distribution of message sizes, shown in Figure A11.2, complements the latency analysis by illustrating the data payload associated with each interaction. Many interactions, such as those with Tulip or the EDC protocol messages, involve very small payloads (often just a few bytes), acting as simple triggers or control signals rather than large data transfers. Conversely, integrations with simulation systems (prodsys, SimController Middleware) or data-rich sources (Kinexon) involve significantly larger message sizes, ranging from kilobytes to megabytes. This reflects the transfer of complex model data, layout configurations, or location datasets, and explains in part the higher latencies observed for these services. The variety in payload

size underscores the framework's flexibility in handling both lightweight events and heavy data-centric communication.

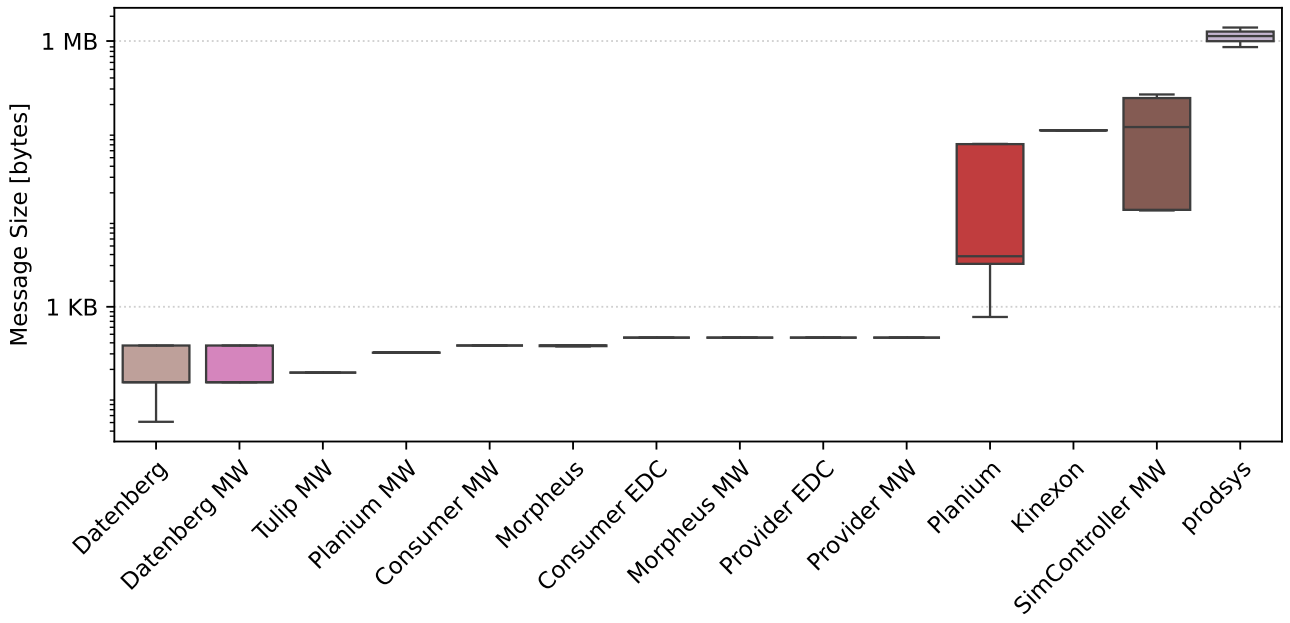

Figure A11.2: Box plot analysis of message sizes for each service.

A12 Case Study 3

A12.1 Production Planning and Control Services

The closed-loop PPC system of case study 3 (Section 7.6) integrates a set of modular planning and validation services whose roles and data dependencies are summarized in Table A12.1. Together, these services execute the sequential and feedback-oriented planning logic visualized in Figure A12.1. Each service operates on a shared AAS-based information model and communicates via the Middleware, ensuring schema conformity and task-agnostic interoperability.

Table A12.1: Overview of utilized PPC services, categorized by task and type.

PPC Task	Service(s)	Type	Input	Output
CPL	prodsys	Planning	System config, orders	Optimized system config
Scheduling	Flexis Scheduler, CP-SAT Solver	Planning	Orders, system config	Schedule
Simulation	SimController, prodsys	Validation	Orders, schedule, system config	System output and utilization
AGV Route Planning	AGV Fleet Management	Planning	Schedule, system config	Validated AGV routes

The PPC loop begins with **CPL**, executed by *prodsys*, which uses meta-heuristics and discrete-event simulation to derive an optimized production system configuration based on an existing system (Figure A12.1a). This output, shown in Figure A12.1b, forms the structural basis for downstream tasks.

Based on this configuration, **Scheduling** assigns orders and operations to available resources. The *Flexis Scheduler* computes an initial plan using constructive heuristics, while the *CP-SAT solver* provides a fallback for constraint-tight or time-critical cases. The resulting fine-grained schedule (Figure A12.1c) captures operation sequences, resource assignments, and timing.

The schedule is then assessed through **Simulation** using either the high-fidelity *SimController* or the lightweight *prodsys* engine. This validation step evaluates throughput, utilization, and lead times (Figure A12.1d), enabling early detection of infeasible or unstable plans. If simulation reveals deviations from the required output or bottlenecks, the process model triggers an automatic replanning cycle.

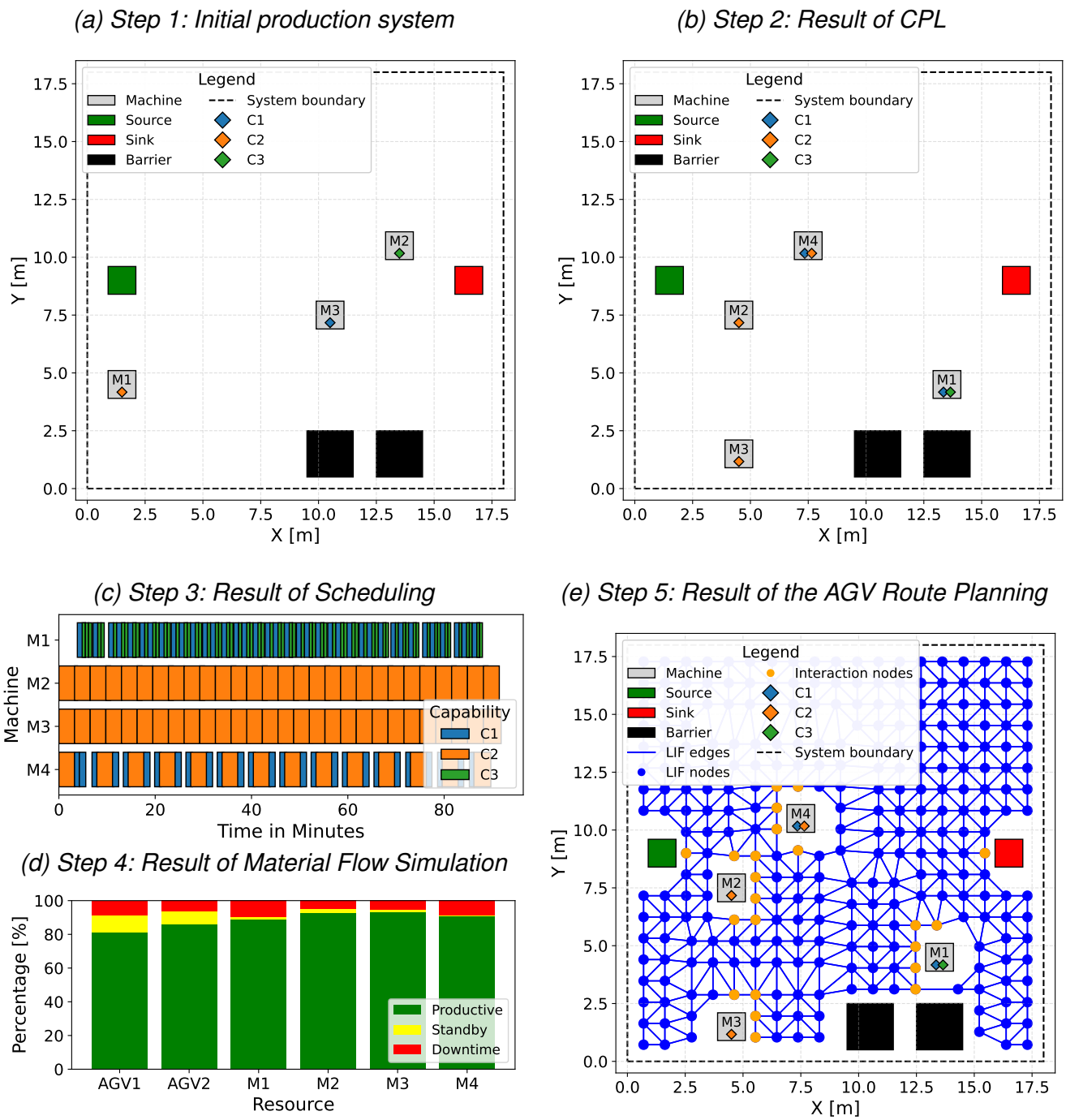

Figure A12.1: Chronological visualization of the four planning steps.

Finally, **AGV Route Planning** refines intralogistics by generating route networks and validating AGV behavior using a multi-agent path-finding algorithm. The AGV fleet management service produces collision- and deadlock-free trajectories (Figure A12.1e), ensuring that material flow can support the planned production schedule.

Together, these services form a tightly integrated chain: CPL defines the system configuration; scheduling allocates work in accordance with this configuration; simulation verifies the

resulting plan; and AGV routing ensures executable logistics. Their interaction based on the SAPN process model and Middleware-driven integration enables automated, repeatable, and reconfigurable closed-loop PPC.

A12.2 Technical Performance Analysis

The analysis of service interactions (Figure A12.2a, Figure A12.2b), message sizes (Figure A12.3a, Figure A12.3b) and service response times (Figure A12.4a, Figure A12.4b) confirmed the different characteristics of IT and OT communication. Planning services (IT) involved high-latency, high-volume interactions, while shop-floor control services (OT) required low-latency, low-volume, high-frequency communication. This validated the architectural decision to decouple planning and control Middleware, as it allows each component to be optimized for its specific requirements. While challenges such as handling gigabyte-scale planning payloads and ensuring low latency for AAS access were encountered and addressed with caching and streaming, the overall technical evaluation was successful.

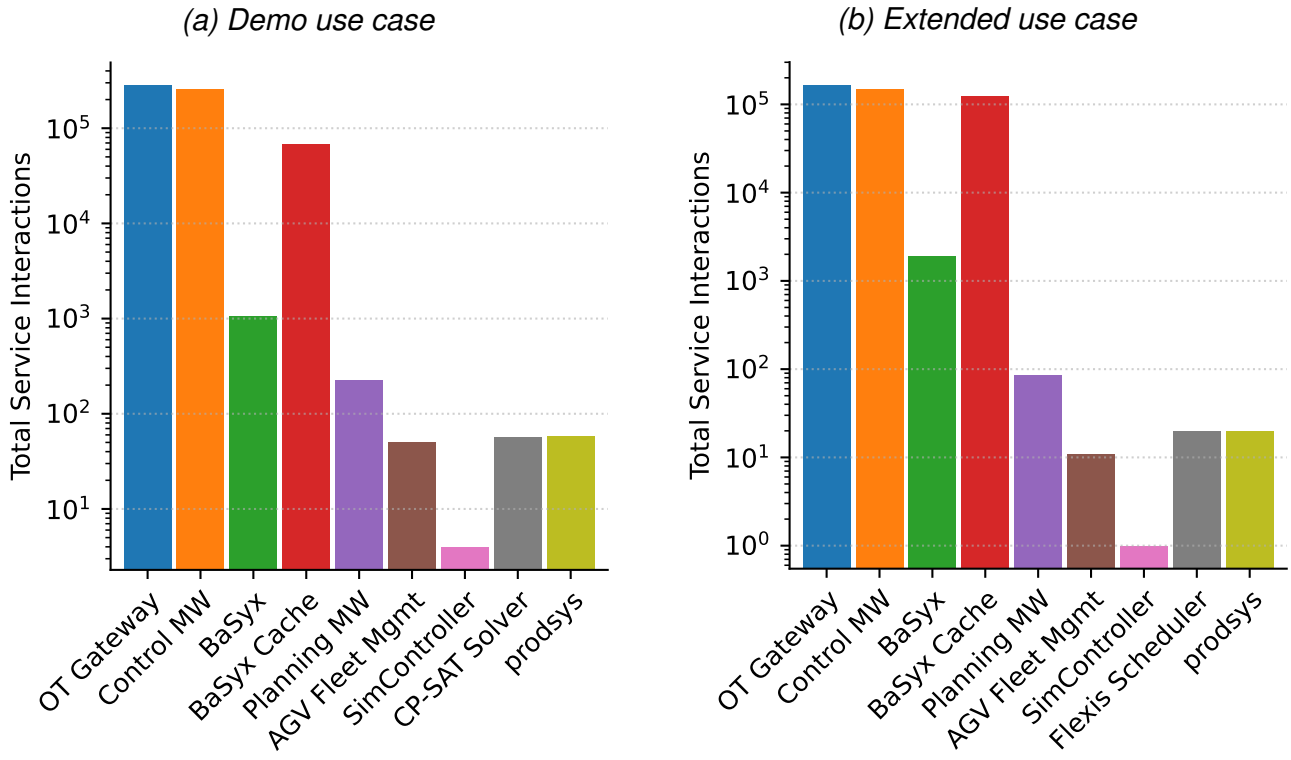

Figure A12.2: Number of Interactions per Service for both use cases.

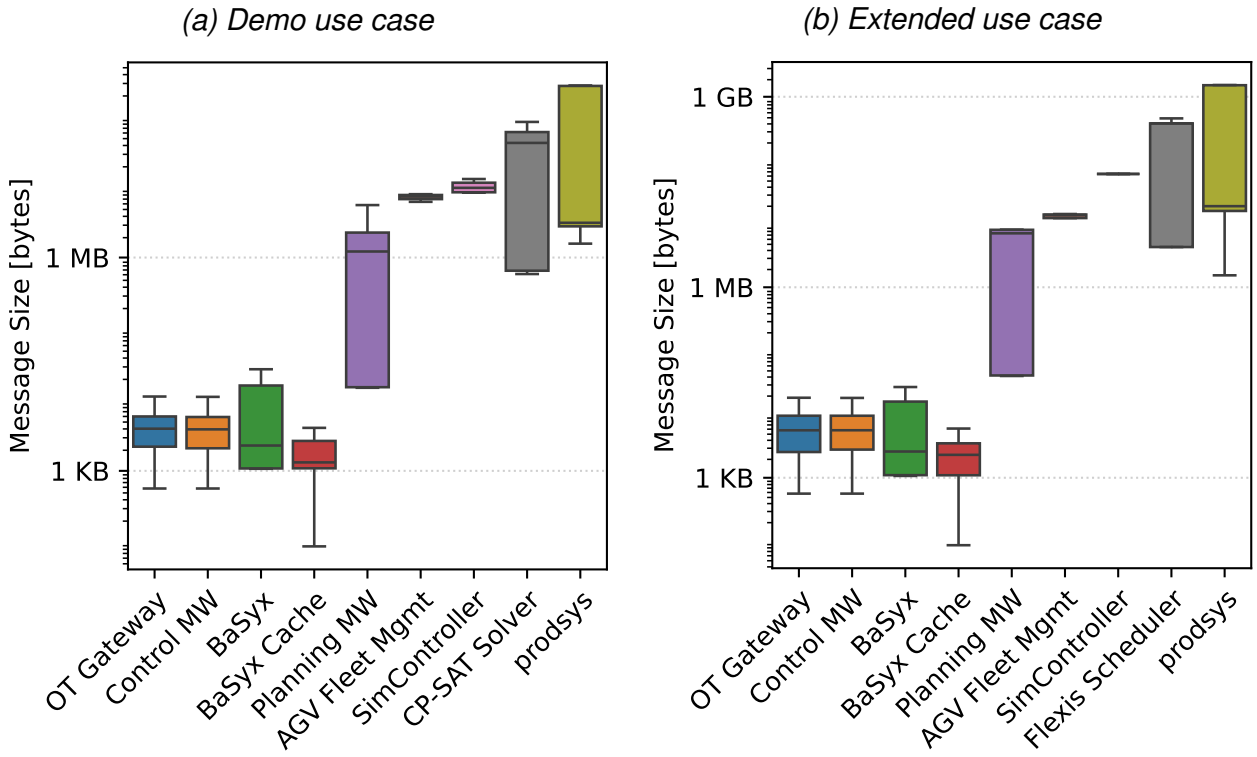

Figure A12.3: Message Size distribution per Service for both use cases.

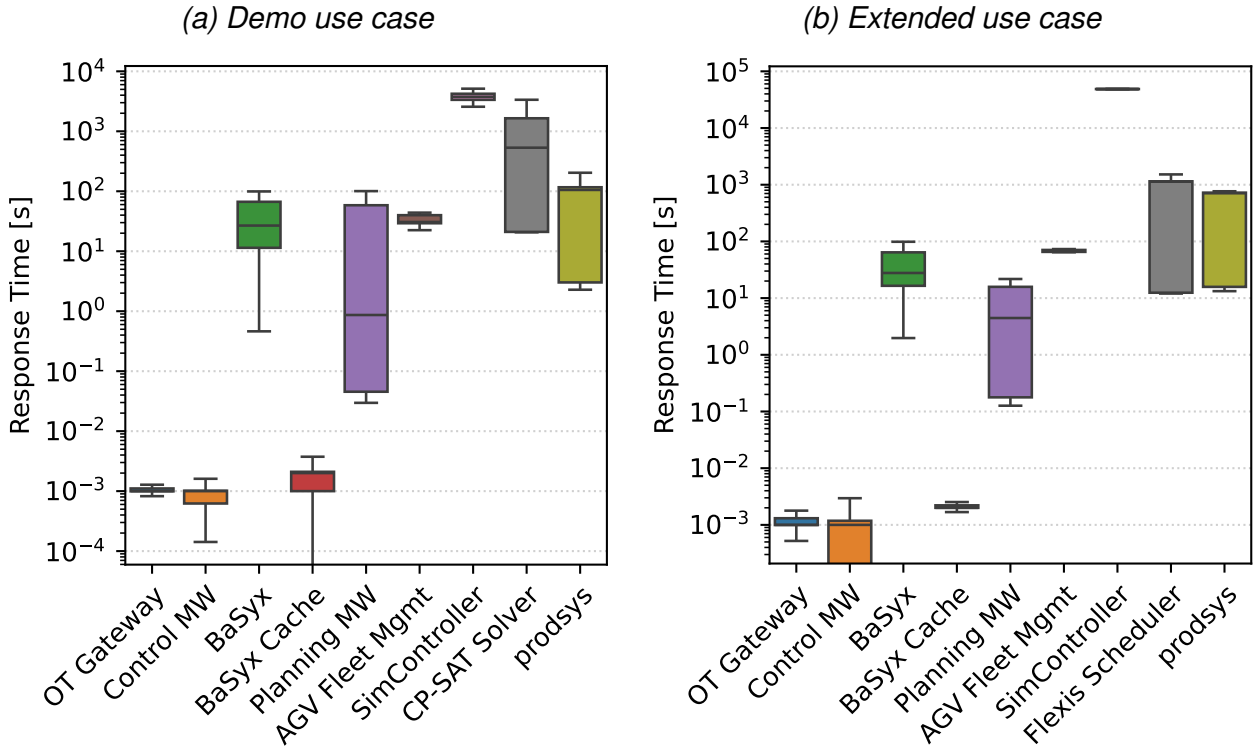

Figure A12.4: Service-wise interaction response time distributions for both use cases in seconds.

Forschungsberichte aus dem wbk
Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Bisher erschienene Bände:

Band 0
Dr.-Ing. Wu Hong-qi

Adaptive Volumenstromregelung mit Hilfe von drehzahlgeregelten Elektroantrieben

Band 1
Dr.-Ing. Heinrich Weiß

Fräsen mit Schneidkeramik - Verhalten des System Werkzeugmaschine-Werkzeug-Werkstück und Prozessanalyse

Band 2
Dr.-Ing. Hans-Jürgen Stierle

Entwicklung und Untersuchung hydrostatischer Lager für die Axialkolbenmaschine

Band 3
Dr.-Ing. Herbert Hörner

Untersuchung des Geräuschverhaltens druckgeregelter Axialkolbenpumpen

Band 4
Dr.-Ing. Rolf-Dieter Brückbauer

Digitale Drehzahlregelung unter der besonderen Berücksichtigung von Quantisierungseffekten

Band 5
Dr.-Ing. Gerhard Staiger

Graphisch interaktive NC-Programmierung von Drehteilen im Werkstattbereich

Band 6
Dr.-Ing. Karl Peters

Ein Beitrag zur Berechnung und Kompensation von Positionierfehlern an Industrierobotern

Band 7
Dr.-Ing. Paul Stauss

Automatisierte Inbetriebnahme und Sicherung der Zuverlässigkeit und Verfügbarkeit numerisch gesteuerter Fertigungseinrichtungen

Band 8
Dr.-Ing. Günter Möckesch

Konzeption und Realisierung eines strategischen, integrierten Gesamtplanungs- und -bearbeitungssystems zur Optimierung der Drehteilorganisation für auftragsbezogene Drehereien

Band 9
Dr.-Ing. Thomas Oestreicher

Rechnergestützte Projektierung von Steuerungen

Band 10
Dr.-Ing. Thomas Selinger

Teilautomatisierte werkstattnahe NC-Programmerstellung im Umfeld einer integrierten Informationsverarbeitung

Band 11
Dr.-Ing. Thomas Buchholz

Prozessmodell Fräsen, Rechnerunterstützte Analyse, Optimierung und Überwachung

Band 12
Dr.-Ing. Bernhard Reichling

Lasergestützte Positions- und Bahnvermessung von Industrierobotern

Band 13
Dr.-Ing. Hans-Jürgen Lesser

Rechnergestützte Methoden zur Auswahl anforderungsgerechter Verbindungselemente

Band 14
Dr.-Ing. Hans-Jürgen Lauffer

Einsatz von Prozessmodellen zur rechnerunterstützten Auslegung von Räumwerkzeugen

Band 15
Dr.-Ing. Michael C. Wilhelm

Rechnergestützte Prüfplanung im Informationsverbund moderner Produktionssysteme

Band 16
Dr.-Ing. Martin Ochs

Entwurf eines Programmsystems zur wissensbasierten Planung und Konfigurierung

Band 17
Dr.-Ing. Heinz-Joachim Schneider

Erhöhung der Verfügbarkeit von hochautomatisierten Produktionseinrichtungen mit Hilfe der Fertigungsleittechnik

Band 18
Dr.-Ing. Hans-Reiner Ludwig

Beanspruchungsanalyse der Werkzeugschneiden beim Stirnplanfräsen

Band 19
Dr.-Ing. Rudolf Wieser

Methoden zur rechnergestützten Konfigurierung von Fertigungsanlagen

Band 20
Dr.-Ing. Edgar Schmitt

Werkstattsteuerung bei wechselnder Auftragsstruktur

Band 21
Dr.-Ing. Wilhelm Enderle

Verfügbarkeitssteigerung automatisierter Montagesysteme durch selbsttätige Behebung prozessbedingter Störungen

Band 22
Dr.-Ing. Dieter Buchberger

Rechnergestützte Strukturplanung von Produktionssystemen

Band 23
Prof. Dr.-Ing. Jürgen Fleischer

Rechnerunterstützte Technologieplanung für die flexibel automatisierte Fertigung von Abkantteilen

Band 24
Dr.-Ing. Lukas Loeffler

Adaptierbare und adaptive Benutzerschnittstellen

Band 25
Dr.-Ing. Thomas Friedmann

Integration von Produktentwicklung und Montageplanung durch neue rechnergestützte Verfahren

Band 26
Dr.-Ing. Robert Zurrin

Variables Formhonen durch rechnergestützte Hornprozesssteuerung

Band 27
Dr.-Ing. Karl-Heinz Bergen

Langhub-Innenrundhonen von Grauguss und Stahl mit einem elektromechanischem Vorschubsystem

Band 28
Dr.-Ing. Andreas Liebisch

Einflüsse des Festwalzens auf die Eigenspannungsverteilung und die Dauerfestigkeit einsatzgehärteter Zahnräder

Band 29
Dr.-Ing. Rolf Ziegler

Auslegung und Optimierung schneller Servopumpen

Band 30
Dr.-Ing. Rainer Bartl

Datenmodellgestützte Wissensverarbeitung zur Diagnose und Informationsunterstützung in technischen Systemen

Band 31
Dr.-Ing. Ulrich Golz

Analyse, Modellbildung und Optimierung des Betriebsverhaltens von Kugelgewindetrieben

Band 32
Dr.-Ing. Stephan Timmermann

Automatisierung der Feinbearbeitung in der Fertigung von Hohlformwerkzeugen

Band 33
Dr.-Ing. Thomas Noe

Rechnergestützter Wissenserwerb zur Erstellung von Überwachungs- und Diagnoseexpertensystemen für hydraulische Anlagen

Band 34
Dr.-Ing. Ralf Lenschow

Rechnerintegrierte Erstellung und Verifikation von Steuerungsprogrammen als Komponente einer durchgängigen Planungsmethodik

Band 35
Dr.-Ing. Matthias Kallabis

Räumen gehärteter Werkstoffe mit kristallinen Hartstoffen

Band 36
Dr.-Ing. Heiner-Michael Honeck

Rückführung von Fertigungsdaten zur Unterstützung einer fertigungsgerechten Konstruktion

Band 37
Dr.-Ing. Manfred Rohr

Automatisierte Technologieplanung am Beispiel der Komplettbearbeitung auf Dreh-/Fräszellen

Band 38
Dr.-Ing. Martin Steuer

Entwicklung von Softwarewerkzeugen zur wissensbasierten Inbetriebnahme von komplexen Serienmaschinen

Band 39
Dr.-Ing. Siegfried Beichter

Rechnergestützte technische Problemlösung bei der Angebotserstellung von flexiblen Drehzellen

Band 40
Dr.-Ing. Thomas Steitz

Methodik zur marktorientierten Entwicklung von Werkzeugmaschinen mit Integration von funktionsbasierter Strukturierung und Kostenschätzung

Band 41
Dr.-Ing. Michael Richter

Wissensbasierte Projektierung elektrohydraulischer Regelungen

Band 42
Dr.-Ing. Roman Kuhn

Technologieplanungssystem Fräsen. Wissensbasierte Auswahl von Werkzeugen, Schneidkörpern und Schnittbedingungen für das Fertigingsverfahren Fräsen

Band 43
Dr.-Ing. Hubert Klein

Rechnerunterstützte Qualitätssicherung bei der Produktion von Bauteilen mit frei geformten Oberflächen

Band 44
Dr.-Ing. Christian Hoffmann

Konzeption und Realisierung eines fertigungsintegrierten Koordinatenmessgerätes

Band 45
Dr.-Ing. Volker Frey

Planung der Leittechnik für flexible Fertigungsanlagen

Band 46
Dr.-Ing. Achim Feller

Kalkulation in der Angebotsphase mit dem selbsttätig abgeleiteten Erfahrungswissen der Arbeitsplanung

Band 47
Dr.-Ing. Markus Klaiber

Produktivitätssteigerung durch rechnerunterstütztes Einfahren von NC-Programmen

Band 48
Dr.-Ing. Roland Minges

Verbesserung der Genauigkeit beim fünfachsigen Fräsen von Freiformflächen

Band 49
Dr.-Ing. Wolfgang Bernhart

Beitrag zur Bewertung von Montagevarianten: Rechnergestützte Hilfsmittel zur kostenorientierten, parallelen Entwicklung von Produkt und Montagesystem

Band 50
Dr.-Ing. Peter Ganghoff

Wissensbasierte Unterstützung der Planung technischer Systeme: Konzeption eines Planungswerkzeuges und exemplarische Anwendung im Bereich der Montagesystemplanung

Band 51
Dr.-Ing. Frank Maier

Rechnergestützte Prozessregelung beim flexiblen Gesenkbiegen durch Rückführung von Qualitätsinformationen

Band 52
Dr.-Ing. Frank Debus

Ansatz eines rechnerunterstützten Planungsmanagements für die Planung in verteilten Strukturen

Band 53
Dr.-Ing. Joachim Weinbrecht

Ein Verfahren zur zielorientierten Reaktion auf Planabweichungen in der Werkstattregelung

Band 54
Dr.-Ing. Gerd Herrmann

Reduzierung des Entwicklungsaufwandes für anwendungsspezifische Zellenrechnersoftware durch Rechnerunterstützung

Band 55
Dr.-Ing. Robert Wassmer

Verschleissentwicklung im tribologischen System Fräsen: Beiträge zur Methodik der Prozessmodellierung auf der Basis tribologisher Untersuchungen beim Fräsen

Band 56
Dr.-Ing. Peter Uebelhoer

Inprocess-Geometriemessung beim Honen

Band 57
Dr.-Ing. Hans-Joachim Schelberg

Objektorientierte Projektierung von SPS-Software

Band 58
Dr.-Ing. Klaus Boes

Integration der Qualitätsentwicklung in featurebasierte CAD/CAM-Prozessketten

Band 59
Dr.-Ing. Martin Schreiber

Wirtschaftliche Investitionsbewertung komplexer Produktionssysteme unter Berücksichtigung von Unsicherheit

Band 60
Dr.-Ing. Ralf Steuernagel

Offenes adaptives Engineering-Werkzeug zur automatisierten Erstellung von entscheidungsunterstützenden Informationssystemen

Band 62
Dr.-Ing. Uwe Schauer

Qualitätsorientierte Feinbearbeitung mit Industrierobotern: Regelungsansatz für die Freiformflächenfertigung des Werkzeug- und Formenbaus

Band 63
Dr.-Ing. Simone Loeper

Kennzahlengestütztes Beratungssystem zur Verbesserung der Logistikleistung in der Werkstattfertigung

Band 64
Dr.-Ing. Achim Raab

Räumen mit hartstoffbeschichteten HSS-Werkzeugen

Band 65,
Dr.-Ing. Jan Erik Burghardt

Unterstützung der NC-Verfahrenskette durch ein bearbeitungselementorientiertes, lernfähiges Technologieplanungssystem

Band 66
Dr.-Ing. Christian Tritsch

Flexible Demontage technischer Gebrauchsgüter: Ansatz zur Planung und (teil-)automatisierten Durchführung industrieller Demontageprozesse

Band 67
Dr.-Ing. Oliver Eitrich

Prozessorientiertes Kostenmodell für die entwicklungsbegleitende Vorkalkulation

Band 68
Dr.-Ing. Oliver Wilke

Optimierte Antriebskonzepte für Räummaschinen - Potentiale zur Leistungssteigerung

Band 69
Dr.-Ing. Thilo Sieth

Rechnergestützte Modellierungsmethodik zerspantechnologischer Prozesse

Band 70
Dr.-Ing. Jan Linnenbuerger

Entwicklung neuer Verfahren zur automatisierten Erfassung der geometrischen Abweichungen an Linearachsen und Drehschwenkköpfen

Band 71
Dr.-Ing. Mathias Klimmek

Fraktionierung technischer Produkte mittels eines frei beweglichen Wasserstrahlwerkzeuges

Band 72
Dr.-Ing. Marko Hartel

Kennzahlenbasiertes Bewertungssystem zur Beurteilung der Demontage- und Recyclingeignung von Produkten

Band 73
Dr.-Ing. Jörg Schaupp

Wechselwirkung zwischen der Maschinen- und Hauptspindelantriebsdynamik und dem Zerspanprozess beim Fräsen

Band 74
Dr.-Ing. Bernhard Neisius

Konzeption und Realisierung eines experimentellen Telemanipulators für die Laparoskopie

Band 75
Dr.-Ing. Wolfgang Walter

Erfolgsversprechende Muster für betriebliche Ideenfindungsprozesse. Ein Beitrag zur Steigerung der Innovationsfähigkeit

Band 76
Dr.-Ing. Julian Weber

Ein Ansatz zur Bewertung von Entwicklungsergebnissen in virtuellen Szenarien

Band 77
Dr.-Ing. Dipl. Wirtsch.-Ing. Markus Posur

Unterstützung der Auftragsdurchsetzung in der Fertigung durch Kommunikation über mobile Rechner

Band 78
Dr.-Ing. Frank Fleissner

Prozessorientierte Prüfplanung auf Basis von Bearbeitungsobjekten für die Kleinserienfertigung am Beispiel der Bohr- und Fräsbearbeitung

Band 79
Dr.-Ing. Anton Haberkern

Leistungsfähigere Kugelgewindetriebe durch Beschichtung

Band 80
Dr.-Ing. Dominik Matt

Objektorientierte Prozess- und Strukturinnovation (OPUS)

Band 81
Dr.-Ing. Jürgen Andres

Robotersysteme für den Wohnungsbau: Beitrag zur Automatisierung des Mauerwerkabaus und der Elektroinstallation auf Baustellen

Band 82
Dr.-Ing. Dipl.Wirtschaftsing. Simone Riedmiller

Der Prozesskalender - Eine Methodik zur marktorientierten Entwicklung von Prozessen

Band 83
Dr.-Ing. Dietmar Tilch

Analyse der Geometrieparameter von Präzisionsgewinden auf der Basis einer Least-Squares-Estimation

Band 84
Dr.-Ing. Dipl.-Kfm. Oliver Stiefbold

Konzeption eines reaktionsschnellen Planungssystems für Logistikketten auf Basis von Software-Agenten

Band 85
Dr.-Ing. Ulrich Walter

Einfluss von Kühlschmierstoff auf den Zerspanprozess beim Fräsen: Beitrag zum Prozessverständniss auf Basis von zerspantechnischen Untersuchungen

Band 86
Dr.-Ing. Bernd Werner

Konzeption von teilautonomer Gruppenarbeit unter Berücksichtigung kultureller Einflüsse

Band 87
Dr.-Ing. Ulf Osmers

Projektieren Speicherprogrammierbarer Steuerungen mit Virtual Reality

Band 88
Dr.-Ing. Oliver Doerfel

Optimierung der Zerspantechnik beim Fertigungsverfahren Wälzstossen: Analyse des Potentials zur Trockenbearbeitung

Band 89
Dr.-Ing. Peter Baumgartner

Stufenmethode zur Schnittstellengestaltung in der internationalen Produktion

Band 90
Dr.-Ing. Dirk Vossmann

Wissensmanagement in der Produktentwicklung durch Qualitäts-methodenverbund und Qualitätsmethodenintegration

Band 91
Dr.-Ing. Martin Plass

Beitrag zur Optimierung des Honprozesses durch den Aufbau einer Honprozessregelung

Band 92
Dr.-Ing. Titus Konold

Optimierung der Fünfachsfräsbearbeitung durch eine kennzahlen-unterstützte CAM-Umgebung

Band 93
Dr.-Ing. Jürgen Brath

Unterstützung der Produktionsplanung in der Halbleiterfertigung durch risikoberücksichtigende Betriebskennlinien

Band 94
Dr.-Ing. Dirk Geisinger

Ein Konzept zur marktorientierten Produktentwicklung

Band 95
Dr.-Ing. Marco Lanza

Entwurf der Systemunterstützung des verteilten Engineering mit Axiomatic Design

Band 96
Dr.-Ing. Volker Hüntrup

Untersuchungen zur Mikrostrukturierbarkeit von Stählen durch das Fertigungsverfahren Fräsen

Band 97
Dr.-Ing. Frank Reinboth

Interne Stützung zur Genauigkeitsverbesserung in der Inertialmesstechnik: Beitrag zur Senkung der Anforderungen an Inertialsensoren

Band 98
Dr.-Ing. Lutz Trender

Entwicklungsintegrierte Kalkulation von Produktlebenszykluskosten auf Basis der ressourcenorientierten Prozesskostenrechnung

Band 99
Dr.-Ing. Cornelia Kafka

Konzeption und Umsetzung eines Leitfadens zum industriellen Einsatz von Data-Mining

Band 100
Dr.-Ing. Gebhard Selinger

Rechnerunterstützung der informellen Kommunikation in verteilten Unternehmensstrukturen

Band 101
Dr.-Ing. Thomas Windmüller

Verbesserung bestehender Geschäftsprozesse durch eine mitarbeiterorientierte Informationsversorgung

Band 102
Dr.-Ing. Knud Lembke

Theoretische und experimentelle Untersuchung eines bistabilen elektrohydraulischen Linearantriebs

Band 103
Dr.-Ing. Ulrich Thies

Methode zur Unterstützung der variantengerechten Konstruktion von industriell eingesetzten Kleingeräten

Band 104
Dr.-Ing. Andreas Schmälzle

Bewertungssystem für die Generalüberholung von Montageanlagen –Ein Beitrag zur wirtschaftlichen Gestaltung geschlossener Facility- Managment-Systeme im Anlagenbau

Band 105
Dr.-Ing. Thorsten Frank

Vergleichende Untersuchungen schneller elektromechanischer Vorschubachsen mit Kugelgewindetrieb

Band 106
Dr.-Ing. Achim Agostini

Reihenfolgeplanung unter Berücksichtigung von Interaktionen: Beitrag zur ganzheitlichen Strukturierung und Verarbeitung von Interaktionen von Bearbeitungsobjekten

Band 107
Dr.-Ing. Thomas Barrho

Flexible, zeitfenstergesteuerte Auftragseinplanung in segmentierten Fertigungsstrukturen

Band 108
Dr.-Ing. Michael Scharer

Quality Gate-Ansatz mit integriertem Risikomanagement

Band 109
Dr.-Ing. Ulrich Suchy

Entwicklung und Untersuchung eines neuartigen Mischkopfes für das Wasser Abrasivstrahlschneiden

Band 110
Dr.-Ing. Sellal Mussa

Aktive Korrektur von Verlagerungsfehlern in Werkzeugmaschinen

Band 111
Dr.-Ing. Andreas Hühsam

Modellbildung und experimentelle Untersuchung des Wälzschälprozesses

Band 112
Dr.-Ing. Axel Plutowsky

Charakterisierung eines optischen Messsystems und den Bedingungen des Arbeitsraums einer Werkzeugmaschine

Band 113
Dr.-Ing. Robert Landwehr

Konsequent dezentralisierte Steuerung mit Industrial Ethernet und offenen Applikationsprotokollen

Band 114
Dr.-Ing. Christoph Dill

Turbulenzreaktionsprozesse

Band 115
Dr.-Ing. Michael Baumeister

Fabrikplanung im turbulenten Umfeld

Band 116
Dr.-Ing. Christoph Gönnheimer

Konzept zur Verbesserung der Elektromagnetischen Verträglichkeit (EMV) in Produktionssystemen durch intelligente Sensor/Aktor-Anbindung

Band 117
Dr.-Ing. Lutz Demuß

Ein Reifemodell für die Bewertung und Entwicklung von Dienstleistungs-organisationen: Das Service Management Maturity Modell (SMMM)

Band 118
Dr.-Ing. Jörg Söhner

Beitrag zur Simulation zerspanungstechnologischer Vorgänge mit Hilfe der Finite-Element-Methode

Band 119
Dr.-Ing. Judith Elsner

Informationsmanagement für mehrstufige Mikro-Fertigungsprozesse

Band 120
Dr.-Ing. Lijing Xie

Estimation Of Two-dimension Tool Wear Based On Finite Element Method

Band 121
Dr.-Ing. Ansgar Blessing

Geometrischer Entwurf mikromechatronischer Systeme

Band 122
Dr.-Ing. Rainer Ebner

Steigerung der Effizienz mehrachsiger Fräsprozesse durch neue Planungsmethoden mit hoher Benutzerunterstützung

Band 123
Dr.-Ing. Silja Klinkel

Multikriterielle Feinplanung in teilautonomen Produktionsbereichen – Ein Beitrag zur produkt- und prozessorientierten Planung und Steuerung

Band 124
Dr.-Ing. Wolfgang Neithardt

Methodik zur Simulation und Optimierung von Werkzeugmaschinen in der Konzept- und Entwurfsphase auf Basis der Mehrkörpersimulation

Band 125
Dr.-Ing. Andreas Mehr

Hartfeinbearbeitung von Verzahnungen mit kristallinen diamantbeschichteten Werkzeugen beim Fertigungsverfahren Wälzstoßen

Band 126
Dr.-Ing. Martin Gutmann

Entwicklung einer methodischen Vorgehensweise zur Diagnose von hydraulischen Produktionsmaschinen

Band 127
Dr.-Ing. Gisela Lanza

Simulative Anlaufunterstützung auf Basis der Qualitätsfähigkeiten von Produktionsprozessen

Band 128
Dr.-Ing. Ulf Dambacher

Kugelgewindetrieb mit hohem Druckwinkel

Band 129
Dr.-Ing. Carsten Buchholz

Systematische Konzeption und Aufbau einer automatisierten Produktionszelle für pulverspritzgegossene Mikrobauteile

Band 130
Dr.-Ing. Heiner Lang

Trocken-Räumen mit hohen Schnittgeschwindigkeiten

Band 131
Dr.-Ing. Daniel Nesges

Prognose operationeller Verfügbarkeiten von Werkzeugmaschinen unter Berücksichtigung von Serviceleistungen

Im Shaker Verlag erschienene Bände:

Band 132
Dr.-Ing. Andreas Bechle

Beitrag zur prozesssicheren Bearbeitung beim Hochleistungs-fertigungsverfahren Wälzschälen

Band 133
Dr.-Ing. Markus Herm

Konfiguration globaler Wertschöpfungsnetzwerke auf Basis von Business Capabilities

Band 134
Dr.-Ing. Hanno Tritschler

Werkzeug- und Zerspanprozessoptimierung beim Hartfräsen von Mikrostrukturen in Stahl

Band 135
Dr.-Ing. Christian Munzinger

Adaptronische Strebe zur Steifigkeitssteigerung von Werkzeugmaschinen

Band 136
Dr.-Ing. Andreas Stepping

Fabrikplanung im Umfeld von Wertschöpfungsnetzwerken und ganzheitlichen Produktionssystemen

Band 137
Dr.-Ing. Martin Dyck

Beitrag zur Analyse thermische bedingter Werkstückdeformationen in Trockenbearbeitungsprozessen

Band 138
Dr.-Ing. Siegfried Schmalzried

Dreidimensionales optisches Messsystem für eine effizientere geometrische Maschinenbeurteilung

Band 139
Dr.-Ing. Marc Wawerla

Risikomanagement von Garantieleistungen

Band 140
Dr.-Ing. Ivesa Buchholz

Strategien zur Qualitätssicherung mikromechanischer Bauteile mittels multisensorieller Koordinatenmesstechnik

Band 141
Dr.-Ing. Jan Kotschenreuther

Empirische Erweiterung von Modellen der Makrozerspanung auf den Bereich der Mikrobearbeitung

Band 142
Dr.-Ing. Andreas Knödel

Adaptronische hydrostatische Drucktascheneinheit

Band 143
Dr.-Ing. Gregor Stengel

Fliegendes Abtrennen räumlich gekrümmter Strangpressprofile mittels Industrierobotern

Band 144
Dr.-Ing. Udo Weismann

Lebenszyklusorientiertes interorganisationelles Anlagencontrolling

Band 145
Dr.-Ing. Rüdiger Pabst

Mathematische Modellierung der Wärmestromdichte zur Simulation des thermischen Bauteilverhaltens bei der Trockenbearbeitung

Band 146
Dr.-Ing. Jan Wieser

Intelligente Instandhaltung zur Verfügbarkeitssteigerung von Werkzeugmaschinen

Band 147
Dr.-Ing. Sebastian Haupt

Effiziente und kostenoptimale Herstellung von Mikrostrukturen durch eine Verfahrenskombination von Bahnerosion und Laserablation

Band 148
Dr.-Ing. Matthias Schlipf

Statistische Prozessregelung von Fertigungs- und Messprozess zur Erreichung einer variabilitätsarmen Produktion mikromechanischer Bauteile

Band 149
Dr.-Ing. Jan Philipp Schmidt-Ewig

Methodische Erarbeitung und Umsetzung eines neuartigen Maschinenkonzeptes zur produktflexiblen Bearbeitung räumlich gekrümmter Strangpressprofile

Band 150
Dr.-Ing. Thomas Ender

Prognose von Personalbedarfen im Produktionsanlauf unter Berücksichtigung dynamischer Planungsgrößen

Band 151
Dr.-Ing. Kathrin Peter

Bewertung und Optimierung der Effektivität von Lean Methoden in der Kleinserienproduktion

Band 152
Dr.-Ing. Matthias Schopp

Sensorbasierte Zustandsdiagnose und -prognose von Kugelgewindetrieben

Band 153
Dr.-Ing. Martin Kipfmüller

Aufwandsoptimierte Simulation von Werkzeugmaschinen

Band 154
Dr.-Ing. Carsten Schmidt

Development of a database to consider multi wear mechanisms within chip forming simulation

Band 155
Dr.-Ing. Stephan Niggeschmidt

Ausfallgerechte Ersatzteilbereitstellung im Maschinen- und Anlagenbau mittels lastabhängiger Lebensdauerprognose

Band 156
Dr.-Ing. Jochen Conrad Peters

Bewertung des Einflusses von Formabweichungen in der Mikro-Koordinatenmesstechnik

Band 157
Dr.-Ing. Jörg Ude

Entscheidungsunterstützung für die Konfiguration globaler Wertschöpfungsnetzwerke

Band 158
Dr.-Ing. Stefan Weiler

Strategien zur wirtschaftlichen Gestaltung der globalen Beschaffung

Band 159
Dr.-Ing. Jan Rühl

Monetäre Flexibilitäts- und Risikobewertung

Band 160
Dr.-Ing. Daniel Ruch

Positions- und Konturerfassung räumlich gekrümmter Profile auf Basis bauteilimmanenter Markierungen

Band 161
Dr.-Ing. Manuel Tröndle

Flexible Zuführung von Mikrobauteilen mit piezoelektrischen Schwingförderern

Band 162
Dr.-Ing. Benjamin Viering

Mikroverzahnungsnormal

Band 163
Dr.-Ing. Chris Becke

Prozesskraftrichtungsangepasste Frässtrategien zur schädigungsarmen Bohrungsbearbeitung an faserverstärkten Kunststoffen

Band 164
Dr.-Ing. Patrick Werner

Dynamische Optimierung und Unsicherheitsbewertung der lastabhängigen präventiven Instandhaltung von Maschinenkomponenten

Band 165
Dr.-Ing. Martin Weis

Kompensation systematischer Fehler bei Werkzeugmaschinen durch self-sensing Aktoren

Band 166
Dr.-Ing. Markus Schneider

Kompensation von Konturabweichungen bei gerundeten Strangpressprofilen durch robotergestützte Führungswerkzeuge

Band 167
Dr.-Ing. Ester M. R. Ruprecht

Prozesskette zur Herstellung schichtbasierter Systeme mit integrierten Kavitäten

Band 168
Dr.-Ing. Alexander Broos

Simulationsgestützte Ermittlung der Komponentenbelastung für die Lebensdauerprognose an Werkzeugmaschinen

Band 169
Dr.-Ing. Frederik Zanger

Segmentspanbildung, Werkzeugverschleiß, Randschichtzustand und Bauteileigenschaften: Numerische Analysen zur Optimierung des Zerspanungsprozesses am Beispiel von Ti-6Al-4V

Band 170
Dr.-Ing. Benjamin Behmann

Servicefähigkeit

Band 171
Dr.-Ing. Annabel Gabriele Jondral

Simulationsgestützte Optimierung und Wirtschaftlichkeitsbewertung des Lean-Methodeneinsatzes

Band 172
Dr.-Ing. Christoph Ruhs

Automatisierte Prozessabfolge zur qualitätssicheren Herstellung von Kavitäten mittels Mikrobahnerosion

Band 173
Dr.-Ing. Steven Peters

Markoffsche Entscheidungsprozesse zur Kapazitäts- und Investitionsplanung von Produktionssystemen

Band 174
Dr.-Ing. Christoph Kühlewein

Untersuchung und Optimierung des Wälzschälverfahrens mit Hilfe von 3D-FEM-Simulation – 3D-FEM Kinematik- und Spanbildungssimulation

Band 175
Dr.-Ing. Adam-Mwanga Dieckmann

Auslegung und Fertigungsprozessgestaltung sintergefügter Verbindungen für µMIM-Bauteile

Band 176
Dr.-Ing. Heiko Hennrich

Aufbau eines kombinierten belastungs- und zustandsorientierten Diagnose- und Prognosesystems für Kugelgewindetriebe

Band 177
Dr.-Ing. Stefan Herder

Piezoelektrischer Self-Sensing-Aktor zur Vorspannungsregelung in adaptronischen Kugelgewindetrieben

Band 178
Dr.-Ing. Alexander Ochs

Ultraschall-Strömungsgreifer für die Handhabung textiler Halbzeuge bei der automatisierten Fertigung von RTM-Bauteilen

Band 179
Dr.-Ing. Jürgen Michna

Numerische und experimentelle Untersuchung zerspanungsbedingter Gefügeumwandlungen und Modellierung des thermo-mechanischen Lastkollektivs beim Bohren von 42CrMo4

Band 180
Dr.-Ing. Jörg Elser

Vorrichtungsfreie räumliche Anordnung von Fügepartnern auf Basis von Bauteilmarkierungen

Band 181
Dr.-Ing. Katharina Klimscha

Einfluss des Fügespalts auf die erreichbare Verbindungsqualität beim Sinterfügen

Band 182
Dr.-Ing. Patricia Weber

Steigerung der Prozesswiederholbarkeit mittels Analyse akustischer Emissionen bei der Mikrolaserablation mit UV-Pikosekundenlasern

Band 183
Dr.-Ing. Jochen Schädel

Automatisiertes Fügen von Tragprofilen mittels Faserwickeln

Band 184
Dr.-Ing. Martin Krauße

Aufwandsoptimierte Simulation von Produktionsanlagen durch Vergrößerung der Geltungsbereiche von Teilmodellen

Band 185
Dr.-Ing. Raphael Moser

Strategische Planung globaler Produktionsnetzwerke
Bestimmung von Wandlungsbedarf und Wandlungszeitpunkt mittels multikriterieller Optimierung

Band 186
Dr.-Ing. Martin Otter

Methode zur Kompensation fertigungsbedingter Gestaltabweichungen für die Montage von Aluminium Space-Frame-Strukturen

Band 187
Dr.-Ing. Urs Leberle

Produktive und flexible Gleitförderung kleiner Bauteile auf phasenflexiblen Schwingförderern mit piezoelektrischen 2D-Antriebselementen

Band 188
Dr.-Ing. Johannes Book

Modellierung und Bewertung von Qualitätsmanagementstrategien in globalen Wertschöpfungsnetzwerken

Band 189
Dr.-Ing. Florian Ambrosy

Optimierung von Zerspanungsprozessen zur prozesssicheren Fertigung nanokristalliner Randschichten am Beispiel von 42CrMo4

Band 190
Dr.-Ing. Adrian Kölmel

Integrierte Messtechnik für Prozessketten unreifer Technologien am Beispiel der Batterieproduktion für Elektrofahrzeuge

Band 191
Dr.-Ing. Henning Wagner

Featurebasierte Technologieplanung zum Preforming von textilen Halbzeugen

Band 192
Dr.-Ing. Johannes Gebhardt

Strukturoptimierung von in FVK eingebetteten metallischen Lasteinleitungselementen

Band 193
Dr.-Ing. Jörg Bauer

Hochintegriertes hydraulisches Vorschubsystem für die Bearbeitung kleiner Werkstücke mit hohen Fertigungsanforderungen

Band 194
Dr.-Ing. Nicole Stricker

Robustheit verketteter Produktionssysteme
Robustheitsevaluation und Selektion des Kennzahlensystems der Robustheit

Band 195
Dr.-Ing. Anna Sauer

Konfiguration von Montagelinien unreifer Produkttechnologien am Beispiel der Batteriemontage für Elektrofahrzeuge

Band 196
Dr.-Ing. Florian Sell-Le Blanc

Prozessmodell für das Linearwickeln unrunder Zahnspulen
Ein Beitrag zur orthozyklischen Spulenwickeltechnik

Band 197
Dr.-Ing. Frederic Förster

Geregeltes Handhabungssystem zum zuverlässigen und energieeffizienten Handling textiler Kohlenstofffaserzuschnitte

Band 198
Dr.-Ing. Nikolay Boev

Numerische Beschreibung von Wechselwirkungen zwischen Zerspanprozess und Maschine am Beispiel Räumen

Band 199
Dr.-Ing. Sebastian Greinacher

Simulationsgestützte Mehrzieloptimierung schlanker und ressourceneffizienter Produktionssysteme

Band 200
Dr.-Ing. Benjamin Häfner

Lebensdauerprognose in Abhängigkeit der Fertigungsabweichungen bei Mikroverzahnungen

Band 201
Dr.-Ing. Stefan Klotz

Dynamische Parameteranpassung bei der Bohrungsherstellung in faserverstärkten Kunststoffen unter zusätzlicher Berücksichtigung der Einspannsituation

Band 202
Dr.-Ing. Johannes Stoll

Bewertung konkurrierender Fertigungsfolgen mittels Kostensimulation und stochastischer Mehrzieloptimierung
Anwendung am Beispiel der Blechpaketfertigung für automobile Elektromotoren

Band 203
Dr.-Ing. Simon-Frederik Koch

Fügen von Metall-Faserverbund-Hybridwellen im Schleuderverfahren
ein Beitrag zur fertigungsgerechten intrinsischen Hybridisierung

Band 204
Dr.-Ing. Julius Ficht

Numerische Untersuchung der Eigenspannungsentwicklung für sequenzielle Zerspanungsprozesse

Band 205
Dr.-Ing. Manuel Baumeister

Automatisierte Fertigung von Einzelblattstapeln in der Lithium-Ionen-Zellproduktion

Band 206
Dr.-Ing. Daniel Bertsch

Optimierung der Werkzeug- und Prozessauslegung für das Wälzschälen von Innenverzahnungen

Band 207
Dr.-Ing. Kyle James Kippenbrock

Deconvolution of Industrial Measurement and Manufacturing Processes for Improved Process Capability Assessments

Band 208
Dr.-Ing. Farboud Bejnoud

Experimentelle Prozesskettenbetrachtung für Räumbauteile am Beispiel einer einsatzgehärteten PKW-Schiebemuffe

Band 209
Dr.-Ing. Steffen Dosch

Herstellungsübergreifende Informationsübertragung zur effizienten Produktion von Werkzeugmaschinen am Beispiel von Kugelgewindetrieben

Band 210
Dr.-Ing. Emanuel Moser

Migrationsplanung globaler Produktionsnetzwerke
Bestimmung robuster Migrationspfade und risiko-effizienter Wandlungsbefähiger

Band 211
Dr.-Ing. Jan Hochdörffer

Integrierte Produktallokationsstrategie und Konfigurationssequenz in globalen Produktionsnetzwerken

Band 212
Dr.-Ing. Tobias Arndt

Bewertung und Steigerung der Prozessqualität in globalen Produktionsnetzwerken

Band 213
Dr.-Ing. Manuel Peter

Unwuchtminimale Montage von Permanentmagnetrotoren durch modellbasierte Online-Optimierung

Band 214
Dr.-Ing. Robin Kopf

Kostenorientierte Planung von Fertigungsfolgen additiver Technologien

Band 215
Dr.-Ing. Harald Meier

Einfluss des Räumens auf den Bauteilzustand in der Prozesskette Weichbearbeitung – Wärmebehandllung – Hartbearbeitung

Band 216
Dr.-Ing. Daniel Brabandt

Qualitätssicherung von textilen Kohlenstofffaser-Preforms mittels optischer Messtechnik

Band 217
Dr.-Ing. Alexandra Schabunow

Einstellung von Aufnahmeparametern mittels projektionsbasierter Qualitätskenngrößen in der industriellen Röntgen-Computertomographie

Band 218
Dr.-Ing. Jens Bürgin

Robuste Auftragsplanung in Produktionsnetzwerken
Mittelfristige Planung der variantenreichen Serienproduktion unter Unsicherheit der Kundenauftragskonfigurationen

Band 219
Dr.-Ing. Michael Gerstenmeyer

Entwicklung und Analyse eines mechanischen Oberflächenbehandlungsverfahrens unter Verwendung des Zerspanungswerkzeuges

Band 220
Dr.-Ing. Jacques Burtscher

Erhöhung der Bearbeitungsstabilität von Werkzeugmaschinen durch semi-passive masseneinstellbare Dämpfungssysteme

Band 221
Dr.-Ing. Dietrich Berger

Qualitätssicherung von textilen Kohlenstofffaser-Preforms mittels prozessintegrierter Wirbelstromsensor-Arrays

Band 222
Dr.-Ing. Fabian Johannes Ballier

Systematic gripper arrangement for a handling device in lightweight production processes

Band 223
Dr.-Ing. Marielouise Schäferling, geb. Zaiß

Development of a Data Fusion-Based Multi-Sensor System for Hybrid Sheet Molding Compound

Band 224
Dr.-Ing. Quirin Spiller

Additive Herstellung von Metallbauteilen mit dem ARBURG Kunststoff-Freiformen

Band 225
Dr.-Ing. Andreas Spohrer

Steigerung der Ressourceneffizienz und Verfügbarkeit von Kugelgewindetrieben durch adaptive Schmierung

Band 226
Dr.-Ing. Johannes Fisel

Veränderungsfähigkeit getakteter Fließmontagesysteme
Planung der Fließbandabstimmung am Beispiel der Automobilmontage

Band 227
Dr.-Ing. Patrick Bollig

Numerische Entwicklung von Strategien zur Kompensation thermisch bedingter Verzüge beim Bohren von 42CrMo4

Band 228
Dr.-Ing. Ramona Pfeiffer, geb. Singer

Untersuchung der prozessbestimmenden Größen für die anforderungsgerechte Gestaltung von Pouchzellen-Verpackungen

Band 229
Dr.-Ing. Florian Baumann

Additive Fertigung von endlosfaserverstärkten Kunststoffen mit dem ARBURG Kunststoff-Freiform Verfahren

Band 230
Dr.-Ing. Tom Stähr

Methodik zur Planung und Konfigurationsauswahl skalierbarer Montagesysteme – Ein Beitrag zur skalierbaren Automatisierung

Band 231
Dr.-Ing. Jan Schwennen

Einbringung und Gestaltung von Lasteinleitungsstrukturen für im RTM-Verfahren hergestellte FVK-Sandwichbauteile

Band 232
Dr.-Ing. Sven Coutandin

Prozessstrategien für das automatisierte Preforming von bebinderten textilen Halbzeugen mit einem segmentierten Werkzeugsystem

Band 233
Dr.-Ing. Christoph Liebrecht

Entscheidungsunterstützung für den Industrie 4.0-Methodeneinsatz
Strukturierung, Bewertung und Ableitung von Implementierungsreihenfolgen

Band 234
Dr.-Ing. Stefan Treber

Transparenzsteigerung in Produktionsnetzwerken
Verbesserung des Störungsmanagements durch verstärkten Informationsaustausch

Band 235
Dr.-Ing. Marius Dackweiler

Modellierung des Fügewickelprozesses zur Herstellung von leichten Fachwerkstrukturen

Band 236
Dr.-Ing. Fabio Echsler Minguillon

Prädiktiv-reaktives Scheduling zur Steigerung der Robustheit in der Matrix-Produktion

Band 237
Dr.-Ing. Sebastian Haag

Entwicklung eines Verfahrensablaufes zur Herstellung von Batteriezellstapeln mit großformatigem, rechteckigem Stapelformat und kontinuierlichen Materialbahnen

Band 238
Dr.-Ing. Raphael Wagner

Strategien zur funktionsorientierten Qualitätsregelung in der Serienproduktion

Band 239
Dr.-Ing. Christopher Ehrmann

Ausfallfrüherkennung von Ritzel-Zahnstangen- Trieben mittels Acoustic Emission

Band 240
Dr.-Ing. Janna Hofmann

Prozessmodellierung des Fünf-Achs-Nadelwickelns zur Implementierung einer trajektoriebasierten Drahtzugkraftregelung

Band 241
Dr.-Ing. Andreas Kuhnle

Adaptive Order Dispatching based on Reinforcement Learning
Application in a Complex Job Shop in the Semiconductor Industry

Band 242
Dr.-Ing. Andreas Greiber

Fertigung optimierter technischer Oberflächen durch eine Verfahrenskombination aus Fliehkraft-Tauchgleitschleifen und Laserablation
Prozesseinflüsse und Prozessauslegung

Band 243
Dr.-Ing. Jan Niclas Eschner

Entwicklung einer akustischen Prozessüberwachung zur Porenbestimmung im Laserstrahlschmelzen

Band 244
Dr.-Ing. Sven Roth

Schädigungsfreie Anbindung von hybriden FVK/Metall-Bauteilen an metallische Tragstrukturen durch Widerstandspunktschweißen

Band 245
Dr.-Ing. Sina Kathrin Peukert

Robustheitssteigerung in Produktionsnetzwerken mithilfe eines integrierten Störungsmanagements

Band 246
Dr.-Ing. Alexander Jacob

Hochiterative Technologieplanung
Rekursive Optimierung produkt- und fertigungsbezogener Freiheitsgrade am Beispiel der hybrid-additiven Fertigung

Band 247
Dr.-Ing. Patrick Moll

Ressourceneffiziente Herstellung von Langfaser-Preforms im Faserblasverfahren

Band 248
Dr.-Ing. Eric Thore Segebade

Erhöhung der Verschleißbeständigkeit von Bauteilen aus Ti-6Al-4V mittels simulationsgestützer Zerspanung und mechanischer Mikrotexturierung

Band 249
Dr.-Ing. Shun Yang

Regionalized implementation strategy of smart automation within assembly systems in China

Band 250
Dr.-Ing. Constantin Carl Hofmann

Vorausschauende und reaktive Mehrzieloptimierung für die Produktionssteuerung einer Matrixproduktion

Band 251
Dr.-Ing. Paul Ruhland

Prozesskette zur Herstellung von hybriden Faser-Metall-Preforms
Modellbildung und Optimierung des Binderauftrags und der Drapierung für stabförmige Bauteile

Band 252
Dr.-Ing. Leonard Schild

Erzeugung und Verwendung von Anwendungswissen in der industriellen Computertomographie

Band 253
Dr.-Ing. Benedikt Klee

Analyse von Phaseninformationen in Videodaten zur Identifikation von Schwingungen in Werkzeugmaschinen

Band 254
Dr.-Ing. Bruno Vargas

Wälzschälen mit kleinen Achskreuzwinkeln
Prozessgrenzen und Umsetzbarkeit

Band 255
Dr.-Ing. Lucas Bretz

Function-oriented in-line quality assurance of hybrid sheet molding compound

Band 256
Dr.-Ing. Bastian Rothaupt

Dämpfung von Bauteilschwingungen durch einstellbare Werkstückdirektspannung mit Hydrodehnspanntechnik

Band 257
Dr.-Ing. Daniel Kupzik

Robotic Swing Folding of three-dimensional UD-tape-based Reinforcement Structures

Band 258
Dr.-Ing. Bastian Verhaelen

(De-)Zentralisierung von Entscheidungen in globalen Produktionsnetzwerken
Strategie- und komplexitätsorientierte Gestaltung der Entscheidungsautonomie

Band 259
Dr.-Ing. Hannes Wilhelm Weinmann

Integration des Vereinzelungs- und Stapelbildungsprozesses in ein flexibel und kontinuierlich arbeitendes Anlagenmodul für die Li-Ionen Batteriezellfertigung

Band 260
Dr.-Ing. Florian Stamer

Dynamische Lieferzeit-Preisgestaltung in variantenreicher Produktion
Ein adaptiver Ansatz mithilfe von Reinforcement Learning

Band 261
Dr.-Ing. Patrick Neuenfeldt

Modellbildung des Tauchgleitschleifens zur Abtrag- und Topografievorhersage an komplexen Geometrien

Band 262
Dr.-Ing. Boris Matuschka

Energieeffizienz in Prozessketten: Analyse und Optimierung von Energieflüssen bei der Herstellung eines PKW-Getriebebauteils aus 16MnCr5

Band 263
Dr.-Ing. Tobias Schlagenhauf

Bildbasierte Quantifizierung und Prognose des Verschleißes an Kugelgewindetriebspindeln
Ein Beitrag zur Zustandsüberwachung von Kugelgewindetrieben mittels Methoden des maschinellen Lernens

Band 264
Dr.-Ing. Benedict Stampfer

Entwicklung eines multimodalen Prozessmodells zur Oberflächenkonditionierung beim Außenlängsdrehen von 42CrMo4

Band 265
Dr.-Ing. Carmen Maria Krahe

KI-gestützte produktionsgerechte Produktentwicklung
Automatisierte Wissensextraktion aus vorhandenen Produktgenerationen

Band 266
Dr.-Ing. Markus Netzer

Intelligente Anomalieerkennung für hochflexible Produktionsmaschinen
Prozessüberwachung in der Brownfield Produktion

Band 267
Dr.-Ing. Simon Raphael Merz

Analyse der Kinematik und Kinetik von Planetenwälzgewindetrieben

Band 268
Dr.-Ing. Rainer Maria Silbernagel

Funktionsorientierte Qualitätsregelung in Produktionsnetzwerken
Qualitätsmanagement in der Produktion hochpräziser Produkte durch netzwerkweite Datenintegration

Band 269
Dr.-Ing. Jonas Nieschlag

Gestaltung und Prozessanalyse für im Schleuderverfahren hergestellte FKV-Metall-Hohlstrukturen

Band 270
Dr.-Ing. Lukas Matthias Weiser

In-Process Porositätserkennung für den PBF-LB/M-Prozess

Band 271
Dr.-Ing. Leonard Vincent Overbeck

Digital Twins of production systems
Automated validation and update of material flow simulation models with real data

Band 272
Dr.-Ing. Felix Klenk

Transparenzsteigerung in der Rückführungslogistik zur Verbesserung der Materialbedarfsplanung für das Remanufacturing

Band 273
Dr.-Ing. Benjamin Bold

Kompensation der Wrinkle-Bildung beim Kalandrieren von Lithium-Ionen-Kathoden
Vom Prozessverständnis des Kalandrierens bis zur Prozessoptimierung mittels Anti-Wrinkle-Modul

Band 274
Dr.-Ing. Daniel Gauder

Adaptive in-line Qualitätsregelung in der Mikro-Verzahnungsfertigung

Band 275
Dr.-Ing. Fabian Sasse

Ontologie-basierte Entscheidungsunterstützung für die Auswahl von Messsystemen in unreifen Produktionsprozessen

Band 276
Dr.-Ing. Jonas Hillenbrand

Unsupervised Condition-Monitoring für Kugelgewindetriebe mittels Acoustic Emission

Band 277
Dr.-Ing. Manuela Neuenfeldt

Untersuchung des Einflusses der PBF-LB-Stellgrößen auf die zerspanende Bearbeitung additiv gefertigter Stahlbauteile

Band 278
Dr.-Ing. Marvin Carl May

Intelligent production control for time-constrained complex job shops

Band 279
Dr.-Ing. Philipp Gönnheimer

Automatisierte Bereitstellung von Maschinensteuerungsdaten in Brownfield-Produktionssystemen
Ein Beitrag zur Digitalisierung von Bestandsanlagen am Beispiel von Werkzeugmaschinen

Band 280
Dr.-Ing. Markus Schäfer

Kollisionsvermeidung für Endeffektoren mit integriertem LiDAR-System in der MRK
Ein Beitrag zur Mensch-Roboter-Kollaboration

Band 281
Dr.-Ing. Oliver Brützel

Decision Support System for the Optimisation of Global Production Networks
Development of a Digital Twin for Product Allocation and Robust Line Configuration

Band 282
Dr.-Ing. Gregor Graf

Qualifizierung der Legierung FeNiCoMoVTiAl im LPBF-Prozess unter Verwendung einer Doppellaser-Belichtungsstrategie

Band 283
Dr.-Ing. Maximilian Torsten Halwas

Kompaktwickelprozess zur Erhöhung der Performance von Statoren elektrischer Traktionsantriebe

Band 284
Dr.-Ing. Magnus Kandler

Menschzentriertes Implementierungsvorgehen für das Digitale Shopfloor Management - Förderung der Selbstorganisation unter Berücksichtigung der Mitarbeiterakzeptanz

Band 285
Dr.-Ing. Michael Baranowski

Additive Herstellung endlosfaserverstärkter Kunststoffbauteile mit dem Laser-Sinterprozess
Maschinentechnik, Prozessentwicklung und -modellierung

Band 286
Dr.-Ing. Tobias Storz

Flexibel automatisierte Assemblierung von Li-Ionen-Pouchzellen
Agile Anlagentechnik für die Prozesskette Stapelbildung, Kontaktierung und Heißsiegeln

Band 287
Dr.-Ing. Nikolas Sven Matkovic

Additive Individualization of Continuous-Discontinuous Reinforced Thermoplastics

Band 288
Dr.-Ing. Marco Wurster

Planung und Steuerung agiler hybrider Demontagesysteme im Remanufacturing

Band 289
Dr.-Ing. Felix Johannes Wirth

Prozessgeregelte Formgebung von Hairpin-Steckspulen für elektrische Traktionsmotoren

Band 290
Dr.-Ing. Patrizia Konstanze Gartner

Konzept eines Selbstheilungsmechanismus für Polymerelektrolytmembranen
Optimierung der Lebensdauer und der Effizienz von Brennstoffzellen

Band 291

Dr.-Ing. Jens Schäfer

Funktionsintegriertes Handhabungssystem zur geometrieflexiblen, positionsgenauen Einzellagenstapelung in der Brennstoffzellenstackfertigung

Band 292
Dr.-Ing. Gwen Louis Steier

Strategischer Fit in globalen Produktionsnetzwerken
Entscheidungsunterstützung für die strategische Netzwerkkonfiguration

Band 293

Dr.-Ing. Louis Schäfer

Assistierte, modellbasierte Grobplanung von Produktionssystemen mittels Mehrzieloptimierung:
Anwendung am Beispiel hochautomatisierter Schweißlinien für die Automobilzuliefererindustrie

Band 294
Dr.-Ing. Jan-Philipp Kaiser

Autonomous View Planning using Reinforcement Learning
Modeling and Application for Visual Inspection in Remanufacturing

Band 295
Dr.-Ing. Wilken Wößner

Identifikation und Reduktion der Ursachen von Unwuchtänderungen an Permanentmagnetrotoren elektrischer Traktionsantriebe

Band 296
Dr.-Ing. Ann-Kathrin Wurba

Reduktion der Längsfaltenbildung während des Kalandrierens von Batterieelektroden

Band 297
Dr.-Ing. Simon Mangold

Automatisierte Demontage von Schraubverbindungen für das Remanufacturing
Konzeption, Aufbau und Betrieb einer Demontagezelle

Band 298
Dr.-Ing. Eduard Gerlitz

Flexibles Trennen von Zellkontaktierungen in Lithium-Ionen-Batteriemodulen
Ein Beitrag zur automatisierten und flexiblen Demontage von Traktionsbatterien

Band 299
Dr.-Ing. Edgar Mühlbeier

Mechatronisches Koppelsystem für die prozessunabhängige, kraftgeregelte Kopplung von seriellen Roboterkinematiken

Band 300
Dr.-Ing. Martin Benfer

Decision-Making in Production Network Configuration
A Design Framework for Digital Twins of Global Production Networks

Band 301
Dr.-Ing. Victor Lubkowitz

Keramikverstärkte Aluminiumwerkstoffe für das pulverbettbasierte selektive Laserschmelzen
Beschichtung, Werkstoff- und Oberflächeneigenschaften von Mikro-B4C- und Nano-TiC-verstärkten AlSi10Mg-Feedstocks, verarbeitet im PBF-LB-Prozess

Band 302
Dr.-Ing. Alex Frey

Datenbasierte Erstellung und Überprüfung von Modellen zur Produkt- und Prozesskonfiguration

Band 303
Dr.-Ing. Johannes Schubert

Werkstoff- und Prozessanalyse zur Herstellung keramischer Werkstoffverbunde mittels badbasierter Photopolymerisation (VPP-LED)

Band 304
Dr.-Ing. Tassilo Arndt

Simulative und experimentelle Untersuchung des Rotationsunrunddrehens zur hochproduktiven Herstellung unrunder Bauteilquerschnitte

Band 305
Dr.-Ing. Kamal Husseini

Modellierung maschinenseitiger Einflüsse auf den Vereinzelungs- und Stapelbildungsprozess von Batteriezellen

Band 306
Dr.-Ing. Marco Friedmann

Automatisierte Konfiguration von Greiferfingern aus einem modularen Baukasten

Band 307
Dr.-Ing. Jannik Konstantin Schwalm

Hämmerndes Drehen
Entwicklung eines hybriden Prozesses zur hauptzeitparallelen mechanischen Oberflächenbehandlung und Oberflächentexturierung während der Zerspanung

Band 308
Dr.-Ing. Dominik Mayer

Analyse prozessübergreifender Wirkzusammenhänge zur Modellierung der Stapelbildung in der Batteriezellfertigung
Ein Beitrag zur Bewertung geometrischer Effekte in Elektroden auf die Stapelgenauigkeit

Band 309
Dr.-Ing. Markus Heim

Demontage von Magneten aus permanentmagneterregten Synchronmaschinen
Ressourceneffizienz durch Werterhaltung

Band 310
Dr.-Ing. Ludwig Benedikt Hausmann

Kinematisches Twisten von Statoren mit Hairpin-Wicklung für elektrische Traktionsmotoren

Band 311
Dr.-Ing. Kai Uwe Drechsel

Verarbeitung von agglomerierendem 316L Pulver im pulverbettbasierten Schmelzen mittels Laserstrahl
Entwicklung eines ultraschallangeregten Beschichtungsprozesses und Untersuchung der Prozessstabilität

Band 312
Dr.-Ing. Ali Bilen

Datengetriebene Qualitätsregelung in der Fertigung von Mikro-Kronenrädern
Körperschallbasierte virtuelle Messung zur Vorhersage geometrischer Qualitätsmerkmale mit Unsicherheitsquantifizierung

Band 313
Dr.-Ing. Sebastian Tim Behrendt

Service-oriented Production Planning and Control of Reconfigurable Manufacturing Systems