ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 38

Settore Concorsuale: 01/B1 - INFORMATICA

Settore Scientifico Disciplinare: INF/01 - INFORMATICA

TACTICAL ANALYSIS OF MANUFACTURING PROCESSES VIA SIMULATION

Presentata da: Marco Ferrati

Coordinatore Dottorato

Paola Salomoni

Supervisore

Angelo Di Iorio

Co-supervisore

Davide Rossi

Esame finale anno 2026

Abstract

This thesis aims to make tactical analysis of manufacturing processes more accessible and effective through simulation. It addresses the challenge of creating simulation models that are both easy to build and sufficiently expressive to represent realistic production dynamics. The focus is on enabling analysts with limited programming experience to evaluate manufacturing systems at the tactical level, including resource allocation, process reconfiguration, and throughput optimization.

The research proposes a hybrid activity- and resource-based simulation paradigm that combines the intuitive abstraction of Business Process Modeling with the operational detail of manufacturing simulation. This approach bridges conceptual modeling and executable simulation, allowing users to construct and analyze models through structured, model-driven specifications rather than code. The framework supports stochastic behavior, resource constraints, and inter-process dependencies, ensuring both usability and representational fidelity.

A comprehensive methodology is introduced, covering model definition, data-driven parameterization, and systematic experimentation to support tactical decision-making. Case studies show how the simulation environment enables exploration of alternative process configurations and performance trade-offs without requiring specialized programming skills. The results demonstrate that simulation-based tactical analysis helps connect operational detail with strategic intent in manufacturing. By supporting what-if analyses under uncertainty and variability, the framework transforms simulation into a practical decision-support tool and promotes experimentation and continuous improvement.

Beyond its immediate applications, the research establishes the foundation for a multi-model platform integrating data from multiple sources and levels of abstraction. Future extensions include linking forecasting and logistics data to align market demand with supply-chain constraints, incorporating transactional processes to capture information flow and decision delays, and integrating shop-floor sensor and IoT data for dynamic model and KPI updates. These directions support progressive model fidelity, component reuse, and consistent semantics, positioning the platform as a basis for data-centric manufacturing planning and future digital-twin ecosystems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Manufacturing systems are complex, dynamic environments where countless interdependent processes must operate in harmony to achieve efficiency, quality, and flexibility [3]. Even small changes, such as introducing a new machine, adjusting a workstation layout, or modifying a production schedule, can ripple across the entire production line, affecting costs, throughput, and delivery times [26]. Understanding these effects before implementing them in reality is crucial for decision-makers seeking to improve productivity while minimizing risks and expenses [41].

Simulation provides a powerful means to achieve this understanding. It enables the modeling of real-world processes and the exploration of what-if scenarios without requiring costly or disruptive physical experimentation [62]. In manufacturing, simulation allows analysts and managers to test the impact of process improvements, automation strategies, or new production configurations, supporting data-driven decisions in environments where trial and error would be prohibitively expensive or impractical [64, 79, 51]. As a result, simulation has become a cornerstone of modern manufacturing analysis and continuous improvement initiatives.

This work was motivated by a collaboration with a manufacturing consulting firm specializing in the implementation of lean principles across diverse industries. In practice, the firm's analysts want to rely on simulation to validate process changes before deploying them on the factory floor. However, a recurring challenge emerged: existing simulation tools, while powerful, are often too complex and technical for analysts without programming backgrounds. Most classical tools adopt a resource-centric modeling paradigm, focusing on physical entities such as machines, operators, and materials. While this representation offers high fidelity, it also demands extensive model configuration and detailed data, which can overwhelm non-technical users. Conversely, activity-centric approaches, such as those found in Business Process Management (BPM), are much easier to learn and use but typically lack the expressive power needed to represent manufacturing-specific behaviors and dependencies.

The main problem this thesis addresses is how to make simulation models both

easy to build and sufficiently expressive for realistic manufacturing scenarios. The main objective of this research is to enable analysts with limited programming experience to perform tactical-level manufacturing analyses effectively. To achieve this, we propose a novel hybrid activity- and resource-based simulation paradigm that bridges the gap between the intuitive process abstraction of BPM and the detailed operational view required in manufacturing. The research contributes both conceptually and practically:

- **Conceptually**, through the definition of a conceptual framework and a new meta-model specifically designed to represent manufacturing processes from an activity-centric perspective.
- **Practically**, through the development of ARMS (Activity-Resource Modeling Simulator), a simulation platform that supports modeling, execution, and analysis of business and manufacturing processes using this new paradigm.

Unlike traditional resource-based models that require granular configuration of every machine and material flow, ARMS adopts a hybrid approach that combines the activity-centric abstraction of BPM with the resource-awareness essential to manufacturing simulation. At its core, ARMS represents processes as sequences of activities, capturing what is done and how tasks relate in time and logic, while also explicitly modeling the resources that execute them. These resources include executors (such as machines, operators, or tools), products (items transformed during the process), and inventories (storage or buffering locations). This hybrid structure preserves the intuitive and high-level perspective of BPMN (Business Process Model and Notation) [65], making it accessible to analysts without programming expertise, while ensuring that critical manufacturing aspects like material flow, capacity constraints, and utilization can still be represented and analyzed. As a result, ARMS allows decision-makers to perform tactical analyzes focused on throughput, bottlenecks, and resource allocation without the overhead of constructing fully detailed low-level models.

The proposed activity-based paradigm also draws inspiration from the success of process modeling in BPM, where graphical representations such as BPMN are widely used to communicate process logic among stakeholders. However, standard BPM tools fail to account for the rich resource-related aspects of manufacturing, such as machine utilization, material flow, and setup dependencies. ARMS fills this gap by extending the activity-based perspective of BPMN to incorporate the essential dynamics of manufacturing systems, providing a balanced framework that is both accessible to non-programmers and expressive enough for meaningful simulation analysis.

The approach has been developed and validated through real-world use cases provided by our industrial partner, including production processes from the eyeglass and gear manufacturing sectors. These cases helped shape and evaluate the proposed meta-model and platform, ensuring their applicability to realistic industrial contexts. Experimental results demonstrate that ARMS can perform tactical analyses, such as evaluating throughput, identifying bottlenecks, and

testing alternative processes, at a level of accuracy comparable to that of a well-established resource-centric simulator, while requiring significantly less modeling effort.

The remainder of this thesis is organized as follows.

- Chapter 2 outlines the main challenges and motivations that drive this research, highlighting the gaps in current methodologies.
- Chapter 3 presents the theoretical foundations and related work necessary to contextualize the study, including key concepts and prior approaches in the field.
- Chapter 4 details the research methodology, data sources, and techniques employed to address the research objectives.
- In Chapter 5, the proposed meta-model is formulated and analyzed, with emphasis on its assumptions, parameters, and computational design.
- Chapter 6 describes the simulation platform implementation, deep diving into each component of the system (modeler, simulation engine, and dashboard).
- Chapter 7 discusses the evaluation conducted, the results, and performance metrics, providing insights into the effectiveness and limitations of the proposed approach.
- Chapter 8 summarizes the main findings, draws conclusions, and proposes directions for future work.

In summary, this thesis introduces a new hybrid activity- and resource-based modeling paradigm for manufacturing simulation, supported by a practical tool that empowers non-programming analysts to perform tactical analyses efficiently. By bridging the gap between intuitive process representation and manufacturing system complexity, this work contributes to making simulation a more accessible, flexible, and effective decision-support tool for modern manufacturing.

Chapter 2

Motivation

This research originated within an initiative carried out in collaboration with a manufacturing consulting firm. The firm's overarching objective was to empower its consultants, professionals with limited or no programming expertise, to construct and analyze models of real-world manufacturing processes through simulation. The ultimate goal was to facilitate tactical-level decision-making by enabling quick and accessible simulation-based analyses of production scenarios.

2.1 Tactical Analysis and Target Users

Tactical analysis occupies a middle ground between strategic planning and operational control. Whereas strategic decisions define the long-term direction of a company, such as market positioning, investments, or technological shifts, and operational decisions concern the day-to-day management of production lines, scheduling, or maintenance, tactical decisions deal with how to most effectively deploy resources and organize processes within existing strategic boundaries. Tactical questions are typically framed in terms of "how should we adapt?" rather than "what should we pursue?" or "how should we execute?" [68]. In industrial contexts, answering these questions requires a tactical simulation: a specialized modeling approach used to evaluate mid-term scenarios where performance, cost, and flexibility must be balanced.

In the context of manufacturing, the need for tactical simulation arises when evaluating structural or logic changes before implementation. Typical questions include:

- Should we hire additional staff or reskill existing workers?
- Would it be more efficient to invest in multiple specialized machines or a single flexible one?
- How should we modify the process flows to reduce lead times or environmental impact?

- What would be the cost implications of implementing these changes?

Such questions are central to sustaining competitiveness in dynamic industrial environments. They require the ability to test hypotheses, explore alternatives, and estimate outcomes, ideally before committing to real-world interventions. Tactical simulation provides a natural support for this type of reasoning: it allows decision-makers to experiment safely with what-if scenarios, assess Key Performance Indicators (KPIs), and visualize how different resource allocations or process structures affect overall performance.

In most organizations, however, the ability to perform simulation-based tactical analysis is not evenly distributed. Manufacturing consultants and business analysts, the professionals who best understand the operational and organizational realities of production, are often not the same individuals who build the simulation models. The conventional workflow requires consultants to articulate ideas and requirements that are then translated by simulation engineers or programmers into executable models. This dependency introduces communication overhead, slows iteration, and restricts the exploratory nature of tactical decision-making.

Our research is motivated by the need to democratize tactical simulation for two complementary user profiles:

- Manufacturing consultants, who possess in-depth domain expertise but limited programming experience, and
- Business and process analysts, who focus on performance evaluation, bottleneck identification, and data-driven improvement initiatives.

Both groups require a paradigm where tactical simulation is no longer a coding task, but a modeling task. They need an environment that combines the conceptual expressiveness of process modeling with the analytical rigor of simulation, so that they can directly link their understanding of processes to quantitative analysis. Meeting this requirement demands more than hiding programming details behind user interfaces; it calls for a methodological integration between activity-centric process modeling and resource-aware simulation semantics. This integration is the foundation of our approach and the central motivation for the work presented in this thesis.

2.2 Identifying the Gap

The initial phase of this work involved a thorough review of the state of the art to assess the suitability of existing modeling and simulation tools for this purpose. Our findings quickly revealed a significant gap between the capabilities of available tools and the practical needs of manufacturing consultants.

Most tools currently used in the manufacturing domain [53], both academic prototypes and commercial products such as FlexSim®, AnyLogic®, Arena® [27], and Warteschlangensimulator [42], are built upon a resource-centric paradigm.

These tools employ Discrete Event Simulation (DES) engines [34, 30], where the production floor is modeled as a network of queues and events, each executor represented by an event queue and connected through routes carrying parts or products.

While these tools are technically sophisticated and well-suited for detailed operational analyses, their modeling approach is deeply tied to low-level implementation constructs. Although they offer graphical interfaces, users inevitably need to rely on custom code, written in general-purpose or domain-specific languages, to capture realistic process behaviors. What is often presented as an "advanced" option quickly becomes a necessity for any non-trivial case. Consequently, such tools impose a steep learning curve that effectively excludes non-programmers, undermining their accessibility and usefulness for tactical analysis [17].

An additional challenge lies in the level of abstraction these tools operate at. They are inherently designed for detailed, bottom-up modeling, where complexity is expected and precision prioritized. However, tactical decision-making often requires a higher-level abstraction: analysts need to explore what-if scenarios, not simulate every operational detail. Even when simplified, resource-centric tools maintain their intrinsic complexity, offering little support for the abstraction needed at this level. This mismatch between modeling expressiveness and user accessibility renders these tools unsuitable for tactical decision-makers.

2.3 Towards an Activity-Centric Approach

Conversely, activity-centric notations like BPMN [65] are more adept at describing complex processes without the need to resort to coding, due to their ability to capture the subtleties of very complex processes, as shown by the variety of control workflow patterns it supports [75]. BPMN is also routinely adopted to create process models designed for simulation analysis, making it an ideal candidate to pursue our objective.

However, as its name suggests, BPMN is primarily targeted at generic business processes, and its application in manufacturing clashes with the lack of relevant concepts that characterize this domain, mainly because of its limited ability to model resources and how resources and activities relate to each other. We could claim that BPMN is well versed in addressing the control-flow perspective [46] while it lacks relevant abstractions to address a rich resource perspective. More precisely, BPMN's meta-model contains a *Resource* class, but its specification is deliberately vague ("[...] can be Human Resources as well as any other resource assigned to Activities during Process execution time" [65]). This is a well-known limitation, and in fact various scientific works propose different mechanisms to address it.

Several studies highlight the challenges of modeling manufacturing processes with BPMN, particularly in representing resources and performers for tactical what-if analysis. For instance, [36] demonstrates the need to enrich BPMN models with additional resource information to enable tactical scenario evaluation.

Similarly, [28] explores the modeling of manufacturing operations using a catalog of BPMN fragments, revealing recurring difficulties in integrating resources into process models. While the HORSE Project [28] addresses resource management in manufacturing, it does not support simulation or tactical analysis. Further emphasizing this gap, [8] underscores the need for improved resource modeling in business processes. After analyzing the limitations of BPMN and BPSIM, the authors adopt DPMN to model processes with resources. This is a step in the right direction, though their approach still lacks the comprehensive features required for manufacturing-specific scenarios.

To better accommodate manufacturing requirements, some researchers propose extending BPMN. For example, [92] introduces manufacturing-specific elements as BPMN extensions, although their approach presents limited coverage for diverse industrial scenarios and lacks the clear operational semantics needed for simulation analysis, a limitation shared with other similar approaches presented in the past. Another limitation in standard BPMN is the modeling of batching activities, which has been addressed through specialized extensions [71].

Beyond modeling, tactical analysis plays a critical role in manufacturing and business processes. [67] presents a framework for defining and linking KPIs to business processes, facilitating performance evaluation. While simulation-based tactical decision-making is effective, existing approaches often rely on resource-centric tools rather than BPMN. For example, [18] models and simulates a container terminal process using Witness, bypassing BPMN's limitations in resource simulation.

2.4 Research Contributions

The findings from both the literature and our collaboration with domain experts clearly indicate the absence of a comprehensive framework (discussed in Chapter 5) that integrates the strengths of BPMN's activity-centric representation with the resource-awareness and executability required for manufacturing simulation. This realization motivated our research to develop an hybrid activity- and resource-based meta-model, modeling environment, and simulation engine capable of representing manufacturing processes in a way that remains both accessible to non-programmers and semantically robust for simulation-based tactical analysis.

In light of these considerations, this thesis pursues a methodological integration of activity-centric modeling and resource-aware simulation tailored to tactical decision-making in manufacturing. Concretely, it aims to:

- Define an hybrid activity- and resource-based meta-model that preserves BPMN's control-flow strengths while introducing explicit, compositional abstractions for manufacturing resources, their capabilities, capacities, and allocation policies.
- Specify clear operational semantics that link process models to executable,

resource-sensitive simulation behavior, enabling reproducible what-if analyses without ad-hoc coding.

- Design a modeling environment for non-programmers that allows manufacturing consultants and business or process analysts to create, parameterize, and validate simulation-ready models using familiar activity-centric constructs.
- Develop a simulation engine that executes instances of the proposed metamodel efficiently, computes relevant KPIs for tactical analysis (e.g., throughput, lead time, utilization, cost, and sustainability proxies), and supports scenario comparison.
- Demonstrate applicability and value through case studies co-designed with domain experts, showing that the approach shortens iteration cycles, reduces dependence on programming expertise, and yields credible decision support for tactical decision-making.

Together, these objectives structure the contributions of the thesis and provide a roadmap for the chapters that follow: from foundations and metamodeling, through semantics and tooling, to empirical validation in representative manufacturing scenarios.

Chapter 3

Background

This chapter lays the conceptual and methodological groundwork for this thesis. Its focus is the use of computer-based simulation to support tactical decisions in manufacturing. These are decisions that operate on a short- to medium-term horizon and translate strategic intent into executable plans. We begin by characterizing manufacturing and business processes as the operational contexts in which such decisions are made, clarifying common flow and performance notions (throughput, work-in-process, cycle time, flexibility) and the different emphases of activity- versus resource-oriented views. We then introduce simulation as an analytical instrument for exploring complex, stochastic, and time-dependent behavior, with particular attention to what-if analysis and the treatment of uncertainty. Building on this, we review simulation paradigms and software capabilities relevant to the thesis, highlighting requirements for credible modeling (expressiveness, statistical rigor, visualization, and usability). Finally, we survey process-modeling formalisms and tools, from BPMN/BPSim and Petri nets to resource-centric discrete-event environments, and discuss how integrating activity- and resource-based perspectives enables faithful, decision-oriented analyses. Together, these elements define the background against which the thesis methodology and results are developed.

3.1 Manufacturing and Business Processes

Manufacturing and business processes form the operational context for tactical analysis. Both describe structured sequences of activities that transform inputs into valuable outputs, yet they differ in granularity and emphasis.

3.1.1 Manufacturing processes

A manufacturing process transforms materials into useful, portable artifacts and does so by orchestrating information, people, and technology [23]. In practice, it unfolds through coordinated operations among machines, workers, and material-

handling systems arranged in a physical space; the whole arrangement constitutes the *manufacturing system*.

Quantitative analysis of manufacturing systems revolves around a small set of interrelated concepts that connect design choices to operational outcomes.

Throughput (production rate). Throughput is the rate at which finished parts exit the system. It is governed by resource capacities and bottlenecks, and is sensitive to scheduling, routing, and maintenance decisions. In flexible cells and FMSs, even small changes in dispatching or rerouting policies can shift the effective bottleneck and alter realized throughput [64, 20, 60].

Inventories and buffers. Queues and work-in-process (WIP) buffers decouple upstream and downstream operations and absorb variability in processing, setups, arrivals, and failures. Their placement and sizing shape flow resilience and stability, but also tie up capital and space. Studies of automated lines and FMSs routinely show that buffer allocation interacts with stochastic failures and setup patterns, affecting both stability and cost [64, 20].

Cycle time and responsiveness. Cycle time measures how long a part spends from entry to exit, comprising processing, waiting, transport, and setup. It proxies responsiveness and, together with throughput and WIP, reflects classical flow trade-offs. Empirical simulation studies of FMS cells highlight how targeted changes. For example, cutting-tool strategies or resource pacing can reduce cycle time while rebalancing utilization [60, 20].

Flexibility. Flexibility captures the capability to produce multiple variants and to route work over alternative process paths. Recent reviews emphasise that flexibility is multi-faceted and must be tied to specific design and control levers [84]. Building on the input–throughput–output (I–T–O) view, one distinguishes *input flexibility* (e.g., suppliers or materials), *process flexibility* (alternative machines/routes), and *output flexibility* (product variety), and relates these to enabling technologies on the shop floor [45]. In practice, routing and mix flexibility improve responsiveness and robustness to demand shifts but may complicate scheduling and local efficiency [64].

Modeling challenges and trade-offs

Realistic models must reconcile fidelity with parsimony while addressing recurring difficulties:

- **Structural complexity** Modern facilities comprise interacting machines, robots, conveyors/AGVs, shared resources, alternative routings, and non-trivial control logic. Layout, routing, and resource contention are first-order determinants of Flexible Manufacturing Systems (FMS) performance [20, 84].

- **Information demands** Credible models require detailed data on processing and setup times, failure/repair, changeovers, operator behaviour, queues, and lead times; gaps often force simplifying assumptions [64].
- **Variability and randomness** Breakdowns, quality rework, stochastic arrivals, human variability, and transport delays must be represented to obtain meaningful inferences [64, 20].
- **Operational constraints and decision rules** Eligibility restrictions, sequence-dependent setups, heterogeneous operator speeds, assignment and dispatching rules, and maintenance policies define the feasible operating envelope; evaluating flexibility requires these to be explicit [60, 84].

These challenges surface classical trade-offs among throughput, WIP, lead time, and cost. For example, larger buffers can shorten cycle time yet increase inventory cost; added routing flexibility can improve responsiveness but raise coordination overhead or reduce local efficiency. Case studies of FMS cells show that identifying the true bottleneck (e.g., a CNC operation) and changing cutting-tool strategies can lift throughput and reduce cycle time, while rebalancing utilization across robots and conveyors [60].

Simulation, particularly Discrete-Event Simulation (DES) which is discussed later in this chapter, is well suited to explore such trade-offs, experiment with alternative control rules, and assess flexibility options before implementation [64, 20]. In many industrial studies, DES is combined with structured modelling (e.g., IDEF0) and reliability formalisms (e.g., fault-tree analysis) to link process design, risk, and performance [60].

3.1.2 Business processes

A business process is a coordinated set of activities performed within an organizational and technical context that jointly realize a business goal [85]. The perspective foregrounds end-to-end value creation across functional boundaries.

Abstraction levels and representations

Business processes are described at multiple, complementary levels [85]:

- **Organizational level:** goals, roles, responsibilities, governance, and major subprocesses.
- **Operational level:** the ordering and dependencies of activities (often modeled in BPMN, EPCs, or Petri nets).
- **Implemented level:** executable workflows and information systems enabling enactment, monitoring, and control.

Across this spectrum, processes range from transactional, automatable flows (e.g., order-to-cash) to knowledge-intensive, weakly structured work that relies on human judgment.

Performance, variability, and data

Analytical treatment mirrors manufacturing in focusing on flow and resource use: throughput (cases per period), lead time (case start-to-end), waiting times, resource utilization, and cost per case. Real-world processes exhibit variability in arrivals, task durations, branching, rework, and exceptions; business process simulation (BPS) explicitly models these stochastic features [74]. High-quality event data (timestamps, resources, case attributes) enable process mining for discovery, conformance, and performance analysis, and provide inputs and validation signals for simulation [1].

Business Process Management and simulation

Business Process Management (BPM) integrates methods for design, configuration, execution, monitoring, and analysis. Explicit process models underpin simulation and quantitative evaluation, linking managerial and workflow perspectives [1]. Recent surveys emphasize BPS as a tool to analyse performance under varying conditions and to support decision-making and improvement programs [74]. Emerging work connects simulation to quality and governance frameworks, using model-driven metrics to steer improvement [21]. Flexibility and agility remain central: organisations adapt via variant management, exception handling, ad-hoc changes, and virtualised work arrangements [1].

3.2 Simulation: a tool for analysis and decision-making

Simulation is a powerful methodology for studying systems whose complexity, stochasticity, and dynamics make analytical solutions impractical. In essence, it involves building a computational model that represents the real system and then conducting experiments on this model to understand its behavior, test alternative configurations, and evaluate performance indicators. By capturing interdependencies and variability, simulation allows decision-makers to explore possible futures safely and systematically.

A first conceptual distinction lies between *physical* simulation, which uses tangible prototypes or scaled models, and *computer-based* simulation, where the system's behavior unfolds within a digital environment [57]. The present work focuses on computer-based simulation, as it enables rapid experimentation, reproducibility, and the testing of what-if scenarios without disturbing real operations.

3.2.1 The Role of What-If Analysis

What-if analysis is a core technique in simulation-based decision-making. It involves systematically varying model parameters or configurations to examine the consequences of tactical or strategic choices. Each simulation run represents

a plausible system state under a particular combination of conditions, such as resource levels, scheduling policies, or demand patterns. Comparing outputs across runs allows analysts to identify decisions that improve performance and to detect changes that may introduce new inefficiencies [35].

Consider a production setting. An engineer might evaluate the effects of adding a second machine at a bottleneck, modifying shift schedules, or reducing batch sizes; each variant constitutes a what-if experiment. Key performance indicators: throughput, utilization, waiting time, work-in-process, and cost are then analyzed to characterize trade-offs. Because simulation captures stochastic effects (e.g., random breakdowns, variable demand, operator delays), what-if analysis yields probabilistic insights rather than point predictions, revealing both average behaviour and the likelihood of extreme outcomes. This supports risk-informed tactical planning.

Modern simulation platforms increasingly automate what-if exploration via design-of-experiments interfaces and optimization modules, enabling efficient evaluation of hundreds of candidate configurations. In this way, simulation functions as a quantitative laboratory for decision-making: a safe environment in which complex interdependencies can be explored long before real resources are committed.

The principal advantage of simulation to perform what-if analysis is its flexibility: it enables experimentation without operational risk, exposes bottlenecks, and supports quantitative decision-making under uncertainty. These strengths, however, come with costs. Building a valid and credible model demands detailed data, domain expertise, and rigorous verification and validation [77]; moreover, results must be interpreted cautiously, as simplifications and assumptions can shape outcomes [56]. These considerations motivate structured modeling practices and the use of specialized simulation tools.

3.2.2 Types of Simulation

Simulation encompasses multiple paradigms, each suited to a specific kind of system representation [57]:

- **Discrete-Event Simulation (DES)** models systems as sequences of events that occur at discrete points in time. Each event changes the system state, such as an arrival, a machine failure, or a service completion. DES is widely used in manufacturing and logistics because it naturally represents queues, resources, and process flows.

- **Agent-Based Simulation (ABS)** focuses on autonomous entities (agents) that follow behavioral rules and interact with one another. It is well suited for modeling decentralized or human-driven systems, such as collaborative robots or operator decision-making in production.

- **System Dynamics (SD)** models systems through continuous feedback loops among stocks and flows, emphasizing accumulations and long-term

trends. It is often used for strategic policy studies, such as supply chain dynamics or inventory management.

Hybrid models that combine DES, ABS, and SD are increasingly used when different abstraction levels must be integrated [47]. For example, when macro-level demand policies (SD) interact with micro-level production events (DES).

3.2.3 Treatment of Uncertainty and Time

Simulation models can be classified as deterministic or stochastic. Deterministic models yield the same result for a given set of inputs, whereas stochastic models incorporate randomness, producing outputs that vary across replications. The latter are especially relevant for manufacturing systems, where variability and uncertainty are inherent.

Similarly, models may be static or dynamic. Static simulations, such as Monte Carlo models, capture a snapshot or steady-state of the system, while dynamic simulations represent its evolution over time. Tactical analyses in manufacturing almost always require dynamic stochastic simulations, as they must reflect temporal evolution, resource interactions, and process variability.

3.2.4 Simulation Software

Simulation software plays a central role in enabling the development, execution, and analysis of simulation models. Over the past decades, specialized simulation packages have evolved as an alternative to general-purpose programming languages, offering built-in functionalities that reduce programming effort and improve usability. At a broad level, simulation software can be classified as general-purpose, which is applicable to a wide range of domains, or application-oriented, which targets specific areas such as manufacturing, healthcare, or communication networks [24]. Regardless of the type, effective simulation software must combine modeling flexibility, statistical reliability, visualization, and ease of use.

From a methodological perspective, good simulation software should provide sufficient modeling flexibility to represent systems of varying complexity. This includes the ability to define system entities and attributes, capture decision logic, and model resources and queues. At the same time, ease of use and accessibility are essential to ensure that the tool can be effectively adopted not only by simulation experts but also by practitioners and stakeholders. Features such as graphical user interfaces, hierarchical modeling constructs, and debugging aids improve both learning and productivity [31].

A second important characteristic is the integration of visualization and animation. Built-in dynamic graphics help users communicate model logic, validate system behavior, and engage non-technical stakeholders. High-quality animation also supports training and decision-making by presenting the system's evolution over time in an intuitive way [83].

Third, robust statistical and analytical capabilities are necessary to ensure that simulation results are valid and reliable. This includes access to well-designed random number generators, a broad library of probability distributions, methods for replications and confidence interval estimation, and the ability to design experiments. Increasingly, modern software also integrates optimization modules that assist in exploring alternative system designs [4].

Finally, practical aspects such as platform compatibility, computational efficiency, user support, and documentation contribute to the overall effectiveness of a simulation package. Support services, example libraries, and accessible documentation lower the learning curve and increase long-term usability. Comprehensive reporting and graphical output options ensure that model results can be communicated clearly and customized to suit managerial or research needs [16].

In summary, a good simulation software tool should balance flexibility, usability, visualization, statistical rigor, and support infrastructure. These characteristics enable it to serve as a credible and effective decision-support environment across application domains.

When it comes to software products, numerous simulation packages have been developed, each with its own strengths and domain of application. Widely used general-purpose tools include Arena, FlexSim, and Simio, which provide extensive libraries, graphical modeling environments, and strong visualization capabilities [24]. Application-oriented tools are also common, designed for specific domains such as manufacturing systems, healthcare operations, or logistics networks.

In addition to commercial packages, open-source and domain-specific tools are gaining attention. Omnet++, SimPy, DESMO-J are just a few of the possibilities when choosing a simulation software [22]. For example, DESMO-J has been used to build a business process simulator [49], and SimPy to model and simulate for decision making for the proper design of assembly lines, as well as identifying possible unbalances in production lines and overloaded processes [37]. Another example is Kalasim, a discrete-event simulation framework developed in Kotlin that emphasizes flexibility and integration with modern programming practices. Unlike purely graphical tools, it leverages the expressiveness of a programming language to create simulation models while still offering the abstractions needed for efficient modeling. This makes it particularly suitable for research applications or projects where custom logic is required.

3.3 Tactical Manufacturing Analysis: Concepts, Models, and Simulation

Tactical analysis focuses on medium-term decisions that shape the efficiency and responsiveness of manufacturing systems. Decisions concerning capacity allocation, resource planning, or process configuration. Simulation serves as an experimental platform for such analysis, allowing engineers to test potential interventions virtually before implementing them physically.

3.3.1 Between Strategic and Operational Analysis

Foundational management theory formalized the differences between strategic, tactical (often termed management control), and operational decisions. Robert N. Anthony (1965) [5] proposed a now-classic framework distinguishing strategic planning, management control, and operational control as three different system levels (presented as a pyramid in Figure 3.1). Strategic planning decides long-term objectives, policies, and resource commitments (e.g. entering new markets or changing business models). Tactical or management control translates strategy into mid-term plans and budgets, guiding departments on how to implement strategy within set constraints. Operational control oversees short-term execution of tasks, ensuring daily activities meet the tactical plans efficiently. This hierarchy is widely cited as it helps clarify how top-level strategy gets implemented through intermediate plans and day-to-day operations.

Figure 3.1: Pyramid of differences between strategic, tactical, and operational analysis [52]

This tripartite framework is not only theoretically robust but also widely applied across disciplines. For instance, in global logistics and supply chain management, Schmidt and Wilhelm (2000) offer a detailed analysis of how the three levels function within multi-national logistics networks [78]. At the strategic level, decisions focus on the structural design of the supply chain, such as selecting facility locations, determining plant capacities, and investing in production technologies over a multi-year horizon. Tactical decisions operate over mid-range horizons (6–24 months), prescribing policies on material flows, inventory management, and production planning within the network's existing constraints. Finally, operational decisions address daily scheduling and resource deployment to ensure timely deliveries and customer satisfaction. Their work emphasizes that these levels are not isolated; rather, they are interdependent, with tactical and operational feasibility constrained by strategic design, and operational performance feeding back to inform higher-level adjustments.

A contrasting but complementary example is provided by Koliba et al. (2022), who developed a simulation-based platform to analyze human decision-making in livestock biosecurity systems [52]. In this public health and agricultural systems context, strategic decisions are made at the institutional level, setting long-term disease control policies and regulatory frameworks. Tactical planning occurs at the facility level, where managers allocate resources, establish sanitation protocols, and design staff training programs. Operational decisions are enacted by front-line workers through daily biosecurity practices such as disinfection, barrier control, and reporting. The simulation explicitly connects micro-level behaviors to macro-level outcomes, demonstrating how operational compliance can significantly influence the effectiveness of tactical protocols and strategic goals. This example underscores the importance of analyzing decision levels not only in isolation but also in their dynamic interaction across organizational layers.

3.3.2 Analytical Foundations and Simulation for Tactical Decision Support

Tactical manufacturing analysis relies on models that reveal how system structure and variability determine performance. Analytical models offer theoretical clarity and intuition, while simulation extends those principles to complex, realistic contexts. Together, they provide the intellectual and computational basis for making medium-term manufacturing decisions about capacity, resource allocation, and process improvement.

Why Analytical Models Matter for Tactical Analysis

Analytical methods represent the essential relationships among utilization, variability, and flow. They provide simple yet powerful frameworks for predicting performance, benchmarking alternatives, and identifying leverage points for improvement. Although these models assume simplified or steady-state conditions, they remain indispensable for building intuition and defining theoretical limits before resorting to simulation.

Queueing Theory and Factory Physics: flow, variability, and limits

A fundamental analytical pillar is queueing theory [38], which formalizes how variability and resource utilization influence waiting times, work-in-process, and throughput. Even small increases in variability or utilization can cause disproportionate growth in delay, revealing the nonlinear nature of congestion in production systems. Building upon such stochastic insights, the Factory Physics framework [44] synthesizes empirical and theoretical results into a set of production "laws". It defines parameters such as the bottleneck rate and the critical WIP, clarifying that every manufacturing system operates within physical limits beyond which additional inventory ceases to improve throughput. These analytical laws enable tactical analysts to recognize how variability, capacity,

and inventory interact, and where improvement efforts are most likely to yield results.

Theory of Constraints: from insight to managerial action

Complementary to these descriptive frameworks, the Theory of Constraints (TOC) [33] offers a prescriptive methodology for continuous improvement. Its five focusing steps: identify, exploit, subordinate, elevate, and repeat provide a practical cycle for managing the system's constraint. TOC translates analytical understanding into actionable guidance: only interventions at the current bottleneck can enhance overall performance. From a tactical perspective, TOC aligns naturally with simulation, which allows quantitative testing of proposed constraint-relief measures before implementation.

Discrete-Event Simulation in Tactical Practice

While the above analytical foundations are the ground on which every other advancement relies on, discrete event simulation has become a way-to-go tool for tactical manufacturing analysis, enabling decision-makers to evaluate complex scenarios that defy closed-form analysis. DES models imitate the operation of a real production system by representing events (e.g. machine start/stop, job arrival, shift change, breakdown) on a time-stepped basis [50]. This allows a near-realistic reenactment of factory behavior under specified conditions. Crucially, DES can incorporate stochastic variability (random processing times, machine failures, fluctuating arrivals) and interdependencies (e.g. shared resources, batching) that are very difficult to handle analytically. By experimenting on a virtual model, engineers can quantitatively assess the impact of tactical decisions before implementing them on the shop floor. For example, simulation can reveal how adding one extra forklift or one additional hour of overtime affects overall throughput and where new bottlenecks might emerge, all without disrupting actual production.

One of the primary tactical uses of DES is throughput optimization and bottleneck analysis. In a simulation model of a manufacturing line, it is possible to pinpoint the station or resource that is impeding flow by collecting statistics like machine utilization, queue lengths, blockage/starvation time, or waiting time at each stage [81]. Unlike static calculations, a DES can capture shifting bottlenecks, situations where the primary constraint moves between stations over time due to variability or product mix changes. Simulation-based methods have been shown to quickly and accurately identify both persistent and transient bottlenecks in complex systems. For instance, researchers at Toyota Central R&D Labs were among the first to use DES software to analyze production lines, developing automated bottleneck detection via simulation models (the GAROPS simulator) in a discrete manufacturing setting [81]. Following this, companies like General Motors applied similar simulation techniques (e.g. the C-MORE simulation system) to diagnose bottlenecks in their manufacturing networks and guide improvements. Modern DES tools even provide built-in analytics to

highlight the bottleneck resource, using measures such as longest cumulative blocking time or highest utilization as identifiers. By iterating scenarios in a DES (e.g. reallocate tasks, add buffer capacity, adjust cycle times), engineers can test various strategies to relieve the bottleneck and quantify the throughput gains from each option in a risk-free environment.

Another critical tactical application of DES is in resource allocation and capacity planning. Tactical decisions often involve questions like: "How many machines or workers are needed to meet a target output?", "Is it better to invest in an additional line, or run overtime on the existing line?", "What is the optimal product mix given current capacity constraints?". DES supports these decisions by allowing the creation of what-if scenarios. The simulation model can incorporate proposed changes (additional equipment, different staffing levels, altered shift patterns, etc.) and then compute performance outcomes such as throughput, cycle time, resource utilization, and work-in-process levels. Because DES captures interactions and randomness, it provides more reliable forecasts of system performance under each scenario than deterministic calculations. This proves invaluable for tactical capacity planning. For example, Jurczyk-Bunkowska et al. (2021) describe a tactical capacity planning approach that combines DES with throughput accounting to evaluate mid-term expansion options [50]. In their case study, a medium-sized manufacturer used a DES model to analyze the current production state and simulate several capacity expansion scenarios (such as purchasing a new machine to debottleneck a process). The simulation outputs (throughput, cycle time, etc.) were then fed into a financial model (throughput accounting) to assess the profitability of each scenario. This integration demonstrates how DES serves as a decision support tool at the tactical level, by quantifying the operational impact of changes, it informs management whether an investment or reallocation will likely pay off.

Indeed, discrete-event simulation has been applied to a wide range of tactical manufacturing problems. Some notable examples presented in [50] include:

- Analyzing the configuration of flexible manufacturing systems, such as determining the best layout or module combination for a production line.
- Matching capacity to uncertain demand in sectors like healthcare or make-to-order production, by simulating different staffing or machine capacity strategies under variable demand scenarios.
- Studying labor and workflow policies in highly variable manual production processes (e.g. how worker assignments or skill levels affect throughput in a low-automation environment).
- Serving as a decision-making tool for complex assembly lines, for instance simulating various configurations of an automotive assembly process to identify the setup that maximizes throughput without adding excessive WIP.

These use cases show that DES is extremely versatile for medium-term production analysis. It is commonly employed to test improvements such as adding

buffers, changing batch sizes, rearranging layout, adopting new scheduling rules, or introducing quality inspection points, all of which can affect throughput and need careful evaluation. Modern commercial simulation software (e.g. Siemens Tecnomatix Plant Simulation, Arena, FlexSim, AnyLogic) provide templates and libraries specifically for modeling manufacturing systems [81]. The widespread availability of such tools, along with extensive user communities and training resources, has lowered the barrier to simulation modeling for industrial engineers. As a result, even small and medium-sized enterprises are increasingly able to leverage DES for tactical decision support, but it usually requires in-house advanced programming skills to model the more complex scenarios. In summary, DES plays a crucial role at the tactical level by bridging the gap between theoretical analysis and real-world complexity, it provides a safe sandbox to experiment with changes and optimize manufacturing performance at the cost of knowing how to design and implement the scenarios.

Emerging Trends and Advanced Methodologies

Recent developments extend the power and scope of simulation for tactical decision-making:

- Data-Driven Simulation and Digital Twins: One emerging trend is the integration of real-time data into simulation models, leading to the concept of the digital twin. A digital twin is a live, digital replica of the physical production system that can be used for both monitoring and decision support. In manufacturing, simulation-based digital twins allow online adjustments and what-if analyses using up-to-date system data. Rather than relying solely on off-line models, companies are now linking simulations with factory floor data (sensors, IoT devices, production logs) to dynamically detect issues and optimize operations [55, 82]. For example, data-driven methods can perform dynamic bottleneck identification by analyzing buffer levels, machine states, and throughput in real time [81]. If a workcenter's queue starts consistently growing or its utilization spikes, the digital twin can flag it as a emerging bottleneck and evaluate possible interventions. This real-time aspect goes beyond traditional DES, enabling a more proactive tactical response. Research has shown that such data-enhanced approaches can promptly identify transient bottlenecks and help adjust production plans on the fly. The advent of Industry 4.0 technologies is strongly driving this trend, as more factories instrument their equipment and seek to leverage big data analytics alongside simulation models.

- Simulation-Based Optimization: Another important methodology is combining DES with optimization techniques to search for the best system configuration or policy. This is often termed simulation optimization or simheuristics. Instead of manually experimenting with scenarios, analysts use algorithms (e.g. genetic algorithms, simulated annealing, or gradient-based methods) to guide the simulation model toward optimal or near-optimal solutions on objectives like maximizing throughput or

minimizing lead time. For instance, an optimization engine can iteratively adjust resource allocation (number of machines, workers, buffer sizes) in the simulation, evaluating performance, until it converges on an improved design. Many commercial DES platforms now include optimization modules or interfaces to connect with solvers and even machine learning for decision-making [50]. However, it's noted that small and mid-sized firms sometimes find pure optimization or AI approaches daunting due to required expertise and data, which is why user-friendly simulation-with-optimization tools are an active research and development area. The goal is to harness artificial intelligence to more efficiently explore the huge decision space in complex manufacturing systems.

- Parallel and Cloud-Based Simulation: To support the above, there is a push towards faster simulation via parallel processing and cloud computing. Large-scale manufacturing simulations (with many machines, products, and detailed processes) can be computationally intensive. Recent research has explored parallel discrete event simulation (PDES) to distribute the simulation across multiple processors or cores, allowing analyses that were previously too slow or infeasible [43]. Cloud-based simulation services also enable running many what-if scenarios in parallel, which is particularly useful for simulation optimization or for building statistical confidence in results. This technological trend makes it possible to evaluate tactical decisions with high fidelity (e.g. including minute-by-minute variability or complex job-shop routings) without prohibitive run times.

- Hybrid Simulation and Other Paradigms [11]: Modern manufacturing challenges sometimes benefit from hybrid modeling that combines DES with other simulation paradigms like agent-based modeling or system dynamics. For example, human operators or AGVs (automated guided vehicles) might be modeled as autonomous agents with certain behaviors interacting within a discrete-event logistics system. System dynamics, which deals with aggregate flows, might be coupled to DES for higher-level decisions like inventory policies. These hybrid approaches are an active research area to capture both high-level and detailed-level dynamics in tactical planning. They remain less common in practice than pure DES, but are seen in academic research on complex systems (such as supply chain simulations that include both continuous information flows and discrete events).

- Lean and Six Sigma Integration: Another current methodology is the integration of DES into lean manufacturing and Six Sigma initiatives. Simulation is used to validate and quantify improvements proposed by lean techniques (like line balancing or kanban system design) before physical changes are made. In Six Sigma's DMAIC process (Define-Measure-Analyze-Improve-Control), DES can play a role in the Analyze and Improve phases by modeling process variations and testing the impact of changes on defect rates and cycle times [87]. This reflects a more general trend

of simulation being embedded in the decision workflows of manufacturing management rather than being a standalone analysis, it complements value stream mapping, empirical observation, and other tactical analysis tools.

Summary

Queuing theory, Factory Physics, and TOC establish the analytical laws of flow and constraint; discrete-event simulation extends these principles into realistic, data-rich environments. Together they form the methodological backbone of tactical manufacturing analysis, analytical models guide understanding, while simulation enables experimentation and validation under complex, time-dependent conditions.

3.4 Foundations of Process Modeling: Activity-Centric and Resource-Centric Perspectives

Process modeling can be approached from two complementary perspectives: *activity-based* and *resource-based*. The distinction lies in the primary modeling focus: activities and their control flow versus resources and their interactions.

3.4.1 Activity-based Modeling and Simulation Tools

Activity-based modeling views a process as a structured sequence of tasks, decisions, and events. The focus is on the logical and temporal relationships among activities rather than on the underlying resources. This approach is particularly suitable for representing workflows and business processes where coordination and information flow are central.

Petri nets

Among formal representations, Petri nets provide a rigorous foundation [69]. And, as stated by Weske, "Petri nets are one of the best known techniques for specifying business processes in a formal and abstract way" [85].

A Petri net is defined as a directed bipartite graph consisting of two types of nodes: *places* (often drawn as circles) and *transitions* (drawn as rectangles or bars). Directed arcs connect places to transitions (and vice versa), and the state of the system is represented by *tokens* residing in places.

The dynamic behavior is governed by the firing rule of transitions: a transition is enabled and may fire when each of its input places contains at least the required number of tokens; upon firing, the transition consumes tokens from its input places and produces tokens in its output places, atomically changing the marking. This token-based execution semantics naturally represents parallelism and synchronization, multiple transitions can be enabled simultaneously, and the firing of independent transitions can occur in any order, reflecting true concurrency. Thanks to these properties, Petri nets are well-suited for modeling

the concurrent behavior of distributed processes, allowing one to capture complex workflows with choice, iteration, and simultaneous activities. Moreover, Petri nets have a rigorous mathematical foundation that enables formal analysis of process properties. Analysts can leverage Petri net theory to check for conditions such as reachability of certain states, the absence of deadlocks (liveness), or boundedness of resources in a modeled system. Unlike informal diagrammatic notations, Petri nets come with exact execution semantics and a well-developed theory for verification and performance analysis [85]. This makes them a powerful tool for modeling and analyzing concurrent systems, from communication protocols to business workflows, especially when one needs to ensure correctness or study emergent behaviors in complex processes.

Business Process Model and Notation (BPMN)

Building on these formal roots, the Business Process Model and Notation (BPMN) [65] standard offers an accessible graphical language for representing activities, events, and decision gateways. It is a widely adopted standard for graphical representation of business processes [85] thanks to its emphasizing of human interpretability and stakeholder communication, bridging process design and execution. In particular, BPMN exemplifies an activity-centric modeling approach: it focuses on the sequencing of activities and the flow of control between them, providing a diagrammatic way for both business stakeholders and IT specialists to visualize how work is carried out.

A BPMN diagram depicts the process logic in terms of various standardized elements that are intuitive yet expressive. Key BPMN flow objects include:

- **Events**: These represent triggers or results in the process. Every process begins with a Start event and ends with at least one End event. BPMN also defines Intermediate events (e.g., a Timer event or Message event) that can occur during the process flow, modeling things like delays or incoming/outgoing messages.
- **Activities**: These are the units of work in the process. The most common activity is a *Task*, representing a single work step (e.g., "Review Application" or "Ship Order"). BPMN also allows grouping tasks into a *Sub-Process* for hierarchical modeling, encapsulating a set of activities that can be detailed separately.
- **Gateways**: These elements capture the branching and merging of paths in the process. For example, an exclusive decision gateway (XOR) routes the flow into one of several paths based on a condition, whereas a parallel gateway (AND) splits the flow into multiple concurrent paths (or synchronizes them).

These core elements are connected by directed *sequence flows* to form a complete process model, showing the progression from start to end through various activities and decision points. In addition, BPMN supports artifacts and

constructs to enrich the model: for instance, *pools* and *lanes* can be used to denote different participants (organizations, departments, or roles), and data objects can be attached to activities to represent information being produced or consumed. By using this rich set of symbols and notational conventions, BPMN enables the creation of clear and detailed process diagrams that are understandable to stakeholders. The activity-centric nature of BPMN means the diagram explicitly highlights the tasks being performed and their order, abstracting away low-level resource details. This makes BPMN well-suited for high-level process documentation, analysis, and even automation. Indeed, BPMN has become a de facto standard in business process modeling due to its balance of simplicity and expressiveness, and its diagrams can serve as a blueprint for process implementation or improvement initiatives [85].

However, BPMN itself is descriptive; quantitative evaluation requires linking it with simulation engines.

BPSim: Standardized parameterization of BPMN models

The *Business Process Simulation (BPSim)* specification, developed by the Workflow Management Coalition (WfMC) [86], formalizes how simulation parameters can be attached to BPMN elements to describe dynamic, stochastic, and resource-dependent behavior. BPSim provides a structured way to enrich process models with information about timing, control flow, resource usage, and cost, transforming static process diagrams into executable specifications.

The specification defines several categories of simulation parameters:

- **Time parameters**: describing aspects such as task durations, inter-arrival intervals, or delays between events.
- **Control parameters**: defining routing probabilities at decision gateways or iteration frequencies.
- **Resource parameters**: capturing allocation rules, role capacities, calendars, and working schedules.
- **Cost parameters**: enabling the assignment of fixed or variable execution costs to activities or resources.
- **Property parameters**: which represent user-defined attributes or domain-specific extensions.

Each parameter can be defined as a constant, probability distribution, or functional expression, enabling stochastic experimentation. During simulation, these annotations drive the generation of execution traces that reflect realistic process variability. The separation of concerns between the *process definition layer* (BPMN model) and the *simulation configuration layer* (BPSim parameters) promotes interoperability across tools and allows analysts to reuse the same process model under different simulation scenarios. By systematically encoding process behavior, BPSim bridges descriptive process modeling and quantitative analysis, forming the foundation for later simulation engines.

In summary, BPSim represents the standardized foundation of BPMN-based simulation: it defines the vocabulary and structure that enable what-if experimentation and performance evaluation directly on BPMN models.

Scylla: Extensible execution of BPMN-based simulations

Building upon the BPSim framework, the Scylla simulation environment, developed by Pufahl et al. [72], implements a discrete-event execution engine that interprets BPMN models annotated with simulation parameters. Unlike static analysis tools, Scylla transforms BPMN constructs into executable simulation entities, enabling analysts to observe process behavior over simulated time.

Scylla's architecture is based on the DESMO-J discrete-event simulation kernel, which schedules events such as activity start and completion, message exchanges, or resource acquisitions. A key strength of Scylla is its *extensible plug-in architecture*, which allows new BPMN constructs, behavioral rules, or decision mechanisms to be added without modifying the core engine. This design supports advanced modeling features such as:

- **Decision-aware tasks**, linked to Decision Model and Notation (DMN) rules to dynamically alter process routing.
- **Batch or parallel processing**, where multiple process instances are executed jointly or concurrently.
- **Shared resources and inter-process interactions**, allowing the concurrent simulation of multiple workflows competing for capacity.

These capabilities make Scylla particularly effective for conducting what-if analyses within business and service systems. Users can modify simulation parameters, such as task durations, resource pool sizes, or event probabilities to test alternative operational strategies and assess their impact on key performance indicators like cycle time, throughput, and utilization. This is possible by using different models as input to the simulator as shown in figure 3.2. In doing so, Scylla operationalizes the principles established by BPSim, turning annotated BPMN models into dynamic, executable simulations.

In summary, Scylla embodies the transition from descriptive modeling to active experimentation. It provides the computational engine that executes BPSim-parameterized BPMN models, enabling detailed, event-level process analysis within a flexible and extensible framework.

PyBPMN: Bridging performance and reliability in BPMN models

While BPSim and Scylla focus primarily on temporal and resource allocation aspects, the Performability-enabled BPMN (PyBPMN) framework by Bocciarelli et al. [7, 10] extends BPMN semantics to include both performance and reliability modeling, creating a bridge between activity-based and resource-based paradigms.

[1] https://github.com/bptlab/scylla

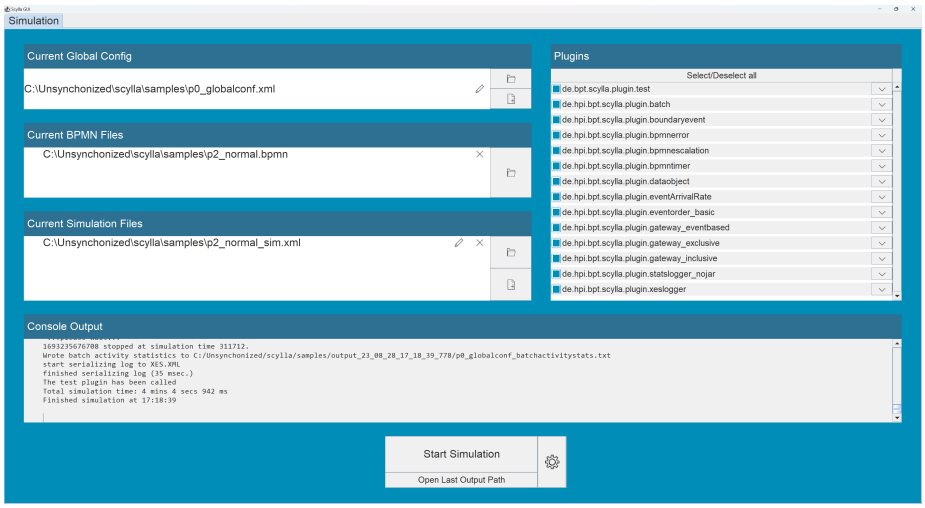

Figure 3.2: Scylla view[1]

PyBPMN aims to represent not only how processes behave under nominal conditions but also how they respond to failures, repairs, and redundancy mechanisms. They are all key concerns in manufacturing and service systems where resource dependability affects throughput.

PyBPMN introduces new meta-classes into the BPMN meta-model to explicitly capture the behavior of resources that execute process activities:

- **PyPerformer** represents an individual resource (human, machine, or system) executing an activity, with attributes such as service time and availability.
- **PyBroker** manages multiple alternative resources and defines selection or allocation policies (e.g., round-robin, reliability-first, or performance-first).
- **PySubsystem** defines composite or redundant resource groups, modeling configurations such as hot, cold, or warm standby.

Each resource entity can be characterized by reliability parameters such as mean time to failure (MTTF) and mean time to repair (MTTR), allowing simulation of both operational and degraded system states. This performability perspective integrates failure and repair behavior into the same model used for performance analysis.

From a methodological standpoint, PyBPMN follows model-driven engineering (MDA) principles. It enables automatic transformation of enriched BPMN diagrams into executable models, ensuring consistency between high-level design and simulation analysis. By embedding reliability attributes directly into process models, PyBPMN allows analysts to assess not only time-based performance metrics but also system resilience, fault tolerance, and recovery behavior.

In essence, PyBPMN represents a hybrid evolution of BPMN: it retains the clarity of activity-based modeling while incorporating the realism of resource-oriented simulation. It thus serves as a conceptual bridge to the next class of tools: resource-based simulators, which model these constraints and dynamics natively.

3.4.2 Resource-based modeling and simulation tools

On the other hand, resource-based modeling centers on the entities that perform work such as machines, workers, vehicles, and on the queues and events that govern their interactions. This paradigm directly reflects the operational constraints of manufacturing and logistics environments, where performance depends on the utilization and synchronization of tangible resources.

Modern resource-based simulation tools, both commercial such as FlexSim and academic as Warteschlangensimulator‘[42], exemplify this approach. They provide object-oriented modeling frameworks in which users construct systems by connecting predefined resource elements such as, processors, conveyors, operators, and queues within a virtual workspace. Processes emerge from the dynamic interaction of these components over simulated time.

FlexSim, for example, offers a 3D modeling environment that visualizes factory layouts and process flows in real time. Its object library encapsulates common manufacturing behaviors (processing, transporting, waiting), while scripting interfaces enable customization of control logic. An example of the complex models FlexSim is capable of is shown in Figure 3.3.

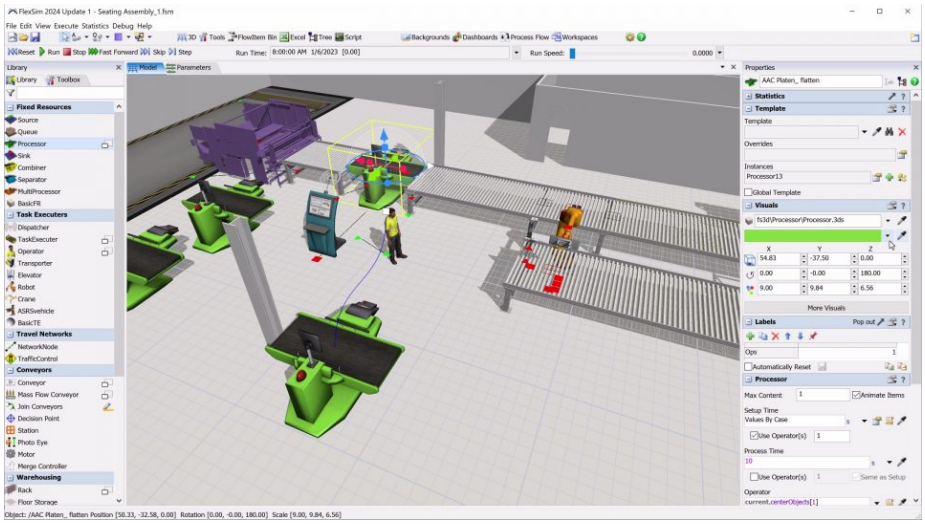

Figure 3.3: FlexSim modeler view[2]

[2] https://www.flexsim.com/wp-content/uploads/2024/04/flexsim-24-1-scaled-e1713145666625.jpg

The Warteschlangensimulator, on the other hand, adopts a more simplistic approach: its 2D environemnt emphasizes transparency of queueing mechanisms, supporting experiments with different scheduling policies, service disciplines, and stochastic parameters. An example of the Warteschlangensimulator environment is shown in Figure 3.4.

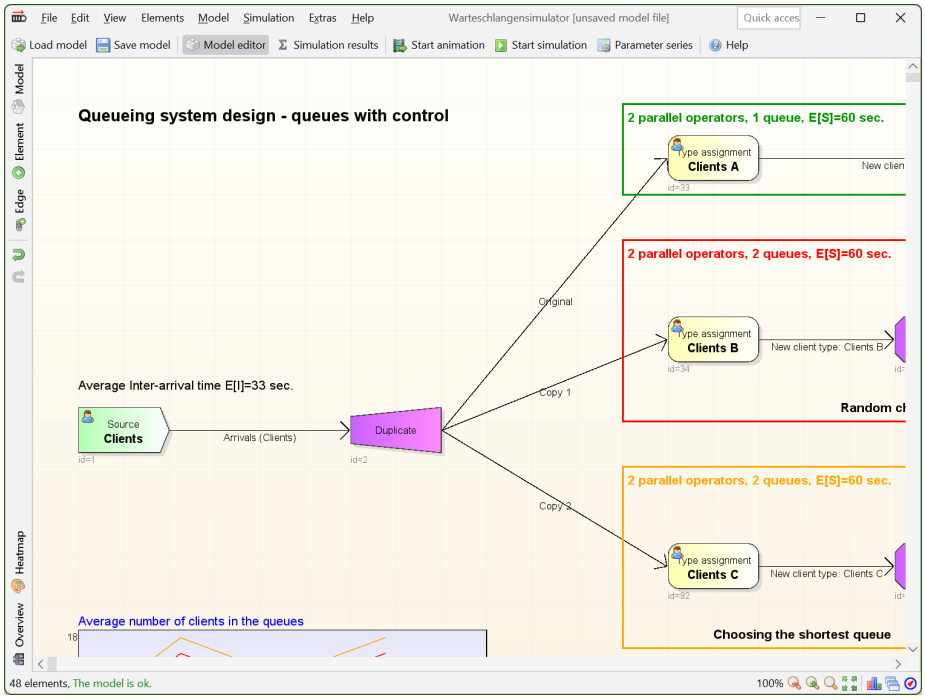

Figure 3.4: Warteschlangensimulator modeler view[3]

Both tools enable the user to run extensive what-if analyses, testing how system performance responds to changes in capacity, layout, or variability, thus supporting tactical decision-making.

Resource-based tools enable the user to run extensive what-if analyses, testing how system performance responds to changes in capacity, layout, or variability, thus supporting tactical decision-making. They explicitly capture variability, blocking, and resource contention, making them ideal for throughput and bottleneck studies. Their main limitation lies in complexity: detailed data is required, and model construction can be time- and skill-intensive.

3.4.3 Integrating activity- and resource-based perspectives

The two paradigms are not competitors but complements. Activity-based modeling clarifies what work must be done and in what order, providing a communica-

[3] https://a-herzog.github.io/Warteschlangensimulator/

tive and conceptual blueprint. Resource-based modeling reveals how that work unfolds under real-world constraints. Integrating both allows tactical analysis that is both conceptually sound and operationally accurate. For instance, a BPMN model may capture the overall workflow of an order fulfillment process, while a resource-based simulation explores how machine failures or operator shifts affect its actual throughput. Petri nets, with their formal semantics and ability to capture concurrency, provide a theoretical bridge between these perspectives, grounding high-level process logic in rigorous execution dynamics.

In summary, process modeling and simulation tools, whether activity- or resource-oriented, serve as the analytical foundation for tactical decision support in manufacturing. They allow analysts to experiment safely with alternative configurations, quantify outcomes, and build a coherent understanding of how local operational changes propagate through the entire system. This integrative view underpins the methodology developed in the present research.

Chapter 4

Methodology

Understanding and formalizing the methodological path followed in this research is essential to establish the scientific validity, reproducibility, and coherence of the proposed approach. This chapter presents the methodological foundations, design rationale, and evaluation strategy that guided the development of the meta-model and its supporting tools for resource-aware, activity-based manufacturing process modeling and simulation described in Chapter 5, 6, and 7.

The research adopts a design science perspective [14], where knowledge is generated through the construction and iterative refinement of artifacts, a meta-model, a modeling tool, and an associated simulation platform. The methodology thus combines conceptual, technical, and empirical components: it first defines a conceptual framework, then translates this foundation into a concrete implementation, and finally validates it through domain-specific case studies and expert evaluations.

The chapter is structured into three main sections:

1. Section 4.1 introduces the meta-model and tool design methodology. It describes the underlying concept, the modeling scope and trade-offs, and the positioning of the proposed approach with respect to BPMN. The section further details the methodological stages that guided the development process, from requirements elicitation and conceptual framework definition to tool implementation and iterative validation.

2. Section 4.2 presents the systematic literature review conducted to ground the research in the state of the art. The review integrates automated data retrieval and manual expert validation to identify and synthesize relevant contributions across the domains of simulation, manufacturing, and business process modeling. The resulting corpus provided both theoretical context and practical requirements for the meta-model and its associated tools.

3. Section 4.3 outlines the evaluation framework adopted to assess the proposed approach. It distinguishes between formative evaluation, conducted

iteratively during development, and summative evaluation, planned to validate the final integrated platform with manufacturing experts. These complementary phases ensure that the proposed solution is conceptually sound, technically consistent, and usable by its intended audience for tactical decision-making.

4.1 Meta-model and tool design

This section describes the design of the proposed meta-model and its supporting tools. It first introduces the overall research goals and modeling scope, then positions the proposal with respect to BPMN, highlights key representation and tool implications, and finally details the conceptual framework and methodological process that guided development.

4.1.1 Overview and goals

The work builds on the motivations introduced in Chapter 2 and follows three main directions:

1. **Defining the modeling boundaries:** establishing the conceptual limits and objectives of the proposed modeling approach, leading to the formulation of a conceptual framework that balances simplicity, descriptive adequacy, and intuitiveness in representing manufacturing systems.
2. **Developing a systematic methodology:** designing a structured process to create a meta-model that complements existing activity-based formalisms with resource-oriented constructs essential for tactical manufacturing simulation.
3. **Designing and implementing the artifacts:** applying this methodology and conceptual framework to the concrete design and realization of both the meta-model and its supporting modeling and simulation tools.

Together, these directions define a coherent progression from conceptual foundations to operational implementation, ensuring consistency between theoretical rationale, meta-model definition, and tool functionality.

4.1.2 Modeling scope and trade-offs

Before delving into design aspects, it is useful to clarify the scope of the models that this work intends to support. Any model is, by definition, an abstraction of reality whose effectiveness depends on balancing three key characteristics:

1. **Simplicity:** minimizing complexity to facilitate model authoring, sharing, and interpretation.
2. **Descriptive adequacy:** ensuring that the model faithfully represents relevant aspects of reality within a defined *viewpoint*.

3. **Intuitiveness:** maintaining alignment with the mental models of domain experts by leveraging familiar abstractions.

These characteristics resist universal quantification. A viewpoint reflects subjective priorities, and intuitiveness depends on user expertise, an expected consequence of the fact that a meta-model must be judged against its intended purpose [19].

In this work, the goal is to enable instance models that, when used for tactical simulation analysis, produce metrics sufficiently aligned with real-world system behavior to inform decision-making. For example, achieving extremely precise estimates of the lead time of a production process would require capturing granular details such as machine layout and component transit times, typically handled by complex code-augmented models (e.g. FlexSim). However, such fidelity entails significant development effort and specialized skills. Here, we deliberately privilege *practical utility* over exhaustive detail: the models are designed for tactical what-if analyses that evaluate process restructuring or resource allocation using metrics that are *close enough* to reality to guide business decisions confidently. By emphasizing high-level insights rather than on-the-spot accuracy, we lower the barrier to iterative, user-driven exploration.

Positioning with respect to BPMN

The proposed meta-model extends BPMN to integrate a resource perspective, while remaining compatible with its underlying activity-based foundation. Many BPMN extensions exist in the literature, enabled by the explicit extension mechanism of BPMN [13]. They space in a variety of fields: from healthcare [63] to social and collaborative processes [12].

Our proposal does not attempt to "improve" BPMN: its success in business process management stems from two fundamental design principles:

1. Domain-agnostic generality and

2. A tight meta-model-to-notation correspondence enabling full semantic interpretation from diagrams.

To incorporate a rich resource dimension, we deliberately deviate from both principles. First, we define a domain-specific BPMN *dialect*. More critically, we introduce meta-model elements that have no direct notational counterparts, meaning process semantics can no longer be entirely captured by the diagram alone. Instead, they must be complemented by ancillary structures such as compatibility matrices (see Section 5.3). This deviation is intentional: it enables a more expressive treatment of manufacturing resources while preserving BPMN's recognizable core.

Representation choices

Introducing non-notational meta-model elements carries practical consequences. Model interpretation now requires access to supplemental information beyond

the graphical layer. Accordingly, a dedicated modeling tool was developed with two complementary capabilities:

- Standard BPMN-like diagramming for activity and flow representation, and
- Specialized widgets for editing and inspecting non-notational elements (e.g., resource compatibility matrices or parameter tables).

This dual design ensures that users can work both visually and semantically with the model. Most quantitative or contextual details, such as expected times for specific executors, are stored as properties within model elements rather than explicitly shown in diagrams. This choice improves readability while retaining necessary detail within the underlying data model.

It is worth noting that most of these details will be captured by various properties associated to the elements of the model. However, for the sake of improving the readability of the proposed meta-model, we will not explicitly represent most of these details in diagrams and images. We want to focus on the definition of the elements and their relationships.

Conceptual framework

The considerations above motivated the definition of a conceptual framework that formalizes the intended balance between simplicity, descriptive adequacy, and intuitiveness. This framework establishes the abstract relationships among activities, resources, and the entities that mediate their interaction, delimiting the scope of what the proposed models aim to represent. It acts as a bridge between the qualitative modeling intent, supporting tactical what-if analysis, and the quantitative requirements of simulation. By structuring how activities depend on and utilize resources within manufacturing processes, the framework provides a coherent foundation for the subsequent meta-model definition and tool implementation. In essence, it captures the conceptual boundaries of the modeling approach, ensuring that the following design decisions remain aligned with the practical objectives of accessibility, analytical depth, and domain relevance.

4.1.3 Design and implementation process

Building on the conceptual groundwork, the meta-model and its supporting tools were developed following a structured, iterative methodology composed of four main stages.

These stages were iterated until stakeholder consensus was achieved. Continuous validation is a defining feature of this co-design process. While external validation (ongoing) will strengthen the generalizability of results, the current outcomes already reflect insights from professionals with decades of experience advising manufacturing clients. For presentation clarity, later sections may describe this process as a linear sequence, though its iterative character remains central to the methodology (see Fig. 4.1).

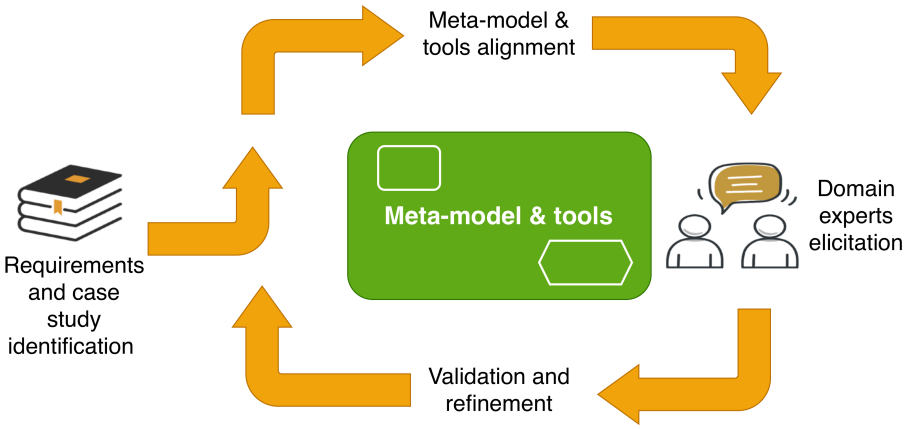

Figure 4.1: Methodology representation

Requirements and case study identification

A hybrid process combining literature analysis and expert elicitation was used to define modeling requirements. Two domain experts from an industrial consulting firm participated in a series of interviews and participatory design sessions [80]. During these sessions, real manufacturing scenarios were collaboratively modeled using BPMN, allowing the research team to identify gaps between existing notational capabilities and practical modeling needs. The resulting insights were consolidated into a formal set of meta-model requirements. Three case studies of increasing complexity were also co-developed to represent a range of manufacturing contexts.

Meta-model design

The meta-model was iteratively designed to satisfy the collected requirements while maintaining conceptual consistency with both the activity-based modeling paradigm and the proposed resource-oriented extensions.

Tool alignment

The modeling and simulation tools were aligned to support the designed meta-model. This alignment ensured that instance models could be created and executed correctly, maintaining compatibility with the simulation engine. The non functional requirements for the tool are better described in 4.1.4.

Validation and refinement

The solution was evaluated through the developed case studies and iteratively refined in collaboration with domain experts. Any discrepancies between expected

and simulated outcomes prompted revisions either to the requirements or to the meta-model constructs themselves.

4.1.4 Choice of technologies and tools

The design of the modeling and simulation platform was guided by several non-functional requirements stemming from the research goals and the target user context. In addition to basic functional adequacy (supporting BPMN-based process modeling and discrete-event simulation), the decisive factors for selecting the underlying technologies were aligned with best practices reported in literature:

- **Performance and efficiency**: High execution speed is crucial to avoid bottlenecks from language or runtime limitations. Simulation researchers note that the choice of implementation language can significantly impact performance; for example, using interpreted languages can introduce overhead, though just-in-time (JIT) compilation can partly bridge the gap with compiled languages [76]. Especially for manufacturing process simulations, efficient engines enable extensive what-if analyses without disrupting real operations [54].

- **Modernity and maintainability**: Using contemporary, expressive, and type-safe programming languages improves code clarity and long-term maintainability [40]. Modern simulation frameworks often favor high-level languages for their ease of use and rapid development benefits. For instance, the kalasim library rebuilt a Python-based simulator in Kotlin to gain a more type-safe API and better test coverage, while preserving clarity and adding modern visualization support. Adopting well-structured, modular architectures in a modern language makes the tool easier to extend and maintain over time, which is essential in fast-evolving domains.

- **Reliability**: Rather than reinventing core simulation logic from scratch, the platform leverages well-tested engines and paradigms. Many state-of-the-art simulation tools are built on decades of development and use the proven process-oriented DES approach introduced by Simula, yielding clear and easy-to-maintain models [39]. Reusing such battle-tested scheduling algorithms and frameworks greatly reduces the risk of errors in complex event-handling code. For example, kalasim began as a rewrite of the mature salabim engine, explicitly reimplementing its core APIs to take advantage of an established design and ensure correctness.

- **Accessibility**: The tool is designed for web-based use, requiring no installation or configuration for end users. Web-based simulation offers significant accessibility advantages, allowing users to run models through a standard browser interface. This trend is reflected in modern simulation offerings – for example, commercial platforms like WITNESS now support

cloud-based simulation access, and free web services exist for BPMN-driven process simulation that enable a "fast start" with no local setup. By deploying the simulation platform via the web, we ensure that business analysts and other stakeholders can easily access and experiment with process models anywhere, fostering broader adoption [15].

- **Ecosystem maturity**: Finally, leveraging technologies with large, active ecosystems of libraries and users was a key consideration. A rich ecosystem accelerates development by providing ready-made components for visualization, data analysis, optimization, etc., and it improves long-term sustainability of the platform. For example, Python-based simulators like SimPy can tap into the extensive scientific Python stack for statistical analysis and plotting of simulation outputs [91]. Established tools also often come with advanced features out-of-the-box thanks to their community; AnyLogic, to illustrate, includes built-in optimization routines (e.g. a genetic algorithm module) to fine-tune simulation parameters [53].

Each of these criteria, performance, modern maintainable design, proven reliability, web accessibility, and ecosystem support, guided the selection of technologies for the platform. This ensured that the resulting simulation tool would be efficient in execution, easier to develop and trust, readily accessible to end users (especially in business process and manufacturing contexts), and sustainable in the long run.

4.2 Literature review

A systematic literature review was conducted to identify, analyze, and synthesize the most relevant scientific contributions across the foundational topics of this research: (i) simulation, (ii) manufacturing, and (iii) business process modeling (with particular emphasis on BPMN). The objective of the review was twofold: to establish a comprehensive understanding of the state of the art and to derive requirements for the meta-model and supporting tools capable of bridging process modeling and manufacturing simulation presented in Section 5.2.1.

4.2.1 Review Objectives and Scope

The literature review aimed to identify modeling approaches, meta-modeling frameworks, and simulation methodologies that could inform the design of a resource-aware, activity-based modeling environment. To ensure methodological rigor, the process combined both automated and manual stages. The automated component maximized coverage across large bibliographic datasets, while the manual phase ensured the inclusion of only conceptually relevant and high-quality studies.

4.2.2 Search Strategy and Query Design

The review began with the manual identification of ten seminal papers for each of the three domains of interest [1]. These papers were selected based on their citation count, methodological relevance, and representativeness of the research field. They formed a reference corpus against which all subsequent works were compared for relevance.

Building on this corpus, six search queries were formulated to capture the key concepts and interrelations between simulation, manufacturing, and process modeling. Each query used combinations of domain-relevant keywords and logical operators. The "+" symbol denoted the logical operator *AND*, while quotation marks specified exact phrase matching. The resulting queries were as follows:

1. "resource modeling" + in + "process simulation";
2. "business process models" "manufacturing operations";
3. "batch activity" + in + "process modeling";
4. "BPMN extensions" "manufacturing processes";
5. "conceptual model" + for + "manufacturing simulation";
6. "core manufacturing simulation data".

These queries were executed using the *SemanticScholar*[2] and *OpenAlex*[3] APIs, which provide programmatic access to extensive bibliographic repositories. For each retrieved publication, metadata, references, and citation networks were collected. This process expanded the dataset beyond direct search results, ensuring broader coverage of related works.

4.2.3 Data Consolidation and Filtering

All retrieved records were merged into a single working dataset to enable systematic filtering and refinement. The data cleaning and filtering process followed several sequential steps:

1. **Traceability filtering:** records lacking a Digital Object Identifier (DOI) were excluded to guarantee verifiability and reproducibility.
2. **Disciplinary relevance:** publications were retained only if they have been classified by the publisher within the domains of *Computer Science* or *Engineering*, corresponding to the scope of this research.

[1] https://doi.org/10.5281/zenodo.17419015
[2] https://www.semanticscholar.org
[3] https://openalex.org

3. **Similarity-based filtering:** a text-similarity analysis was conducted on titles and abstracts (when available), comparing each paper against the reference corpus. To do so, we used SBERT's Semantic Textual Similarity [73]. This generated a cosine similarity score in the range $[0, 1]$, where higher values indicated closer thematic alignment. To ensure cross-domain relevance, only papers with a second-highest similarity score of at least 0.67 were retained, guaranteeing that each paper contributed meaningfully to at least two of the three domains.

4.2.4 Manual Review and Validation

Following automated filtering, the resulting dataset underwent a structured manual validation phase to ensure conceptual relevance and methodological consistency. The dataset was divided into three subsets, and each publication was independently reviewed by two evaluators. Reviewers classified each paper as either *relevant* or *not relevant*. Papers receiving two positive assessments were automatically included, while those receiving two negative assessments were discarded. In cases of disagreement, a third reviewer acted as an arbiter, making the final inclusion decision.

This two-step procedure, automated filtering followed by human validation, combined the advantages of computational scalability and expert judgment. It ensured that the final corpus included only publications that were both technically relevant and conceptually significant.

Figures 4.2 recaps the steps performed for this systematic review of the literature. In blue there are the steps which have been automatized and in green the one manually performed

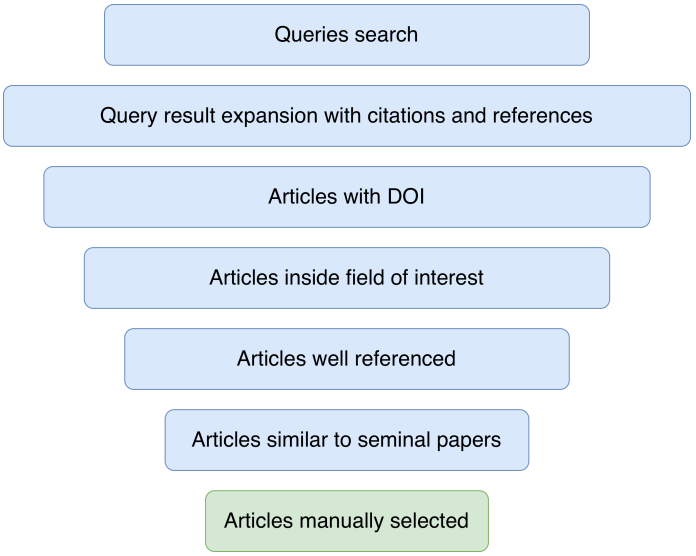

Figure 4.2: Funnel process for the Systematic Literature Review

4.3 Evaluation

The evaluation of the proposed approach is organized in two complementary phases that target distinct objectives and maturity levels of the research outputs. The first phase (*formative evaluation*) has been completed and focused on iterative assessment of the meta-model and tool during development. The second phase (*summative evaluation*) is planned and will validate the final integrated product with manufacturing domain experts who do not possess programming skills.

The overall evaluation pursues three primary objectives:

1. **Conceptual validation:** to verify that the proposed meta-model adequately captures the relevant concepts and relationships required to represent manufacturing processes at the tactical level.

2. **Usability and accessibility assessment:** to evaluate whether domain experts can use the developed modeling environment effectively without requiring programming or simulation expertise.

3. **Decision-level consistency:** to demonstrate that the simulation results generated by the proposed approach are sufficiently aligned with those obtained from established high-fidelity simulation tools to support confident tactical decision-making.

4.3.1 Formative Evaluation: Iterative Validation of the Meta-Model and Tool

The first evaluation phase was interwoven with the development of the meta-model and its supporting tools, following an iterative and co-design process. The goal was to ensure that the evolving solution remained conceptually sound, practically relevant, and usable for its intended audience.

Context and Participants

This phase was carried out in collaboration with two domain experts from an industrial consulting firm specializing in manufacturing process analysis. The experts contributed to defining requirements, co-developing case studies, and iteratively refining both the meta-model and the associated modeling and simulation tools.

Two representative manufacturing processes were selected as reference case studies. These processes were modeled and analyzed using the ARMS prototype developed in this research. After each iteration, structured feedback sessions were conducted to assess (i) the expressiveness of the meta-model, (ii) the usability of the modeling tool, and (iii) the interpretability of the generated simulation results.

Tactical versus Operational Simulation Analysis

To frame the evaluation, it is important to distinguish between tactical and operational analyses. Tactical analyses focus on medium-term decision-making (weeks or months), such as capacity planning, resource allocation, or outsourcing strategies. Operational analyses, by contrast, address short-term, day-to-day optimization issues (e.g., layout design, transfer speeds, or machine cycle times).

The approach proposed in this research deliberately targets the tactical level, privileging model simplicity and interpretability over fine-grained physical accuracy. Accordingly, the evaluation emphasized whether ARMS could produce simulation results that lead to the same tactical conclusions as detailed operational models, even if numerical KPI values differ slightly.

Reference Modeling Environment

To establish a benchmark for assessing ARMS outputs, high-fidelity models of the same processes were built in FlexSim, a professional discrete-event simulation tool.

The construction of FlexSim models required the definition of both physical and logical process components. Logical behavior was implemented through *ProcessFlow* and *FlexScript*, FlexSim's proprietary scripting language.

ProcessFlow enables the specification of process logic through a library of pre-defined activities. However, complex behaviors such as initialization routines, conditional stop criteria, or dynamic data exchange between elements required the use of FlexScript code. Data were managed via global queues acting as shared variables accessible by multiple simulation elements.

Simulation results from FlexSim served as a reference for evaluating the decision-level consistency of ARMS results.

Modeling Procedure

For each manufacturing process, three scenarios were developed:

1. A baseline (*as-is*) scenario.
2. A first tactical what-if scenario introducing moderate configuration changes.
3. A second tactical what-if scenario representing a significant structural modification.

Equivalent scenarios were modeled and simulated in both ARMS and FlexSim, using consistent process logic and input data. Differences between results were analyzed to assess whether both tools would lead decision makers toward the same tactical conclusions.

Evaluation Metrics and Decision Alignment

Simulation results were compared using the following key performance indicators (KPIs). We take the lead time (the total production time for a fixed number of

items to produce) as the main KPI to drive the comparison between ARMS and FlexSim.

Decision alignment was defined as the degree to which ARMS and FlexSim results supported the same tactical decision (e.g., both recommending additional operators to reduce lead time). Under this criterion, numerical discrepancies were acceptable provided that strategic recommendations remained consistent. This ensured the evaluation reflected the intended abstraction level of the proposed approach.

4.3.2 Summative Evaluation: Validation of the Final Product

This subsection outlines a summative evaluation plan intended to assess the usability, learnability, and decision-support effectiveness of the fully integrated ARMS platform. It constitutes a set of guiding principles that must be adapted to the characteristics, constraints, and priorities of the specific deployment context before any data collection occurs. In other words, the descriptions that follow present a project plan rather than results from an evaluation already carried out.

Evaluation Objectives

The evaluation pursues three complementary objectives, stated here as guidelines to be operationalized in the target setting. First, it examines usability and learnability by asking whether non-technical users can independently create, modify, and simulate process models through the graphical interface. Second, it explores expressiveness and adequacy, considering whether the BPMN-based constructs and resource extensions are sufficient to capture salient manufacturing process characteristics. Third, it investigates decision-support capability by verifying whether users can derive meaningful and actionable insights from simulation results. These objectives specify what to evaluate. The exact operational measures and thresholds are to be agreed with stakeholders in the specific industrial context.

Participants and Setting

The plan envisages involving a small cohort of approximately four to six manufacturing experts recruited from the partner consultancy firm. This range is indicative and should be adjusted to the availability and representativeness requirements of the actual site. Participants are expected to reflect diverse roles, such as process engineers, analysts, and operations managers because we want that multiple perspectives are captured. Sessions are conceived as structured workshops in a laboratory environment configured to approximate real-world modeling tasks. Researcher involvement is limited to a short introductory briefing. All subsequent modeling and simulation activities are intended to occur

without assistance. These conditions are design targets to guide local tailoring rather than fixed constraints.

Procedure

Each session is designed to unfold in four stages, with timings and materials serving as templates to be customized on site. The session begins with a tutorial of about 30 minutes that introduces the interface, principal modeling elements, and the simulation workflow. Participants then work independently to model a manufacturing process described through textual and tabular materials that mirror typical industrial documentation. In practice, these materials should be replaced or augmented with documents from the host organization. After establishing the baseline model, participants define and run at least two what-if scenarios, such as adding resources, changing machine assignments, or adjusting batch sizes, using only the graphical interface. The session concludes with analysis and debrief: participants review key performance indicators, explain the rationale behind their decisions, and engage in a semi-structured interview focused on usability, perceived accuracy, and overall satisfaction. All durations, artifacts, and sequencing here are provisional and subject to adaptation to the specific reality.

Data Collection and Analysis

A mixed-methods approach is proposed as a guideline for capturing both quantitative and qualitative evidence of effectiveness. On the quantitative side, indicative measures include the total time required to model and simulate scenarios, the number and type of modeling or configuration errors observed, the proportion of participants completing all tasks without assistance, and the percentage deviation of key results, such as lead time, utilization, and cost relative to reference simulations. On the qualitative side, suggested sources include post-session questionnaires based on the System Usability Scale (SUS), observational notes from think-aloud sessions documenting user reasoning, difficulties, and strategies, and thematic analysis of interview responses on perceived usefulness, abstraction level, and modeling flow. These instruments and metrics are exemplars. The final operationalization (including instruments, benchmarks, and analysis scripts) must be agreed with stakeholders and adapted to the host site's data availability, confidentiality requirements, and decision-making cadence. No data have yet been collected, and the analyses described indicate the planned methods rather than completed work.

Expected Outcomes

The expected outcomes are articulated as anticipated benefits contingent on successful local adaptation and execution of the plan. Specifically, the evaluation aims to generate empirical indications that the tool is accessible to domain experts without programming expertise, to provide evidence supporting the

expressive adequacy of the meta-model in realistic manufacturing contexts, and to confirm that simulated results, despite their abstraction, align with tactical decision-making needs. These expectations are hypotheses to be validated (or revised) once the evaluation is instantiated within the specific organizational setting.

Together, the formative and summative evaluations ensure that the proposed framework and tools are not only conceptually sound and technically coherent but also usable and valuable to their intended audience, manufacturing professionals seeking accessible modeling and simulation capabilities for tactical analysis.

Chapter 5

Modeling and Simulation Framework for Tactical Manufacturing Analysis

This chapter translates the methodological principles defined in Chapter 4.1 into a concrete modeling and simulation framework designed to support tactical analysis of manufacturing systems. Its objective is to build the conceptual and operational foundations upon which simulation-based reasoning, policy testing, and decision support can be consistently developed.

We begin by introducing a conceptual framework that separates the process world activities and their organization from the resources world and the entities that execute or are affected by those activities. This abstraction layer provides a domain-agnostic foundation: while the focus is on manufacturing, the same principles apply to any system where transformations occur under resource constraints.

Next, we perform a requirements identification phase that establishes what the meta-model must represent to support accurate and analyzable manufacturing simulations. This step combines insights from a systematic literature review, covering business process modeling, simulation, and manufacturing domains, with knowledge elicited from domain experts. The outcome is a consolidated set of modeling requirements (R1–R9) that guide the design choices of the meta-model.

Building on these requirements, we propose the meta-model that constitutes the core technical contribution of this chapter. The meta-model operationalizes the conceptual framework by defining entities, relationships, and attributes that describe both the static and dynamic aspects of manufacturing systems. It explicitly captures how activities, executors, and products interact over time, enabling the creation of reproducible, resource-aware, and analyzable simulation instances.

Finally, we introduce the notation and representation scheme used to express

models based on the proposed meta-model. This notation extends familiar BPMN constructs with additional elements for executors, inventories, and transformation rules, and integrates tabular representations for complex relationships such as compatibility and transformation matrices. These conventions ensure clarity, reduce diagrammatic clutter, and maintain compatibility with modeling tools and simulation engines.

Throughout the chapter, a running example is used to anchor the discussion. It illustrates how the conceptual framework, requirements, meta-model, and notation come together to form a coherent and executable modeling approach, setting the stage for the simulation studies presented in the subsequent chapters.

5.1 Conceptual framework

As explained in Section 4.1, the conceptual framework developed in this work is intentionally abstract, designed to provide a clear but flexible foundation for subsequent modeling. As illustrated in Figure 5.1, it consists of two interconnected "worlds": the *process world* and the *resources world*.

The process world assumes a generic activity-based meta-model. It does not prescribe a specific structure, but requires only the existence of two fundamental concepts: *Process* and *Activity*. The concept of *Activity* acts as the bridge between the process and resource worlds. In Figure 5.1, dashed lines are used to denote connections that remain underspecified, while the dashed cloud indicates that the concepts of *Process* and *Activity* belong to a broader meta-model, which will be elaborated later.

The resources world encompasses the entities required to perform or support activities. The two primary resource types are: *Executor*s and *Processable*s. *Executor*s represent active agents capable of carrying out activities, such as human operators or machines. *Processable*s, by contrast, are passive elements that contribute to or are transformed by the process. These include raw materials, intermediate components, finished products, and tools. Depending on the context, processables may function either as *products* (inputs or outputs of the activities) or as *accessories* (tools required to execute them). These distinctions are not captured in detail at the conceptual framework level, but are addressed in the subsequent meta-model.

Although deliberately minimal, this framework is a crucial step in establishing a coherent and unified perspective for the development of a detailed meta-model supporting the simulation and analysis of manufacturing processes. Furthermore, the level of abstraction adopted ensures that the framework can be applied beyond manufacturing, making it suitable for a wide range of domains where processes and resources interact.

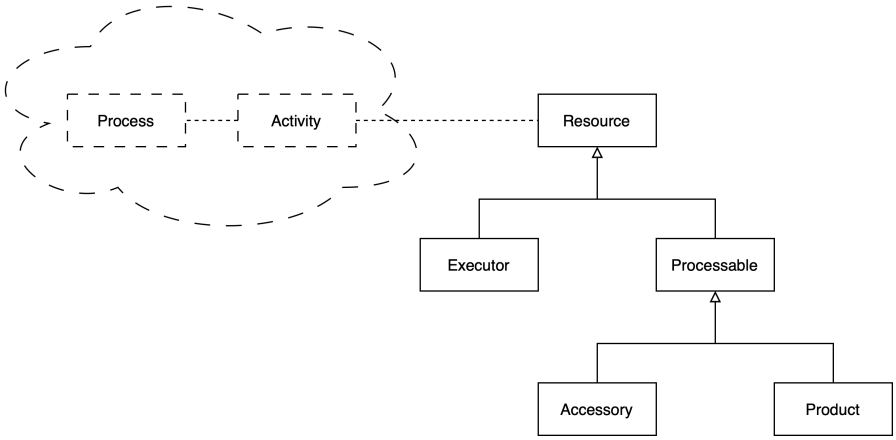

Figure 5.1: Conceptual framework

5.2 Requirements identification

5.2.1 From literature

Building on the methodology described in Chapter 4, we initiated the design of the meta-model by identifying its underlying requirements. To this end, we conducted a Systematic Literature Review (SLR) following the process detailed in Section 4.2. The review aimed to extract and consolidate modeling and simulation needs emerging from three intersecting domains central to this work: *simulation*, *business process modeling*, and *manufacturing*.

The process began with the selection of ten seminal papers representative of these domains. Using an automated collection tool, we expanded this initial corpus by retrieving related articles, their references, and citations. The resulting dataset comprised 1098 papers. We then applied a series of filtering steps to ensure quality and relevance. First, we retained only papers with at least two citations and belonging to the fields of Computer Science or Engineering, yielding 831 papers. Next, we refined the set through a similarity-based selection, comparing titles and abstracts against the seminal works and retaining only those related to at least two of the three domains, obtaining 470 articles. Finally, through a manual selection, we identified 16 key papers from which modeling challenges and requirements could be derived. The process recap and its results are presented in Figure 5.2.

BPMN [65], as the de facto standard for activity-centric modeling, naturally served as the main reference point. Its extensibility has led to numerous domain-specific adaptations, as analyzed in [88], where 52 BPMN extensions are reviewed, only two of which directly address manufacturing contexts. These two, along with other related works, formed the starting point for requirement extraction.

The first of these works, [2], motivated two important requirements. The

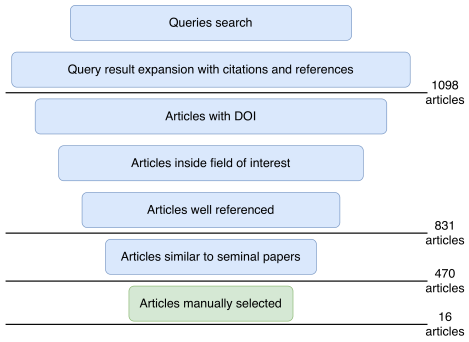

Figure 5.2: Figures for the SLR's funnel process

authors propose extending BPMN to represent task scheduling and to associate an execution time with each task. In their model, the execution time is a property of the *Task* entity. However, domain experts consulted in our study emphasized that execution time often depends on the executor performing the task. Consequently, instead of associating time directly with the activity, we consider it as a derived property that depends on the specific executor assigned to that activity (**R9: Executor-based activity duration**, detailed later). The same paper also extends BPMN with the concept of *inventory*, allowing the tracking of available items, unsold stocks, and provisions used by activities. This insight directly informed **R6: Inventory**, capturing the notion of process interactions with stored products and materials.

The second relevant study, [70], focuses on representing human physical risks in manufacturing through BPMN extensions. While the notation proposed is expressive and practically valuable, it does not introduce new requirements pertinent to the modeling objectives of this dissertation.

Another influential contribution is [9], which introduces the notion of resources and simulation-based analysis for "Cyber-Physical Production Systems". Although this work emphasizes IoT-driven integration and detailed resource modeling, its focus diverges from our goal of representing the complex interplay among activities, executors, and products in a generalizable manner.

In [92], the authors address the modeling of manufacturing processes in BPMN, particularly emphasizing product flows and the combination or separation of materials. Their extension introduces manufacturing-specific tasks, resource concepts (machines, tools, parts, materials), and novel "material gateways" for handling converging and diverging flows. However, product details remain abstracted, revealing the need for more granular constructs to represent transformations, leading to **R1: Product combination/separation**.

A similar limitation is noted in [29], which catalogs manufacturing operations and their BPMN mappings. The authors also highlight the modeling of *batch activities*, where tasks accumulate items until certain conditions are met (e.g., loading a washer that operates on 15 items simultaneously). This idea, also explored in [71], directly motivates **R2: Batch activities**.

Beyond BPMN, several works compare alternative notations for manufacturing workflows. [32] contrasts BPMN, VSM, and IDEF, showing that BPMN lacks mechanisms for representing tangible objects and their transitions, while VSM remains largely qualitative and manual. Similarly, [8] compares BPMN, Processing Networks, DPMN, and PyBPMN, analyzing how activities and resources interact during execution. The authors discuss several challenges related to executor assignment and prioritization, issues directly translatable into our requirements. Specifically, when multiple activities compete for the same resource, prioritization mechanisms become essential (**R7: Activity priority**). The same study notes that an executor may perform multiple activities on the same resource, emphasizing the benefit of maintaining product continuity (e.g., a single operator performing all quality checks on the same product), which informs **R3: Executor affinity**. Moreover, they describe cases where multiple executors or resources jointly perform a task. For example, assembly lines or multi-tool operations, leading to the requirement for representing composite executor relationships (**R5: Multi-executor dependency**).

Among these works, only [9] and [8] explicitly integrate simulation execution. Two simulation-oriented initiatives are particularly noteworthy: *Scylla* [72], an extensible BPMN simulator supporting plugin-based scenarios, and *BPSim* [86], an endorsed standard integrating simulation parameters into BPMN models. Both remain limited by BPMN's inherent resource modeling constraints.

Other authors have approached the problem through domain ontologies rather than activity-centric notations [48, 61]. While these offer highly detailed semantic representations of manufacturing resources, their integration with process-level modeling remains cumbersome.

In the simulation literature, [58] introduces a conceptual framework for WITNESS simulation, and [89] compares modeling approaches (activity cycle diagrams, Petri nets, flow charts) within the EM-Plant software, identifying key modeled entities such as material, information, and resource flows. These contributions reinforce the need to represent production flows explicitly, an aspect already captured by R1. [59] presents the Core Manufacturing Simulation Data (CMSD) standard, encompassing a rich description of layouts, operations, scheduling, and planning. The work highlights the importance of executor selection based on operational parameters (e.g., preferring already active machines over idle ones), leading to **R4: Assignment based on executor's parameters**.

Finally, [75] discusses resource patterns in workflow management systems, classifying how work items are distributed and executed by resources. Although focusing primarily on human workers, many of these patterns generalize to non-human executors, confirming the broader relevance of the requirements identified.

Table 5.1 summarizes the set of requirements distilled from the literature (R1–R7). A bullet indicates that the corresponding paper addresses or implies the associated requirement. The last two rows anticipates the additional requirements later introduced through domain expert elicitation.

Table 5.1: Meta-model requirements

	[29]	[92]	[2]	[8]	[89]	[58]	[59]	[71]	[75]	Elicitation
R1: Product combination and separation	•	•			•	•	•			•
R2: Batch activities	•							•	•	•
R3: Executor affinity				•					•	•
R4: Assignment based on executor's parameter							•		•	•
R5: Multi-executor dependency				•					•	•
R6: Inventory			•							
R7: Activity priority				•						
R8: Relationship between activity, product type, and executor										•
R9: Activity's duration influenced by the assigned executor										•

5.2.2 Requirements Elicitation with domain experts

We involved domain experts in validating and extending the set of requirements identified in the literature. In particular, we applied the participatory design technique to elicit the requirements.

Initially, not every requirements were found by the domain experts elicitation. Only after we explicitly told them we found the concept of inventory (R6) and priority (R7) in the literature, the domain experts confirmed that they are relevant. In addition, the domain experts also highlighted two more challenges to address which were not present in the literature.

First of all, they stressed on the idea of "product/executor compatibility". It is very common, in fact, that a given product can only by produced by a given machine. Note that this cannot be modeled with an activity-centric approach, while this can only be achieved through programming with a resource-centric approach.

The domain experts also suggested another common occurrence: the same task can be executed by different executors, but each of them take a different time to complete that task. For instance, a modern machine could be more efficient than an older one, or their performance can be affected by external factors. Again, with BPMN and other activity-centric approaches, this cannot be modeled, while with all the resource-centric tools we analyzed this can only be achieved through programming.

Thus, we identified two new requirements:

- **R8**: Relationship between activity, product type, and executor
- **R9**: Activity's duration influenced by the assigned executor

5.3 Meta-model

Building on the conceptual framework and guided by the requirements elicited from the literature and domain experts, we now present the meta-model, depicted in Figure 5.3, which serves as the foundation for this research. To satisfy requirements for formal rigor and technical operability, the meta-model is defined using the Ecore meta-metamodel, the de facto industry implementation of the OMG Meta-Object Facility (MOF) [66] standard.

The meta-model delves into the abstract concepts of the process and resources worlds by defining the elements and relationships required for simulation. It is structured around two complementary categories of elements: *structural* and *dynamic*, which are visually distinguished by color in the diagram.

Structural elements constitute the model to be simulated, while dynamic elements are run-time entities that are instantiated and modified at simulation time. Although a simulator capable of ingesting model instances from this meta-model could use its own internal structures to track the system's dynamic evolution, we explicitly include dynamic elements for three reasons that are directly actionable in our solution:

1. **Live system representation** We maintain the instance model as a causally connected view of the running system, following a models@runtime approach [6]: events in the system create and update dynamic elements in the model (e.g., a new *WorkItem* when an activity assigns a job to an *Executor*, an *Assignment* when an executor accepts it, or the change of quantities of a *Product* in an inventory when it is consumed or produced). M@RT approach maintains causally connected models of a system and its context while the system executes, using them as a live knowledge base. This approach brings tractable reflection, proactive decision-making, and explicit assurance into the operational lifecycle.

2. **Encoding an initial history and state** Dynamic elements let the model carry a reproducible starting point for simulation and analysis. For instance, we can pre-populate *Inventories* with on-hand stock and backorder. This avoids ad-hoc simulator bootstrapping code and makes experiments comparable: the same instance model yields the same initial conditions across runs.

3. **Explicit, analyzable resource assignment** By representing job execution as *WorkItem* plus *Assignment* objects, resource usage is first-class: who is doing what, since when, and under which policy. This separation supports concrete runtime policies (e.g., shortest-processing-time first, skill-based routing, batching) and enables what-if analyses (e.g., "what is the effect of adding one *Executor* with cycle time X?", "how many *WorkItems* will miss due dates under policy Y?") directly over model elements rather than simulator internals.

In constructing the static part of the meta-model, our approach began by identifying the primary entities derived from the conceptual framework presented

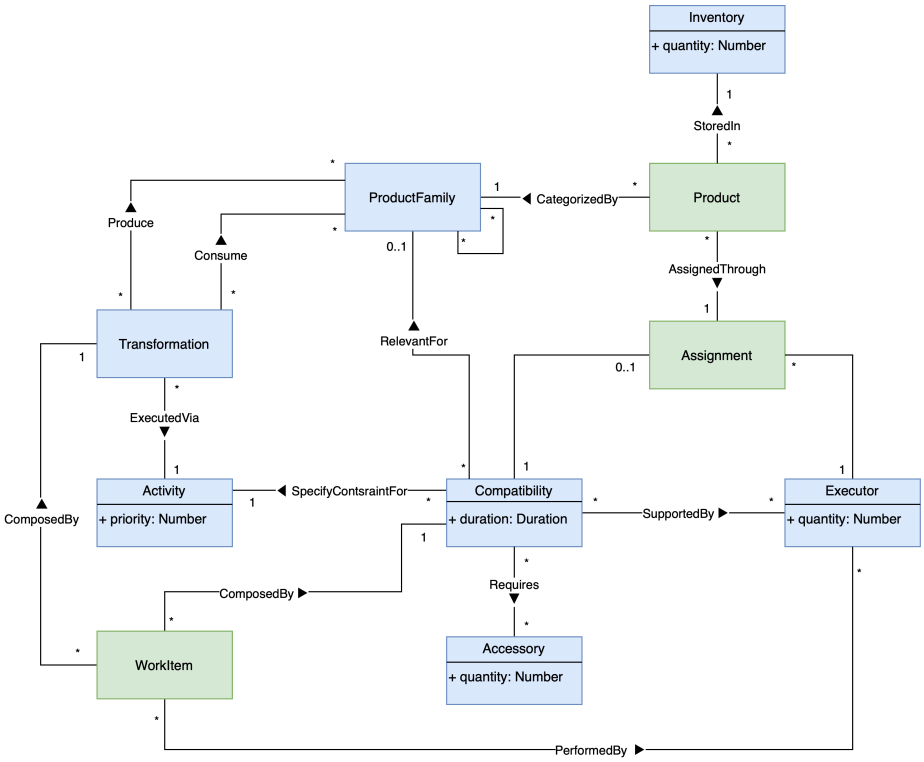

Figure 5.3: Meta-model diagram

in section 5.1. In particular, we started from the concepts of *Activity*, *Executor*, and *Accessory*, which are the leaves of the hierarchy in the framework, since we aim at a concrete meta-model composed of all instantiable elements. The dynamic part of the meta-model roots on *Product*, *WorkItem* and *Assignment*.

We then delineated the associations between these entities, some of which necessitated the inclusion of supplementary entities due to their ternary nature, along with additional attributes. In addition, we added those attributes that were explicitly needed to show the fulfillment of the requirements. Other attributes are left hidden to highlight associations between entities. During the design of the meta-model, we keep two key concepts in mind: (i) the instance models describe manufacturing processes and (ii) they serve as input to a simulator.

5.3.1 Running example

To better understand the kind of problems that the conceptual framework and meta-model try to model, this section presents a running example derived from real-world scenarios.

Two types of frames for eyeglasses are manufactured on a production

line. The goal is to produce 100 frames of Lumen Arc, a sleek, modern design with a minimalist plastic frame, and 100 frames of Verde Vintage, a retro-inspired acetate frame with earthy tones. Each frame consists of two stems (left and right) and the frontal lens mount. These components are printed on a press printer. There are three press printers in the production area (PP1, PP2, PP3). The Lumen Arc stems can only be printed on PP1, while Verde Vintage stems can be printed on PP1 and PP2. The frontal frame for both types can be printed with any of the three printers. Each press printer takes 1 minute to print a stem, 2 minutes to print a frontal frame, and requires the corresponding mold, with only one mold available for each piece. This means that two printers cannot print the same item at the same time. The three printers have a start-up time of 10 minutes, and it is preferable to use a printer that is already running. In addition, frontal frame printing has a higher priority because it takes more time. When a component is printed, it is moved in a work-in-progress inventory. There is one for each type of component. An operator manually assembles the final frame taking the right component from the inventory. There are three operators (O1, O2, and O3) who can assemble the frames. O1 takes 1 minute, while O2 and O3 take 2 minutes each.

Once assembled, the frames are placed in a tumbler, of which there is only one in the production area. The frames are tumbled in a machine for 24 hours and the machine has a capacity of 15 frames at a time.

Two operators (O4 and O5) carry out a quality check activity on one item out of every 20 frames, and it takes 2 minutes. If a frame is found to have some defects, it is discarded and has to be reprinted. The frames are then stored in a warehouse, waiting for the next activity.

Every Monday, a lorry arrives at the production site to collect all the frames that are ready for painting. The painting process is done off-site and takes 2 days. Once the frames are back on the production floor, they undergo a final quality check.

In the next sections, frequent references to this running example will be made. In particular, we will refer to:

- Activities: Print right stem, Print left stem, Print frontal frame, Assemble frame, Perform quality check, Paint;
- Executors: PP1, PP2, PP3, O1, O2, O3, O4

5.3.2 Main entities

Activity

Activity represents a fundamental task within a manufacturing process and serves as a natural modeling unit, particularly in light of the activity-centric approach adopted in this work. In the BPMN meta-model, it directly corresponds to the *Activity* entity. In our running example `Print right stem`, `Print left stem`, `Print frontal frame`, `Assemble frame`, `Perform quality check`, and `Paint` are all examples of activities. The inclusion of the *priority* attribute allows for the representation of precedence constraints or preferential ordering among tasks. For example, `Print frontal frame` may be assigned a higher priority than other printing activities, not because of its execution time per se, but because it takes longer (2 minutes instead of 1), and therefore it may be preferable to start it earlier to avoid delays later in the process. To address the need of specifying a priority between activity (described also in the requirement **R7: Activity priority**), we extended the *Activity* entity with an additional attribute, *priority*, enabling users to assign relative importance levels to different activities.

Executor

The *Executor* entity represents a kind of resource engaged in performing an activity within the manufacturing process. In the context of our running example, several types of executor are identified. These include press printers (`PP1`, `PP2`, and `PP3`), human operators (`O1` through `O5`), and the external `Painter service`, all of whom are considered executor instances within the system. When adopting the meta-model with BPMN this entity can be aligned to *Performer* which "[...] defines the resource that will perform or will be responsible for an Activity" [65]. *Executors* are characterized by the attribute *quantity*, specifying their number of available executor with exactly the same characteristics. Finally, executors have parameters (omitted from the diagram to improve readability) that can be queried by the simulation engine to sort or filter candidate executors when assigning an activity instance to an executor instance. This satisfies the requirement **R4: Assignment based on executor's parameters**. Notice that *Executor* represents specific instances. *Groups* or *roles*, often find in similar approaches are captured here by the means of *Compatibility* described below.

Product and ProductFamily

A *Product* is any object that undergoes processing and transformation during the manufacturing process, representing a single item moving throughout the production floor. Being a dynamic element, its life cycle is managed by the simulator.

To enable users to model the variety of products encountered in a manufacturing setting, we introduce the entity *ProductFamily*. While *Product* is a runtime construct and does not have a direct representation in the static meta-model,

ProductFamily serves as its abstract counterpart. In our running example, the *ProductFamily* includes two types of products: Lumen Arc and Verde Vintage.

Accessory

Accessory represents a resource that might be necessary for an executor to effectively execute a task. In our running example, the molds used by the press printers in are accessories. They are necessary items to complete the printing activities, and there is a limited quantity of them. There is exactly one mold for each component type, and this relationship can be formally represented using their *quantity* attribute in the meta-model that captures scenarios where there's a limited number of these resources available to executors.

Inventory

Inventory represents the quantity of products available on which an activity can draw. It has been added to fulfill the requirement **R6: Inventory**, and it can model all those situations where products are stocked somewhere during the production process. In the context of our running example, they are used as buffer between activities. In particular, they have to be explicitly declared when two activities are not directly connected. The printing activities put the results of their work (the left and right stem, and the frontal frame) in a work-in-progress inventory from which the performer of the following activity (`Assemble`) draws from. Moreover, they are of great importance when we want to model a *Transformation* which is described later.

5.3.3 Associations classes between entities

After the definition of the fundamental entities, we moved on defining the associations between these. We decided not to recur to association classes and adopt simpler elements in the meta-model. Consequently, the items described in this subsection are indeed classes, akin to the elements listed above. The distinction between the two lies in the rationale behind their creation: representing basic concepts in the former case and relating basic concepts among them in the latter case.

Compatibility

Compatibility relates *Activity*, *ProductFamily*, and *Executor*. It allows to describe which executor can work on which kind of product in an activity. Moreover, it carries the information about how long the executor takes to perform the activity. From the perspective of an activity, *Compatibility* can be thought of as a matrix.

The running example describes a scenario where the concept of compatibility is deeply used. Starting from the printing activities, we have:

> Type `A` stems can only be printed on `PP1`, while type `B` stems can be printed on both `PP1` and `PP2`. The frontal frame for both types can be printed with any of the three printers.

This scenario can be depicted with three compatibility matrices, one per each activity. Each matrix has as columns the possible executors that can be assigned to that activity, and in the rows the type of products that will be worked on that activity. For now, the cells of the matrix will have a dot, which means that the executor can work on that item. Later we will introduce, also, the concept of time. Tables 5.2c, 5.2b, and 5.2c show the three compatibility matrices for the printing activities.

	PP1	PP2	PP3
Right stem Lumen Arc	•		
Right stem Verde Vintage	•	•	

(a) Compatibility matrix for "Printing right stem"

	PP1	PP2	PP3
Left stem Lumen Arc	•		
Left stem Verde Vintage	•	•	

(b) Compatibility matrix for "Printing left stem"

	PP1	PP2	PP3
Frontal frame Lumen Arc	•	•	•
Frontal frame Verde Vintage	•	•	•

(c) Compatibility matrix for "Printing frontal frame"

Table 5.2: Compatibility matrices for printing activities

Another scenario in which the compatibility relationship is useful is in which executors take different times to perform an activity. In our running example this happens when:

> There are three operators (`O1`, `O2`, and `O3`) who can assemble the frames. `O1` takes 1 minute, while `O2` and `O3` take 2 minutes each.

This scenario can be represented, again, with the compatibility matrix. This time, instead of a simple "can/cannot", we can use the time the executor takes to perform the activity to fill the cell of the matrix. Table 5.3 shows this example. Furthermore, the model supports activities that operate on product batches, which means, that before the activity can take place it to wait a minimum amount of pieces to operate on. *Compatibility* is well suited to represent this kind of information because the batch quantity for an activity depends on the capacity of the executor and the type of products it is working on. For example, the `Tumbling` activity is performed in a tumbler, which is a machine into which

eyeglasses are put along with another material that will smooth them out. This addition to the *Compatibility* entity allows to fulfill the requirement **R2: Batch activities**.

	O1	O2	O3
Lumen Arc	1 min	2 min	2 min
Verde Vintage	1 min	2 min	2 min

Table 5.3: Compatibility matrix with time

Thanks to the introduction of *Compatibility*, the meta-model fulfills the requirements **R8: Relationship between activity, product type, and executor** and **R9: Activity's duration influenced by the assigned executor**.

In addition, the association between *Compatibility* and *Accessory* allows to model those scenarios where an executor need additional resources to carry out the activity. This allows to fulfill the requirement **R5: Multi-executor dependency**.

A *WorkItem* is created when a task is activated, with one or more potential *Executor*s identified (on the basis of corresponding compatibilities) to perform it. When *WorkItem*s are assigned to specific *Executor*s, an *Assignment* instance is created. This operation can follow different policies (this is a well known optimization problem), the most straight forward approach is an eager policy in which unassigned executors constantly check for available compatible *WorkItem*s to perform and assign them to themselves, marking it as unavailable to other compatible ones.

Assignment

Assignment represents the history of assignments, detailing which activities and specific products each executor has worked on. For example, take the scenario described in table 5.2a, the simulator can decide to assign the executor `PP1` to work on a product belongings to the `Right stem A` family for the activity `Print right stem`. This decision is allowed because the compatibility between activity, executor, and product family is defined through the Compatibility entity. In another activity, for example the last quality check, the same product can be assigned to `O4` instead of `O5` due to the fact that it has already been worked by `O4` in the first quality check. The example shows how this association fulfill the requirement **R3: Executor affinity**.

Product transformation

To explain how products are transformed during the process, there are two associations. There is a self-association *ProductFamily*, and an association class between *Activity* and *ProductFamily*. The self-association allows us to describe from which products a product comes and into which products it can transform.

The *Transformation* entity encapsulates the transformation of a product into another product. In the running example, each activity transforms eyeglass components. For instance, the `Assembly` activity consumes one unit each of the right stem, left stem, and frontal frame to produce a complete frame. Given two product types (`A` and `B`), this results in two distinct transformations linked to the `Assembly` activity:

- T_1 consumes one right stem A, one left stem A, and one frontal frame A to produce a final frame A.
- T_2 consumes one right stem B, one left stem B, and one frontal frame B to produce a final frame B.

With this association, we can fulfill the requirement **R1: Product combination/separation**.

5.3.4 Formal definition of the meta-model

To ensure formal rigor and technical operability, the proposed meta-model is defined using the *Ecore* meta-metamodel, which is the de facto implementation of the OMG Meta-Object Facility (MOF) standard. Ecore provides a precise abstract syntax for all modeling elements, their attributes, and relationships, and enables automatic validation, serialization, and code generation. The graphical representation shown in Figure 5.3 is a derived view of this formal definition and is intended solely to support readability. The complete Ecore specification of the meta-model is provided in Appendix A.

5.4 Notation

Having defined the conceptual framework of the domain and introduced the entities and relationships of the meta-model, the next step is to determine how these elements are represented. Representation is not only a matter of notation or format, but a design choice that directly affects the clarity, analyzability, and implementability of the meta-model. A consistent representation allows the elements of the meta-model to be manipulated, compared, and extended in a systematic way, and it provides the basis for subsequent reasoning, tool support, and empirical validation.

In this section, we describe the representation choices adopted for the entities, relationships, and constraints of the meta-model. We outline the principles that guided these choices, explain how different types of elements are encoded, and illustrate the approach with concrete examples. The aim is to make explicit the link between the conceptual definitions of the meta-model and their operational form, thereby bridging the abstract design and its practical realization.

The representation of the meta-model's structural elements was guided by a set of principles intended to balance conceptual clarity with practical applicability. These principles served as evaluation criteria when selecting among alternative

encoding strategies and ensured coherence across different parts of the meta-model. The main principles are summarized below:

- **Faithfulness to the meta-model**: The representation should preserve the semantics of the domain concepts as defined in the meta-model. Essential information must not be lost in the transition from abstract description to formal representation.
- **Compatibility with tooling and standards**: To ensure practical applicability, the representation is aligned with widely adopted modeling standards and notation (i.e., BPMN) wherever possible. This facilitates integration with existing modeling tools and lowers the entry barrier for practitioners.
- **Usability and simplicity**: The representation should remain accessible and avoid unnecessary complexity, while still being expressive enough to capture the required concepts. A minimal set of constructs is preferred over the introduction of specialized symbols for every element. For instance, tabular data are represented directly using tables rather than dedicated ideograms.

5.4.1 Elements with a representation in the diagram

Following the principles described above, we present the graphical representation adopted for each entity of the meta-model that appears in the diagrams. To keep the notation lightweight and readable, we favor minimal symbols on the canvas and defer operational details (e.g., capacity and assignment constraints) to a sections introduced later.

Activity

An *Activity* denotes a single, indivisible unit of work; in our notation we restrict activities to atomic tasks only. Visually, we reuse the established process-modeling convention: a rounded rectangle containing the activity's name, as illustrated in Figure 5.4. Each activity is intended to transform inputs into outputs without exposing any internal decomposition or sub-flow. Names should be concise and action-oriented so that the purpose of the task is immediately apparent within the control flow. Any further execution conditions, resource considerations, or eligibility rules are not shown on the activity itself; they are captured outside the diagram to preserve clarity.

Executors

Executors represent the entities that can perform activities: humans, machines, services, or organizational units. We depict them as hexagons labeled with the executor's name (Figure 5.5), a distinct shape that separates them from process flow nodes. The diagram expresses only the presence of executors and their

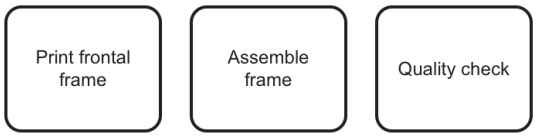

Figure 5.4: Example of activities.

potential to carry out work; it does not convey capacity, shifts, or skill levels. Those aspects are handled by the *Compatibility* concept described later, ensuring that the canvas stays focused on structure while operational constraints remain formally specified but off-diagram.

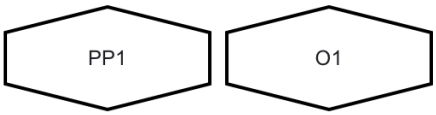

Figure 5.5: Example of executors.

Relationship between Activities and Executors

The connection between an activity and an executor is shown with a non-oriented dashed line (Figure 5.6). This link simply states that the executor is able to perform the activity; it is not an assignment, it carries no multiplicity, and it bears no policy annotations. In particular, requirements such as "an operators need an accessory" or 'a skilled technician takes 5 minutes" are not encoded on this edge. Instead, those rules are expressed in the *Compatibility* specification, which governs feasibility and selection at analysis or execution time while keeping the diagram uncluttered.

Inventory

An *Inventory* denotes a repository for materials or information artifacts that activities may consume or produce. We adopt the conventional database (cylinder) symbol with the inventory's name adjacent the icon, as shown in Figure 5.7. Inventories in the diagram are intentionally lightweight. They do not carry stocking policies, reorder parameters, or capacity limits. Their role is to make explicit the presence of a store and to anchor the logical flow of items without introducing operational details that would distract from the structural view.

Relationships between Activities and Inventories

The movement of materials or information between activities and inventories is indicated with directed dashed arrows (Figure 5.8). An arrow from an activity

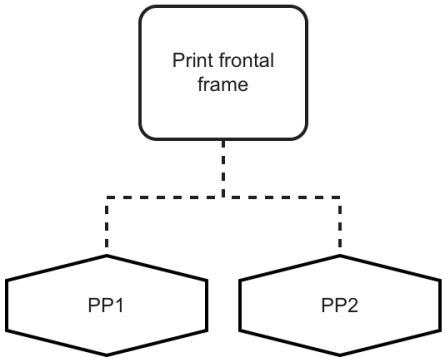

Figure 5.6: Example of a relationship between an activity and two executors.

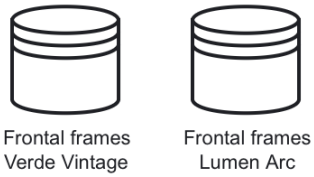

Figure 5.7: Example of inventories.

to an inventory signifies that the activity produces items that are subsequently stored there; conversely, an arrow from an inventory to an activity indicates that the activity draws required items from that store. To maintain readability, these edges remain free of quantities, units, conditions, or batch semantics. Where analyses require such parameters, they are specified outside the diagram, allowing the graphical notation to emphasize directionality and provenance without visual clutter.

5.4.2 Elements without a representation in the diagram

Some model elements are more naturally captured as parameterized tables than as nodes or edges. They tend to be high-cardinality, instance-level, or policy-oriented, and would clutter the canvas if shown graphically. For these, we rely on forms with explicit fields, validation, and import/export so that the diagram remains lightweight while the specification stays executable.

Compatibility

Compatibility expresses, in a precise yet readable way, who can perform what under which conditions. Rather than annotating edges, the modeler completes a *Compatibility* form that binds an *Activity* to an *Executor*, optionally narrowing

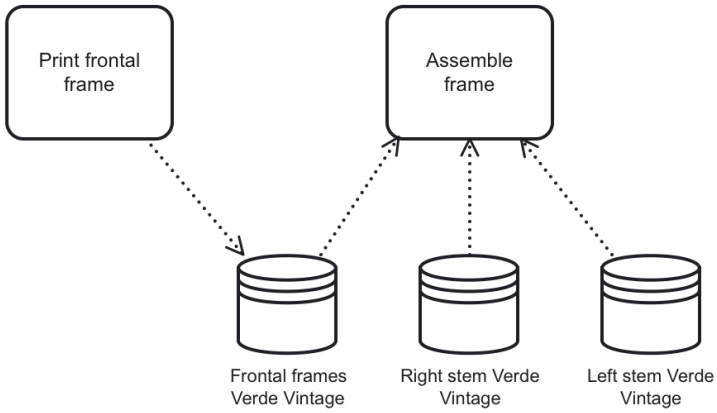

Figure 5.8: Example of a relationship between activities and inventories.

the scope to a *ProductFamily*. The form records the expected *Duration*, the *BatchSize*, and any required *Accessories* listed as `name:quantity` pairs. Read as a sentence, an entry states: "Executor E can perform Activity A (optionally for Product Family P) in d time per batch of size b, using {accessories}." Omitting the product family intentionally lifts the restriction so that the entry applies to all products. Capacity and assignment policies, which are not shown on the diagram, are realized operationally through these compatibility records, allowing dispatch, scheduling, and feasibility checks to reason over executors without adding visual noise.

Transformation

A transformation captures how an activity consumes from and produces into inventories. In the *Transformation* form, the modeler selects an *Activity* and then specifies its *Inputs*: the products and quantities to be withdrawn from designated source inventories, and its *Outputs*: the products and quantities to be deposited into target inventories. Optional attribute updates for the affected products can be included to reflect changes in state, grade, revision, or metadata. Conceptually, the completed form reads like a recipe that the simulator executes atomically: "When A completes, decrement these inputs here, increment those outputs there, and update attributes as specified." By centralizing I/O semantics in a table, we keep the graphical notation free of quantities, units, or batch mechanics while preserving exact execution behavior for analysis.

Accessories

Accessories enumerate auxiliary tools and fixtures as a small catalog. Through the *Accessories* form, each row declares a *Name* and a *QuantityAvailable*. Compatibility entries then reference these catalog items by name, enabling the

dispatch logic to check availability according to the chosen allocation policy. This indirection supports reuse across many activities and executors without duplicating counts or constraints on the diagram.

Validation, defaults, and reuse. All forms enforce schema-level checks such as referential integrity (activities, executors, inventories, and accessories must exist), non-negativity for durations, quantities, and batch sizes, and uniqueness where applicable (e.g., no duplicate compatibility entries for the same (A, E, P) tuple). Sensible defaults are provided, for instance, product-family–agnostic compatibility when the field is left blank, and forms can be imported, exported, and versioned to align with operational data. This approach keeps diagrams uncluttered while retaining the precision needed for simulation, scheduling, and execution.

5.4.3 Model of the running example

Building on the running example introduced in 5.3.1, we can now model the process using the proposed representation.

The basic BPMN process model is presented in Figure 5.9. It contains several standard BPMN elements such as activities, start and end events, parallel gateways, exclusive OR gateways, and time events.

We then enrich the model with executors and inventories. Executors are straightforward to derive from the running example: PP1, PP2, PP3, O1, O2, O3, O4, and O5. In addition, we define executors for painting (Painter) and for discarding defective frames (O6). Inventories are introduced only where products must be stored for later use. In this case, inventories are required between the production of the three components and the assembly activity. The updated diagram is shown in Figure 5.10.

To make explicit what the BPMN diagram leaves implicit, we interleave it with three structured forms that the reader can consult alongside the model. First, the *compatibility* form (Figure 5.11) constrains which printers, molds, and parts may be combined during printing.

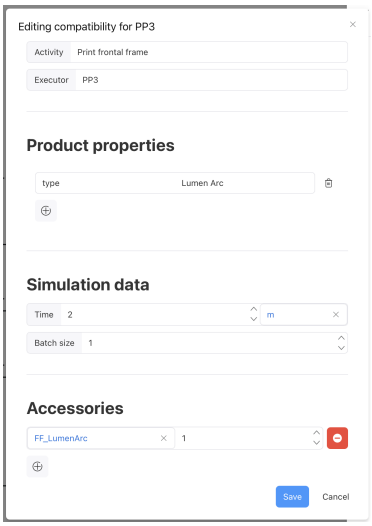

Figure 5.11: Modeling of a compatibility

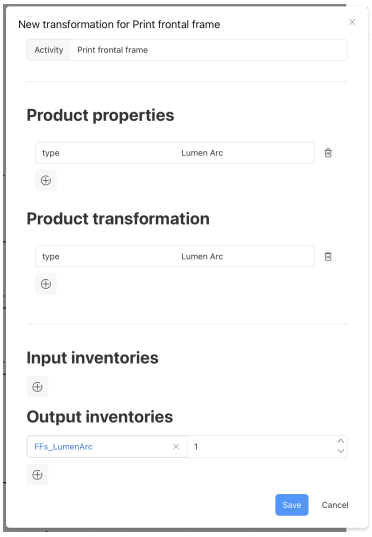

Figure 5.12: Modeling of a transformation

Next, the *transformation* form (Figure 5.12) specifies how items progress between states as activities complete (e.g., how to print frontal frames).

Finally, the *accessories* form (Figure 5.13) enumerates auxiliary materials and their associations needed at assembly time.

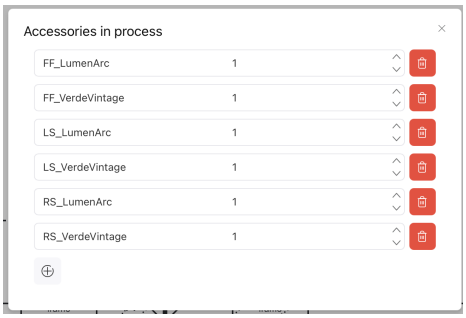

Figure 5.13: Modeling of the list of accessories

Taken together, these forms operationalize the rules that govern the process while the diagram conveys the control and flow structure.

In summary, the representation of the meta-model elements was defined according to a set of guiding principles that balance faithfulness, usability, and compatibility with existing standards. Graphical notations were introduced for the core entities and relationships to ensure clarity and ease of interpretation, while forms were used to capture complementary information not directly conveyed in diagrams. The running example illustrated how these choices work together to provide a coherent and systematic representation of processes, executors, inventories, and their interactions. This representation serves as the foundation for the implementation of the modeler and dashboard developed in the subsequent chapters.

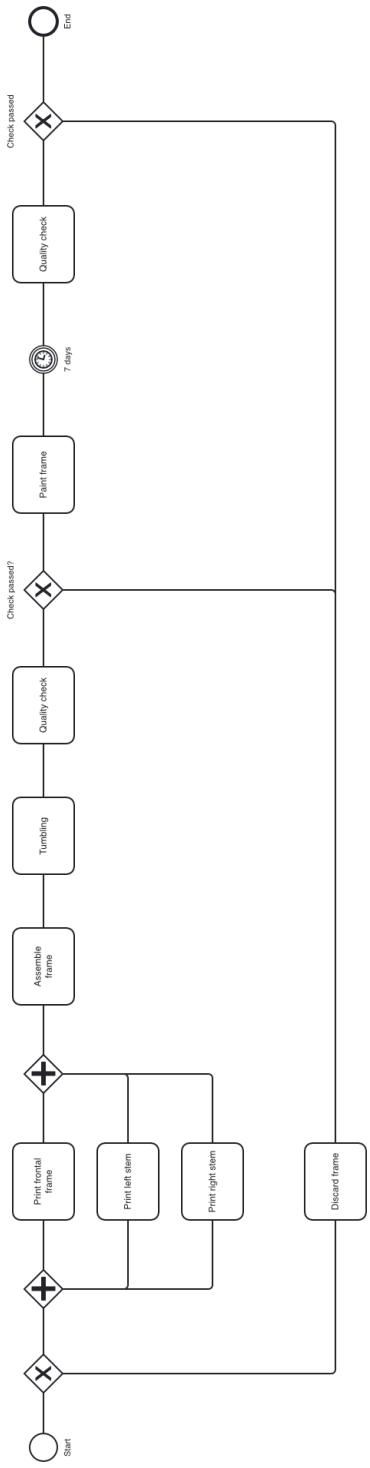

Figure 5.9: Running example, process model only

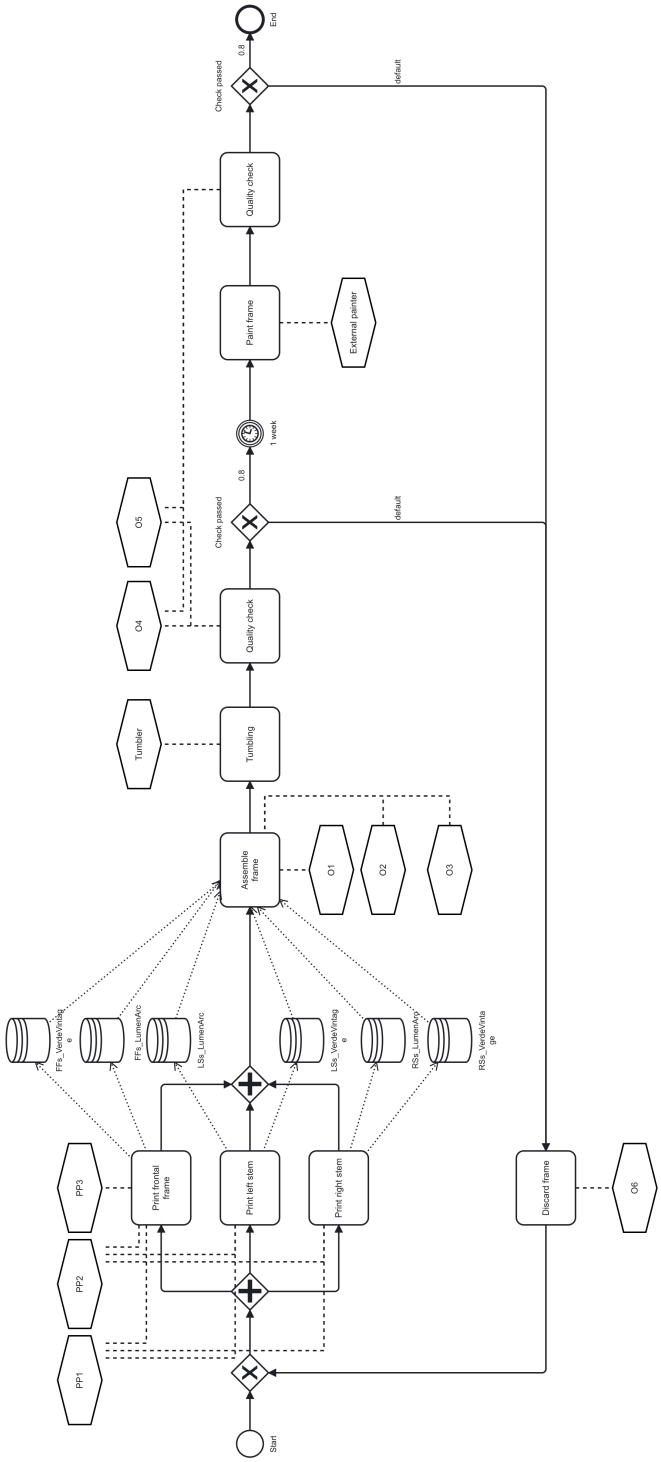

Figure 5.10: Running example, process model with executors and inventories

Chapter 6

ARMS: Modeler, Simulation Engine, and Dashboard

This chapter presents the software system that operationalizes the modeling and simulation approach introduced earlier. The system, called ARMS [25], is composed of three tightly coupled components: a BPMN-based modeler that lets users author simulation-ready process models, a discrete-event simulation engine that executes those models, and a dashboard that surfaces results for analysis and decision support. The three parts are designed to work as a single loop: models are authored and validated, executed to produce measurements, and then inspected to guide iterative refinements to structure, policies, and capacity.

We begin by describing the modeler, which extends standard BPMN with lightweight, simulation-oriented constructs (e.g., *Executor*, *Inventory*, *Compatibility*, *Transformation*). The extensions preserve the familiarity of BPMN while making resource interactions first-class and enforceable through validation rules. A disciplined serialization strategy embeds these additions into the `.bpmn` file so that models remain portable, auditable, and tool-friendly.

Next, we detail the simulation engine, a discrete-event core that instantiates models into an executable representation inspired by Petri nets. The engine de-serializes extended BPMN, performs semantic checks, and drives execution by synchronizing control and product flows, allocating executors and accessories, and applying transformations. Its outputs consist of structured metrics: utilization, waits, throughput, and queue dynamics sufficient to support capacity sizing, batching policies, and targeted performance improvements.

Finally, we introduce the dashboard, which closes the loop by mapping engine outputs back onto the model. Global summaries contrast simulated and ideal times to expose delay sources; local views attribute work and delays to specific activities and executors; and a heatmap overlays availability on the diagram to reveal bottlenecks at a glance. The visualization layer is extensible: additional performance or sustainability indicators (e.g., energy, waste, maintenance costs) can be introduced without altering the core simulation semantics.

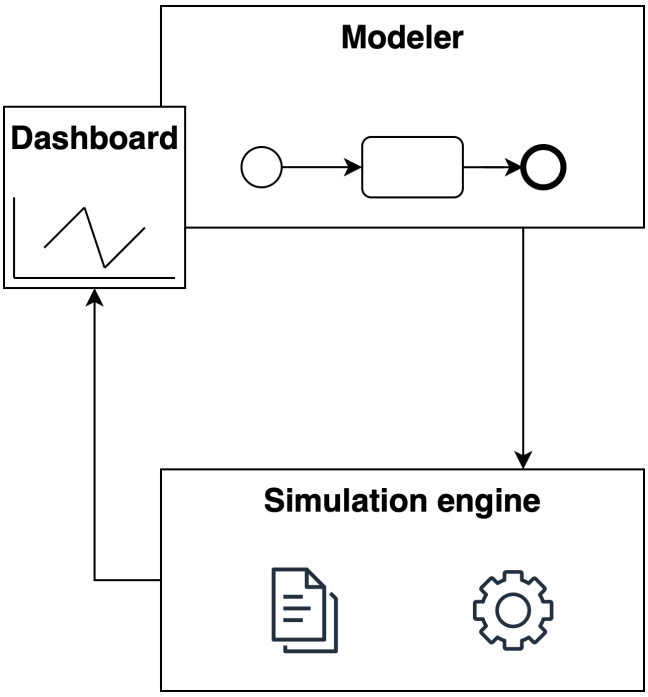

Figure 6.1: ARMS' modeler, simulation engine, and dashboard

6.1 Modeler

The modeler constitutes the primary interface through which users define, visualize, and configure the processes to be simulated. Its main purpose is to extend standard BPMN modeling capabilities to include the resource-related concepts introduced by the meta-model described in Section 5.3. Through these extensions, users can explicitly represent the interaction between processes, resources, and materials, bridging the gap between business process modeling and manufacturing simulation.

6.1.1 Design Rationale

From the earliest stages of design, the modeler was conceived with a clear non-functional objective: accessibility. The target users of this tool are domain experts in manufacturing, engineers, process analysts, and consultants, who often possess deep process knowledge but limited programming or IT skills. Therefore, the modeler had to provide an intuitive, installation-free interface that could be accessed from any standard web browser, without requiring configuration, dependencies, or administrative privileges.

This requirement strongly influenced the choice of implementation technologies, as discussed in Section 4.1.4. To guarantee accessibility, agility, and maintainability, a web-based architecture was adopted. Among the wide range of available frameworks, two complementary technologies were selected to fulfill the system's functional and non-functional needs: React and BPMN.js. React provides the general-purpose, modular foundation for building interactive user interfaces, while BPMN.js offers the domain-specific infrastructure for BPMN diagram creation and manipulation. Together, they form the technological backbone of a modeling environment that is both expressive and usable.

6.1.2 Technologies Adopted

React

The entire front-end was implemented as a React[1] application. React's component-based architecture and declarative rendering model enable the development of complex interactive interfaces while maintaining modularity and readability. Its virtual DOM and reactive state management ensure high responsiveness even in diagram-heavy views, where multiple interface components must synchronize in real time.

This choice directly supports the non-functional requirements of *accessibility* and *agility*: the modeler runs entirely in the browser, independent of the underlying operating system or hardware, and requires no installation. Furthermore, React's extensive ecosystem of libraries facilitated rapid development and seamless integration with domain-specific components such as BPMN.js and Recharts. As a result, the modeler achieves both user accessibility and developer

[1] https://react.dev

maintainability, two essential criteria for research software intended to evolve over time.

BPMN.js

To support BPMN diagram editing, the modeler integrates `bpmn-js`[2], an open-source JavaScript library that provides a complete BPMN 2.0 rendering and editing toolkit. This library supplies a customizable canvas, palette, and properties panel, together with a well-documented API for extending the BPMN metamodel and its graphical representation.

By adopting BPMN.js, the project inherited a mature and widely validated implementation of the BPMN standard, thereby ensuring both *notation correctness* and *modeling consistency*. Rather than reimplementing a graphical editor from scratch, a task that would have demanded substantial effort and risked compromising semantic fidelity, development has instead focused on extending BPMN with the new constructs required by the resource-aware meta-model.

6.1.3 Implementation

The final implementation of the modeler[3] is a fully web-based application built on the two technologies described above: React and bpmn-js. An overview of the interface is shown in Figure 6.2. The system architecture was designed to preserve the intuitive modeling experience of BPMN while extending its expressiveness to encompass resource dynamics and simulation-related attributes.

Building on `bpmn-js`, the modeler introduces several domain-specific extensions and enhancements:

- **BPMN extension:** New modeling elements, such as *executors*, *inventories*, *compatibilities*, and *transformations* were added to capture the resource perspective of manufacturing processes. These extensions allow users to specify how activities consume, produce, or depend on resources, enriching BPMN with a resource-aware semantics.
- **Rendering mechanism:** Custom visualization logic integrates resource information directly into the process diagram. Executors, inventories, and their interconnections are rendered alongside standard BPMN elements, ensuring a unified visual representation.
- **Validation rules:** A dedicated linter validates model consistency by checking the presence and correct linkage of the new entities. For instance, it warns users when an activity lacks an associated executor or when required configuration parameters are missing for simulation.
- **Configuration widgets:** Complementary interface components allow users to define compatibilities between executors and activities, specify

[2] https://bpmn.io/toolkit/bpmn-js/
[3] https://github.com/jjocram/ARMS-Editor

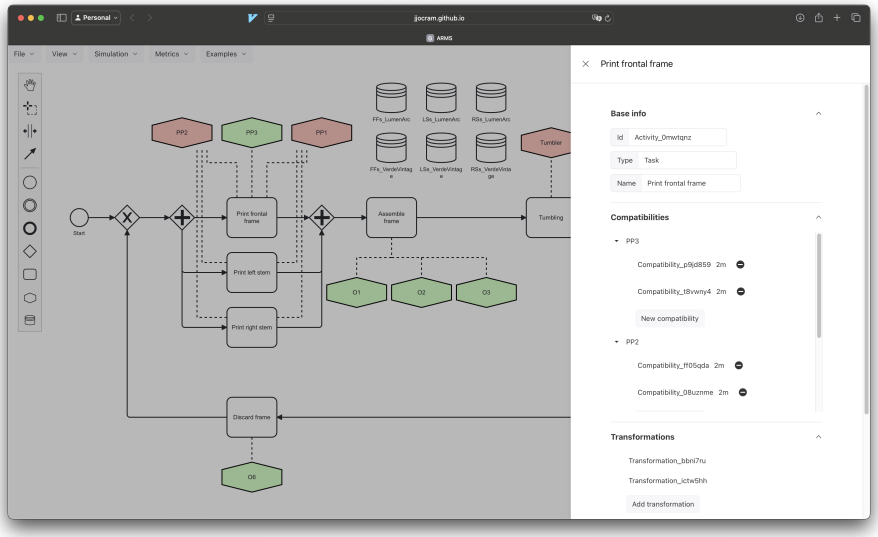

Figure 6.2: Screenshot of the modeler

product transformations, assign accessories, set activity priorities, and indicate production requests. These widgets abstract complex data relationships into user-friendly forms.

Through these extensions, the modeler transcends the descriptive capabilities of standard BPMN, enabling the creation of models that are both semantically valid and *simulation-ready*. The combination of React and BPMN.js proved instrumental in achieving this balance: React ensures fluid interaction and maintainable component logic, while BPMN.js guarantees adherence to BPMN standards and provides a solid foundation for graphical extensibility. Together, they realize a modeling environment that embodies the key non-functional goals of this research, *efficiency, accessibility, and reliability*.

6.1.4 Model serialization

The system leverages BPMN's extensible file format to serialize the process model information. BPMN-js provides an easy way to implement this extension mechanism by adding data to the `.bpmn` file representing the model within the `<bpmn:process>` tag for elements appearing in the diagram and `<bpmn:extensionElement>` for elements not depicted in the diagram, such as compatibilities, transformations, and accessories. Specifically, six new elements are introduced:

- *Executor*: contains all the data belonging to the executor such has a name, quantity, energy consumption, cost per unit of time, waste generation, and maintenance cost.

```
<factory:executor
    id="ExecutorActivity_1wiacyp"
    name="PP1"
    quantity="1"
    cost="0"
    energyConsumption="0"
    wasteGeneration="0"
    maintenanceCost="0">
      <bpmn:incoming>Flow_01vz5bn</bpmn:incoming>
      <bpmn:incoming>Flow_16j7t77</bpmn:incoming>
      <bpmn:incoming>Flow_0sd6qg4</bpmn:incoming>
</factory:executor>
```

- *Connection*: an undirected line between an executor and an activity. it will be represented in the diagram with a dashed line specifying that at least one compatibility exists between the two elements;
- *Compatibility*: represents a compatibility between an activity, an executor, and a product. It specifies the duration, which accessories it needs, and the batch size.

```
<factory:compatibility
    id="Compatibility_h0dhhck"
    time="2"
    timeUnit="m"
    batch="1"
    idActivity="Activity_0mwtqnz"
    idExecutor="ExecutorActivity_1wiacyp">
        <factory:productProperty
            key="type"
            value="Lumen Arc" />
        <factory:accessoryCompatibility
            id="Accessory_9rbyt4i"
            quantity="1" />
</factory:compatibility>
```

- *Inventory*: represents a specific type of product. It is characterized by a name and a the quantity of items inside at the begging of the simulation.

```
<factory:inventory
    id="Inventory_1vb8c3l"
    name="FFs_LumenArc"
```

```
    startQuantity="0"
/>
```

- *Transformation*: it is associated with an activity and specified the product properties is applies to, the updated product properties after the activity completion, the required input products and the resulting output products;

```
<factory:transformation
    id="Transformation_ictw5hh"
    activityId="Activity_0mwtqnz">
        <factory:transformationProductProperty
            key="type"
            value="Verde Vintage" />
        <factory:transformationProductPropertyToApply
            key="type"
            value="Verde Vintage" />
        <factory:transformationOutput
            id="Inventory_0mxqq7o"
            inventoryId="Inventory_0mxqq7o"
            quantity="1" />
      </factory:transformation>
```

- *ProductRequest*: represent which products we want to produce during the process;

```
<factory:productRequest
    id="ProductRequest_xugxeq2"
    quantity="100">
        <factory:productProperty
            key="type"
            value="Lumen Arc" />
</factory:productRequest>
```

- *Accessory*: represents which and how many accessories are available in the process.

```
<factory:accessory
    id="Accessory_9rbyt4i"
    name="FF_LumenArc"
```

```
    quantity="1"
/>
```

6.2 Simulation engine

The simulation engine[4] represents the computational core of the ARMS platform. It transforms the extended BPMN models produced by the modeler into executable simulations, coordinating process logic, resource behavior, and product flow over simulated time. Conceptually, the engine bridges the descriptive layer of BPMN with the operational semantics required for discrete-event simulation (DES).

6.2.1 Design Rationale and Technological Foundations

The design of the simulation engine was shaped by three primary non-functional objectives: efficiency, correctness, and interoperability. From an efficiency standpoint, the simulator had to handle industrial-scale models without incurring bottlenecks typical of interpreted runtimes. For correctness, it was essential to rely on a proven DES kernel rather than reimplementing scheduling and event management from scratch. Finally, interoperability was required to expose simulation services to the web-based modeler through lightweight APIs, enabling seamless end-to-end interaction between modeling and execution.

These considerations (described in Section 4.1.4) motivated the adoption of three complementary technologies:

- **Kotlin**, as the core implementation language, providing modern expressiveness and high runtime efficiency.
- **Kalasim**, as the underlying discrete-event simulation framework, ensuring reliability and temporal correctness.
- **Ktor**, as the asynchronous HTTP façade enabling communication with the web-based front end.

Together, these components form a cohesive, scalable, and maintainable architecture that supports high-performance simulation through a modern, coroutine-driven programming model.

Kotlin

Kotlin was selected as the foundation of the simulation engine due to its balance between performance, safety, and interoperability. As a statically typed language compiling to JVM bytecode, it avoids the runtime overheads of interpreted

[4] https://github.com/jjocram/ARMS-Simulation-Engine

languages while retaining near-native efficiency. Its concise syntax, null-safety, and structured concurrency via coroutines enable us to manage thousands of concurrent simulation entities without blocking I/O or user requests. Kotlin's full compatibility with the Java ecosystem further streamlines XML parsing, serialization, and scripting for model import and execution.

Kalasim

The core simulation logic builds upon Kalasim, a process-oriented DES framework in Kotlin. Kalasim provides a robust event scheduler and a coroutine-first API, allowing simulation components to behave as lightweight sequential processes. By adopting this framework, we inherited validated mechanisms for event management and monitoring, focusing our effort on domain semantics rather than rebuilding a simulation kernel.

Ktor

The engine is exposed as a standalone web service via Ktor. Ktor's coroutine-based, non-blocking model aligns naturally with Kotlin and Kalasim, supporting concurrent requests while simulations are running. It serves as the bridge to the React modeler: models are submitted as JSON, simulations are triggered remotely, and results are returned through REST endpoints fulfilling the accessibility requirement of running entirely from the browser.

6.2.2 Alternatives

We conducted a comparative evaluation of alternative technologies and frameworks against the non-functional requirements defined in Section 4.1.4. Although technically sound in isolation, each alternative conflicted with one or more of our priorities (efficiency, modernity, correctness-through-reuse, accessibility, and architectural control):

SyDEVS. SyDEVS offers a disciplined DEVS-based [90] approach and strong performance characteristics. However, we discarded it primarily due to the *language and ecosystem trade-off*: C++ complicates developer productivity and maintainability for a research codebase intended to evolve quickly, and it does not integrate as naturally with our web-first architecture. Moreover, DEVS's modeling idioms would have required additional adaptation layers to realize our BPMN- and resource-centric semantics, reducing agility.

Salabim and SimPy. Both are mature, widely used Python simulation libraries. Nevertheless, we rejected a Python-based stack because it directly contradicts our efficiency requirement: interpreter overhead and concurrency constraints (e.g., GIL in common configurations) introduce scalability risks for larger models and multi-user scenarios. Following the requirements, we favored

a compiled, statically typed language with structured concurrency and JVM interoperability.

It is worth noting that Salabim[39] is the root from which Kalasim is derived. For this reason they share many concepts.

Scylla. Scylla's extensibility [72] is appealing; however, our work requires *tight control over BPMN execution semantics* to embed resource-awareness and Petri-net-inspired synchronization directly into the core. Relying on third-party extension points would have constrained key technical decisions and imposed coupling to external release cycles. For these reasons, we preferred an in-house implementation of BPMN execution atop Kalasim, ensuring end-to-end semantic coherence with our meta-model.

6.2.3 Simulation Formalism

Internally, processes follow a Petri-net-inspired formalism with *places* and *transitions*. Places hold *Token* instances representing dynamic entities, while transitions specify conditions for token movement. This provides a rigorous yet flexible structure for capturing both control flow and resource behavior. Control and product flows are synchronized via paired *ControlTokens* and *ProductTokens* sharing the same identifier, effectively embedding a colored-Petri-net interpretation within BPMN semantics.

6.2.4 Model de-serialization

Before executing a simulation, the engine must first construct the process model from the extended BPMN input file. The model is provided using BPMN's standard XMI XML encoding format and may include ARMS-specific extensions such as inventories, compatibilities, and product requests. The file is parsed and normalized into strongly typed data transfer objects (DTOs), with custom namespaces (e.g., `factory:compatibility`, `factory:transformation`) translated into Kotlin structures.

The parser instantiates the corresponding objects for each process element and links them according to their identifiers, thereby reconstructing the overall structure of the process. Sequence flows are converted into executable routing predicates, currently supporting both probabilistic values and scripted conditions via the Kotlin JSR223 engine. During this phase, the engine also performs syntactic and semantic checks to ensure that all references are well-formed and that no inconsistencies exist in the model. If errors are detected, execution is halted and descriptive feedback is returned to the modeler.

6.2.5 Entities

Each BPMN element and metamodel entity is implemented as a distinct class, instantiated during model parsing. Their behavior is defined in terms of predefined execution semantics expressed through places and transitions. In addition to

standard BPMN constructs, several auxiliary entities extend the expressiveness of the engine to capture product, resource, and execution dynamics.

The most relevant entities are:

- *Token*: The fundamental unit moved across the system. Three types are distinguished:
  - *ControlToken*: Represents the control flow of the process.
  - *ProductToken*: Represents the movement and transformation of products, with support for key-value property maps and executor-affinity tags to enforce routing constraints.
  - *ResourceToken*: Encodes limited resources such as inventory items or accessories.
- *Place*: A container that holds tokens, with UUID-aware extraction ensuring deterministic batching while preserving token identity across flows.
- *Transition*: Connects input and output places, enforcing conditions that govern firing. Transitions synchronize control and product tokens carrying the same identifier.
- *ProductRequest*: Defines a set of *ProductFamilies* to be produced during a simulation run.

Beyond these core entities, the following metamodel-based components govern simulation dynamics more directly:

- *Compatibility*: Defines the mapping between an activity and a specific executor, together with product selection criteria.
- *WorkItem*: Represents an executable task queued for an executor, including duration, required accessories, input/output products, and consumable resources.
- *Executor*: Implemented as Kalasim components with individual job queues, batching policies, accessory requirements, and service-time semantics. Executors consume input resources, hold for task duration, and produce specified outputs.
- *Transformation*: Encapsulates material balance logic, consuming resources and producing outputs during execution.
- *Accessory*: Auxiliary resources represented as pools of tokens, which must be acquired by executors before task execution.
- *ProductFamily*: A flexible key-value representation of product types.
- *Product*: An instantiation of a product token adhering to its *ProductFamily*.

- *Inventory*: A dedicated place for managing material stocks, supporting retrieval and replenishment operations.
- *Assignment*: Records executor-task mappings within product tokens, enabling affinity tracking and downstream routing constraints.

6.2.6 Simulation execution

The execution of a simulation begins at the BPMN *StartEvent*, where initial *ControlTokens* and *ProductTokens* are generated according to defined *ProductRequests*. Tokens are then propagated through the process model. Each component independently evaluates its firing conditions, reacting instantaneously to events in the Kalasim queue.

The semantics of BPMN elements are modeled as follows:

- *StartEvent*: Instantiates new tokens at time zero.
- *EndEvent*: Collects arriving tokens and records results.
- *Parallel split gateway*: Duplicates tokens and forwards them to all outbound places.
- *Parallel join gateway*: Synchronizes tokens with matching identifiers from multiple inbound places.
- *Exclusive OR split gateway*: Delegates guard evaluation to scripted conditions (defaulting to probabilistic rules).
- *Exclusive OR join gateway*: Triggers when any inbound transition fires.
- *Activity*: Matches control and product tokens, identifies compatible executors based on product attributes and compatibility rules, and creates corresponding work items.

Each executor monitors its queue and, upon availability of sufficient compatible work items, begins execution. Execution involves acquiring required accessories, consuming input tokens per transformation rules, advancing the simulation clock, and producing outputs. Upon completion, resources are released and tokens are forwarded downstream.

The simulation run terminates once all requested products reach terminal places or no further actions can be triggered. A watcher component ensures goal-driven termination, as opposed to fixed time horizons.

6.2.7 Metrics and reporting

During execution, the engine records detailed performance metrics. These include queueing delays, processing times, and executor-level statistics such as throughput, utilization, and queue lengths. Metrics are collected using

TimeDeltaMetric for duration tracking and snapshot-based collectors for work-in-progress estimates. Extension properties consolidate metrics into executor reports, which are returned in JSON format.

The web service exposes an endpoint (`/simulate`) that accepts BPMN models, executes them in an isolated simulation environment, and returns results. The output (an example of which is shown below) includes executor diagnostics, simulation horizon, and resource utilization summaries. These results are consumed by the companion web-based BPMN modeler for visualization and analysis.


```
{
  "simulation": {
    "totalTime": 50406
  },
  "executors": [
    {
      "id": "ExecutorActivity_1wiacyp-0",
      "maxWaitTimeInQueue": 399,
      "avgWaitTimeInQueue": 141.80248772869254,
      "sumWaitTimeInQueue": 69876,
      "busy": 582.0,
      "idle": 49817.083333333336,
      "processedItems": 482,
      "avgQueueLength": 1.55,
      "varQueueLength": 1.8199502829E8,
      "stdQueueLength": 18.36,
      "maxQueueLength": 299,
      "activities": [
        {
          "id": "Activity_0mwtqnz",
          "maxWaitTimeInQueue": 198,
          "avgWaitTimeInQueue": 99.0,
          "sumWaitTimeInQueue": 9900,
          "busy": 200,
          "processedItems": 100
        },
        {
          "id": "Activity_1ouy2bw",
          "maxWaitTimeInQueue": 299,
          "avgWaitTimeInQueue": 115.84722222222223,
          "sumWaitTimeInQueue": 25023,
          "busy": 216,
          "processedItems": 216
        },
        ...
```

```
      ]
    },
    ...
  ]
}
```

The metrics are not merely descriptive; they are instruments for tactical decision making. The JSON report tells us how long work waits, how fast it is processed, how busy each executor is, and how queues evolve over time. With that, we can right-size the number of executors, tune batch sizes, and justify investments that shorten activity durations. These are just an example of what kind of tactical decision can be made starting from the results of the simulation.

In what follows, we give some possible actions that can be derived from the results of the simulation.

Right-sizing executors (scale up / scale down)

Each executor report includes time spent busy versus idle, items processed, and queue statistics (average and maximum lengths, average and maximum wait). Together they indicate whether we are chronically under-provisioned, comfortably balanced, or over-provisioned.

- **When to add executors.** If users or jobs regularly breach the agreed waiting-time target (for example, the 95th percentile wait exceeds the Service Level Agreement (SLA) for several consecutive reporting windows), and queue lengths trend upward, add capacity. This is especially clear when executors are busy most of the time and queues still grow.
- **When to remove executors.** If queues are consistently near zero and executors spend long stretches idle, we are paying for capacity we do not need. In that case, remove a unit of capacity and observe the next few windows to confirm SLAs remain green.
- **Avoiding oscillations.** Use hysteresis (the dependence of the state of a system on its history): require stronger evidence to reverse a recent scaling decision, and impose a minimum "cooldown" between adjustments. This prevents the system from see-sawing in response to short bursts.
- **Handling bursts vs. chronic load.** Very high maximum waits or spikes in maximum queue length, coupled with modest average wait, suggest bursty arrivals rather than chronic under-capacity. Prefer temporary or elastic executors that come online only when queues cross a high-watermark, rather than permanently increasing headcount.

Batch-size tuning

Some activities process items in batches, trading off efficiency against additional waiting while a batch fills.

- **Increase batch size** when setup or coordination overhead dominates and queues are usually small. The report will show frequent short busy bursts separated by idle time; larger batches amortize the overhead and increase throughput.
- **Decrease batch size** when the next activity waits too long for batches to fill. Here we see large maximum waits and rising average queue length even though executors are not fully utilized. Smaller batches (or "dispatch early if the oldest item is getting old") reduce tail latency.
- **Use dynamic batching.** Tie the batch size to current congestion: allow larger batches when queues are long and shrink them when queues are short. This is simple to implement and tracks the operating point without a full reconfiguration.

Investing in shorter activity durations

Sometimes the most effective lever is to make executors work faster rather than add more executors.

- **Prioritize true bottlenecks.** Look for activities with high busy time, consistently high utilization, large accumulated waiting time, and queues that never fully drain. Small reductions in their service time often yield outsized reductions in end-to-end delay.
- **Estimate the payoff.** Run a pair of what-if scenaris: baseline and "improved executor" (e.g., 10% faster). Compare SLA compliance, total waiting time, and throughput. If a modest speedup on one activity improves system-wide waiting more than adding an executor elsewhere, the investment has a strong case.
- **Spend where it matters.** Favor improvements on stages that are both heavily utilized and feed many downstream paths; avoid spending on stages that are rarely busy or already hidden behind upstream bottlenecks.

Additional levers enabled by the same metrics

- **Backpressure and admission control.** If an upstream queue swells while a downstream pool is lightly loaded, allow more release from upstream. If downstream queues are swelling, throttle upstream releases to protect tail latency.
- **Prioritization.** If SLA-critical classes show worse tail waits than non-critical work, switch from first-come-first-served to a priority rule or shortest-processing-time at the affected executor.
- **Load balancing.** If executors in the same pool show uneven busy times or waits, adjust routing so arrivals prefer the less-loaded executors until utilization evens out.

- **Timeouts and retries.** Large variability in queue length with occasional spikes suggests transient stalls. Adding jitter to retries or smoothing bursty submissions can reduce these spikes without increasing capacity.

6.3 Dashboard

The dashboard constitutes the main interface through which simulation results are presented to the user. It is closely integrated with the modeler, adapting its behavior once simulation outputs become available. The dashboard supports three distinct modes of visualization:

1. **Global results:** When no element of the process diagram is selected, the dashboard displays aggregate outcomes for the entire process execution in a dedicated widget.

2. **Local results:** When a specific element in the process diagram is selected, the widget shows the detailed results corresponding to that element.

3. **Heatmap:** Once the simulation results are ready, each executor in the diagram is automatically color-coded according to its availability during the simulation.

For global results, the dashboard provides a bar chart that compares the total simulated execution time of the process with its theoretical duration. The theoretical, or *ideal*, time is calculated by summing, for each product request and activity, the processing time required under the assumption of no waiting. Consequently, any discrepancy between the simulated and ideal values highlights the presence of delays, which arise when executors are not immediately available to process incoming jobs.

For local results, the information displayed depends on the type of element selected. When an activity is chosen, two aspects of its performance are presented. The first is the proportion of working time contributed by each executor associated with that activity. The second is the distribution of the number of products processed by those executors. Both measures are visualized using pie charts, enabling a clear comparison of workload allocation among executors.

Figure 6.3 illustrates this functionality with the activity *"Print frontal frame"*, which is executed by three different executors. The pie chart highlights how the workload is distributed across them.

When an executor is selected, the dashboard provides two complementary types of performance data.

The first concerns the distribution of the executor's workload across the activities to which it is assigned. This is visualized through a pie chart that shows the proportion of total working time dedicated to each activity.

The second type of information focuses on the processing times experienced by jobs within the executor. Three measures are reported: the *ideal*, *average*, and *worst* time. The *ideal* time corresponds to the pure processing time, that is, the time required for an item to be completed assuming an empty queue. The

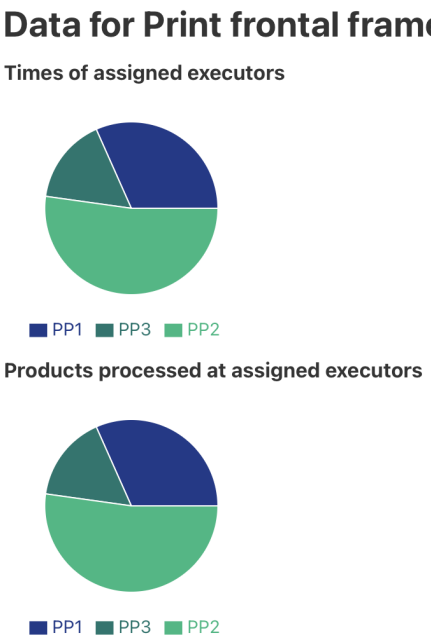

Figure 6.3: Data shown when an activity is selected

average time is calculated as the sum of each job's waiting time in the queue plus its processing time, divided by the total number of jobs handled by the executor. Finally, the *worst* time reflects the maximum observed waiting time plus the processing time, representing the most extreme delay experienced.

For each activity connected to the executor, these three measures are displayed using individual bar charts. The charts are then presented together, enabling direct comparison of performance across activities. Figure 6.4 provides an example in which executor "PP1" is shown to have contributed to three distinct activities.

Finally, at the conclusion of each simulation, the tool automatically assigns a background color to every executor, providing a visual indication of its performance. This color-coding is based on either the executor's availability or its queue length, with three possible states represented by green, yellow, and red. By default, the availability metric is used. This metric is calculated as the ratio between the total time an executor has been engaged (i.e., actively processing tasks, plus the time items have spent waiting in its queue) and the overall duration of the simulation. In this way, the availability measure captures both utilization and congestion effects.

Figure 6.5 illustrates this mechanism by showing executors highlighted in different colors according to their computed availability values.

The system is also designed to be extensible: additional performance and

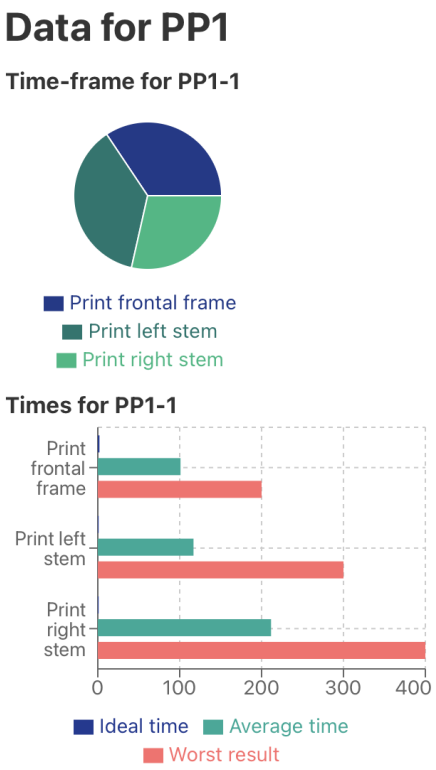

Figure 6.4: Data shown when an executor is selected

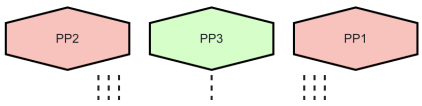

Figure 6.5: Executors with different background color based on their availability

sustainability indicators can be integrated without modifying the core simulation logic. In particular, the data layer already tracks emissions, waste generation, and maintenance costs, which can be surfaced in the dashboard, employed in alternative color-mapping schemes, or used for other kind of analyses. The metrics presented here should therefore be regarded as an initial set intended to showcase the simulator's capabilities rather than an exhaustive catalogue.

Chapter 7

Evaluation

This chapter evaluates the proposed ARMS framework through two complementary perspectives. First, it examines its relevance and usability based on expert feedback. Second, it explores whether ARMS can deliver meaningful tactical insights even while abstracting away low-level operational details.

To achieve this, two manufacturing processes were modeled and analyzed using ARMS, in collaboration with domain experts from an industrial consulting firm. After the modeling sessions, structured debriefs were conducted to capture their perceptions regarding the usefulness of the outputs, as well as the clarity and intuitiveness of the modeling workflow.

While both ARMS and conventional simulation tools aim to support decision-making, their focus differs. Tactical-level analyses address medium-term planning issues, such as balancing capacity, evaluating outsourcing options, or considering new equipment purchases, whereas operational analyses deal with short-term efficiency improvements like layout optimization or material flow adjustments. Our evaluation therefore focuses on ARMS's suitability for tactical contexts rather than operational fine-tuning.

To verify the reliability of ARMS outputs, each process was also modeled in FlexSim, a high-fidelity 3D simulation tool. The FlexSim models serve as a reference point for comparing key performance indicators (KPIs). While FlexSim captures highly detailed spatial and procedural information, its complexity and the technical expertise it demands make it impractical for typical tactical users. ARMS, by contrast, deliberately abstracts these details to make modeling faster and more accessible, yet aims to preserve decision alignment, meaning that even if numerical KPIs differ slightly, the resulting managerial conclusions remain the same.

For both case studies, two what-if scenarios were simulated and compared between ARMS and FlexSim. The analysis focused primarily on production lead time, with complementary consideration of resource utilization, work-in-progress (WIP), and production cost. Together, these experiments demonstrate that ARMS effectively balances usability, abstraction, and tactical decision relevance.

7.1 Process modeling and simulation

7.1.1 Bookends process

The first scenario centers on a small carpentry shop that produces two models of wooden bookends.

> A small carpentry shop produces two models of wooden bookends: "Oak Curve" and "Maple Block". Each set consists of two identical wooden pieces that are sanded and polished. The first step is cutting, which is done using a CNC cutting machine. Cutting one piece takes three minutes. Only one type of wood can be cut at a time, as switching between wood types requires a five-minute adjustment to the saw blade. Next, each piece must be sanded. There is only one sanding station which is operated by an operator, and sanding each piece takes two minutes. After sanding, the same operator brings the product to a polish station which is ten meters from the sanding station to avoid the contamination of the wood residue of the sanding. The polishing process takes three minutes per piece. During polishing, a finishing oil is applied, and then the bookend sets are packed in boxes. This final step is also performed by the same operator that does the polishing.

Figure 7.1 illustrates this process as modeled in ARMS.

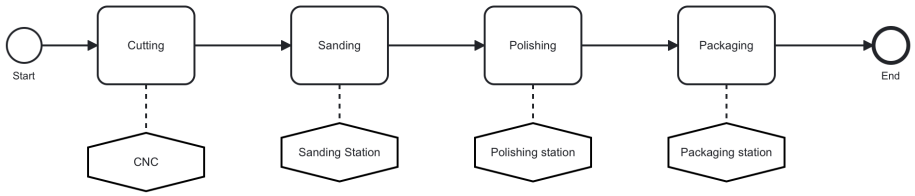

Figure 7.1: Bookends process modeled with ARMS

The tactical question explored here was straightforward: "How many operators are needed to ensure an efficient workflow?" Two alternative what-if scenarios were examined:

- Scenario 2 (we assume scenario 1 is the one described at the begging of the section): increase the number of operators from one to two.
- Scenario 3: increase the number of operators from one to three.

The results, summarized in Figure 7.2, show the time required to produce ten Oak Curve and ten Maple Block bookends under these different staffing levels and operator movement speeds.

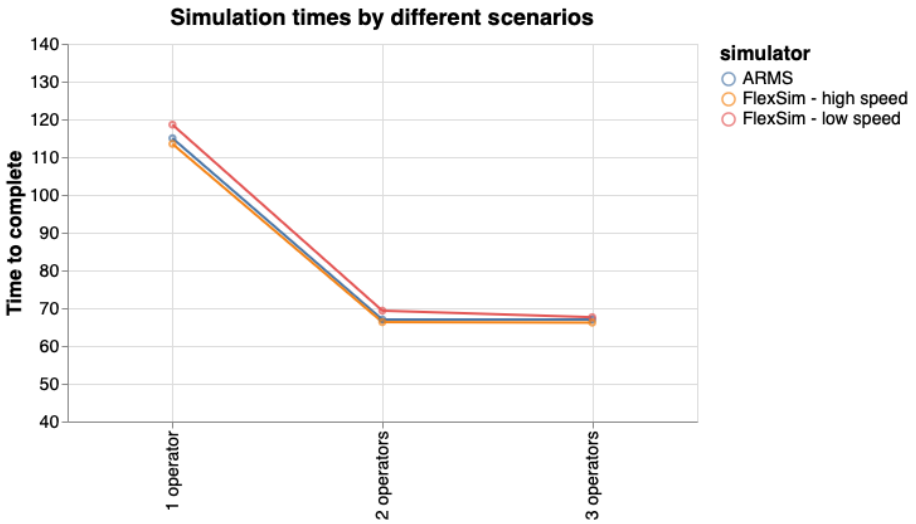

Figure 7.2: Time to complete the bookends process with different amount of operators. High and low speed refers to the operators' movement speed.

7.1.2 Desk lamps process

The second process describes the assembly of desk lamps in a furniture factory.

> A furniture factory produces desk lamps, among other items. Each desk lamp consists of three components: a base, an arm, and a lampshade. The lamps are available in two color variants: black and white. All components are pre-manufactured and delivered daily. They are stored on a rack that holds up to fifty units of each component. Two workers, Alice and Bob, handle the assembly. The assembly process involves connecting the base, attaching the arm, and screwing on the shade, which takes five minutes per lamp. Alice assembles only black lamps, while Bob assembles only white ones. Each assembled lamp must be tested at a testing station, where it is plugged in and evaluated. Each test takes two minutes. If a lamp fails, it is returned to the assembly station for rework. The rework rate is approximately five percent. The packaging is performed by two additional workers, Charlie and David. Each lamp is bubble wrapped and boxed. Charlie takes one minute per lamp, while David takes two minutes. At the end of the day, a courier collects all the ready-to-ship lamps.

Figure 7.3 presents the process modeled in ARMS.

Here, the tactical objective was to determine the optimal combination of operator skills and staffing. Two what-if scenarios were explored:

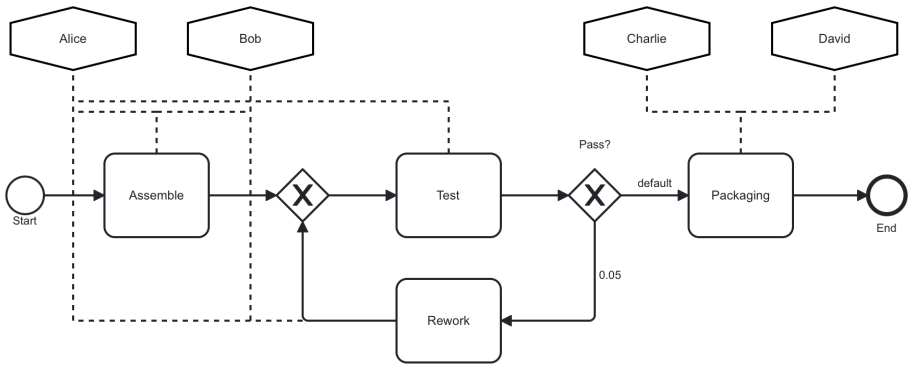

Figure 7.3: Lamps process modeled with ARMS

- Scenario 2: increase the skill of Alice reducing the time she takes in assembling a lamp from five minutes to two.
- Scenario 3: increase the skill of Alice as in Scenario 2 but remove Bob reducing the total amount of personnel. In this case Alice has to take care of white lamps as well.

Figure 7.4 summarizes the results for the production of ten white and fifteen black lamps across these scenarios.

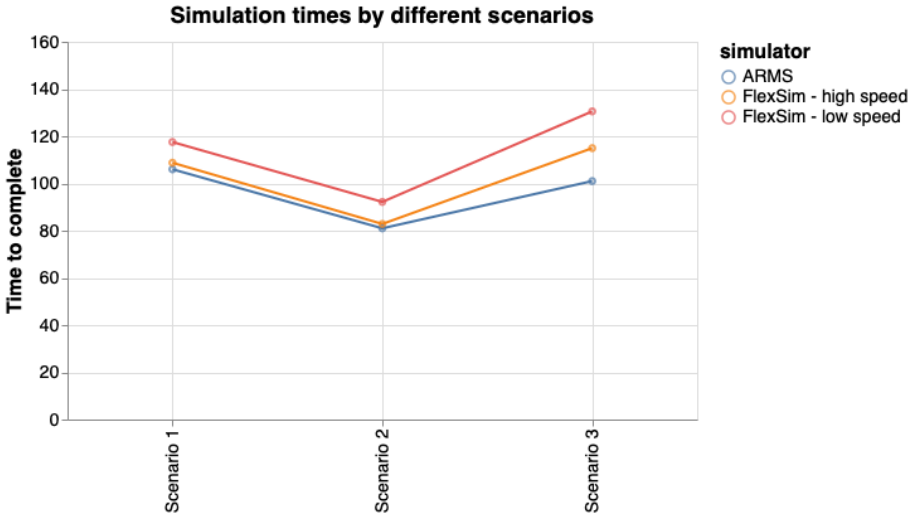

Figure 7.4: Time to complete the lamps process in different scenarios. High and low speed refers to the operators' movement speed.

7.1.3 Feedback from experts

After completing both modeling exercises, domain experts participated in structured debrief sessions to assess their experience with ARMS. They reported that the tool successfully supported the intended tactical analyses helping them, for instance, to evaluate whether to hire additional operators or invest in skill development to improve productivity. Overall, they found ARMS intuitive and sufficiently powerful for scenario exploration.

At the same time, they highlighted several opportunities for improvement:

- **Automation data handling**: Experts emphasized the importance of automating the import of data from external sources, especially large Excel-based inputs such as compatibility matrices. Manual entry was considered both time-consuming and prone to errors.
- **Dashboard enhancements**: They suggested enriching the simulation dashboard with additional KPIs to better support data-driven tactical decisions.

7.2 Discussion

The results presented in Figures 7.2 and 7.4 show that while the numerical outcomes from ARMS and FlexSim are not identical, they lead to the same managerial conclusions. In other words, despite its higher level of abstraction, ARMS enables decision-makers to reach the same tactical insights as they would with a full-fidelity simulation.

The differences between the two tools stem primarily from their treatment of physical space. FlexSim explicitly simulates three-dimensional movement, meaning that travel between stations consumes simulation time. When the movement speed of operators in FlexSim was increased, the results converged closely with those from ARMS.

In the lamp case, particularly in Scenario 3 (where Alice handles all assembly, testing, and rework), a residual difference remained even under faster movement conditions. This is explained by the fact that Alice's workload effectively doubles in FlexSim, increasing the frequency of her movements and thus her total process time.

Notice that, in general, the increase in processing times observed in FlexSim can be replicated in ARMS by adding "transfer" activities between existing tasks and by exploiting the *affinity* between activities and executors to ensure that each executor always performs the corresponding movement activity for the products they handle. However, this approach presents two main problems for only limited improvements toward the ground-truth result.

1. **Manual configuration effort**: Unlike FlexSim, which automatically calculates transfer times based on layout and speed, ARMS requires manual estimation of these durations. Any layout change would therefore necessitate reconfiguring all affected transfer activities.

2. **Scope of abstraction**: Adding these detailed transfer steps would push ARMS beyond its intended abstraction level. The model would become cluttered with auxiliary activities unrelated to the core production logic, reducing usability without adding meaningful tactical insight.

For these reasons, ARMS intentionally maintains its simplified representation, prioritizing clarity and tactical focus over operational exactness. The evaluation confirms that this balance, between abstraction and analytical reliability, enables ARMS to serve effectively as a decision-support tool for tactical manufacturing analysis.

Chapter 8

Conclusions

This research set out to address a persistent challenge in manufacturing analysis: how to enable tactical decision-making through simulation without relying on complex, programming-intensive simulators. Traditional simulation tools are powerful but often inaccessible to analysts without technical backgrounds, limiting their practical use in tactical or continuous improvement contexts. Conversely, activity-centric modeling tools offer intuitive representations of workflows but lack the expressiveness needed to represent manufacturing-specific behaviors such as resource constraints, material flows, and operational dependencies. This thesis has demonstrated that it is possible to reconcile these two paradigms, activity-centric and resource-centric modeling, through a hybrid conceptual framework, a new meta-model, and the ARMS simulation platform.

8.1 Summary of the work

The motivation behind this work stemmed from real industrial needs: analysts within manufacturing consulting environments required a means to perform reliable what-if analyses without extensive technical training. The thesis first reviewed the limitations of existing simulation approaches and argued for a hybrid paradigm capable of preserving BPMN's intuitiveness while enriching it with the realism of resource-oriented modeling. Building upon this foundation, a novel meta-model was proposed, extending the activity-centric view with explicit constructs for executors, processables, and inventories.

This conceptual layer enables a more coherent representation of how resources interact with activities, making it possible to reason about both process flow and operational constraints within a unified modeling environment. The conceptual framework was realized through the Activity–Resource Modeling Simulator (ARMS), a platform that integrates three main components: a graphical modeler based on BPMN extended with manufacturing semantics, a simulation engine that executes models according to the new meta-model, and a dashboard for performance analysis and visualization. Together, these components provide an

accessible end-to-end environment for modeling, simulation, and tactical decision support.

The evaluation chapter demonstrated the feasibility and effectiveness of ARMS through industrial scenarios derived from real-world experiences. Compared to a traditional simulator such as FlexSim, ARMS required fewer steps to build and execute manufacturing process simulations, while achieving comparable accuracy in key tactical indicators such as lead time and throughput. Domain experts involved in the validation process confirmed that ARMS moves in the right direction: it simplifies model creation, reduces configuration effort, and maintains sufficient realism to support tactical decision-making. Their feedback also highlighted areas for improvement, including more streamlined data input and an enriched set of key performance indicators.

8.2 Main contributions

This research makes two primary contributions, one conceptual and one practical, each advancing the state of the art in manufacturing process simulation.

1. **A new hybrid meta-model** that integrates activity-centric and resource-centric perspectives. This meta-model provides a unified vocabulary for representing processes where resources and activities co-determine system behavior. It extends a generic activity-centric approach to capture manufacturing-specific concepts such as executor availability, product flow, and inventory buffering, while retaining the intuitive logic of process diagrams.

   This meta-model is formally defined using the Ecore specification, providing a machine-readable schema that ensures structural integrity and consistency of models. In the current state of the work, this Ecore definition serves as a formal reference that guided the design and implementation of the ARMS platform, but it is not yet directly exploited by the simulator or the modeler. Nevertheless, this explicit formalization establishes a solid foundation for future Model-Driven Engineering extensions. Future technical iterations will leverage the existing Ecore definition to implement automated model-to-model transformations, particularly to support formal model checking, systematic model validation, and automated code generation.

2. **The ARMS simulation platform**, an operational implementation of the hybrid paradigm. ARMS demonstrates how analysts can perform what-if analyses and tactical decision-making without requiring programming skills. By lowering the entry barrier to simulation-based analysis, ARMS makes tactical decision support more accessible to professionals in manufacturing and business domains.

Beyond these direct outputs, the research contributes methodologically by providing a structured approach to integrating conceptual modeling with tool design and empirical evaluation. It shows that bridging theoretical rigor and

usability is not only possible but essential for extending simulation's role in modern manufacturing decision processes.

8.3 Limitations and Future Works

While the results confirm the feasibility and advantages of the proposed approach, several limitations point to promising avenues for future research.

Scope of validation

The validation of ARMS was conducted with a limited number of industrial case studies and domain experts, primarily within a single consulting context. Although the feedback was encouraging, broader empirical testing is needed across different manufacturing sectors (e.g., discrete, process, and hybrid manufacturing) and organizational roles (e.g., planners, engineers, operations managers). Expanding the scope of validation would strengthen the generalizability of the findings and reveal domain-specific modeling needs that could inform the evolution of the meta-model and tool.

Coverage of BPMN and manufacturing semantics

The current implementation supports only a subset of BPMN elements and manufacturing constructs. While sufficient for tactical-level analyses, it limits expressiveness for more complex scenarios involving parallel resource contention, exception handling, or dynamic reconfiguration of processes. Future work will aim to:

- Achieve fuller compliance with the BPMN 2.0 standard.
- Enrich the meta-model with manufacturing-specific features such as maintenance activities and executors availability.
- Introduce interoperability mechanisms to connect ARMS with existing enterprise tools.

Integration and multi-level simulation

A key future direction is to extend ARMS towards *multi-level simulation*, enabling integration of data and models across different abstraction layers: from operational shop-floor processes to tactical planning and strategic decision-making. Figure 8.1 represents this idea. This would involve:

- Connecting ARMS with a **forecasting and logistics systems** to simulate end-to-end supply chains, linking demand fluctuations with production capacity and inventory policies;
- Incorporating **sensor and IoT data** to allow dynamic model updates based on real-time factory conditions;

- Integrating with **transactional processes** (e.g., document or workflow management) to capture decision delays, communication bottlenecks, or administrative overheads.

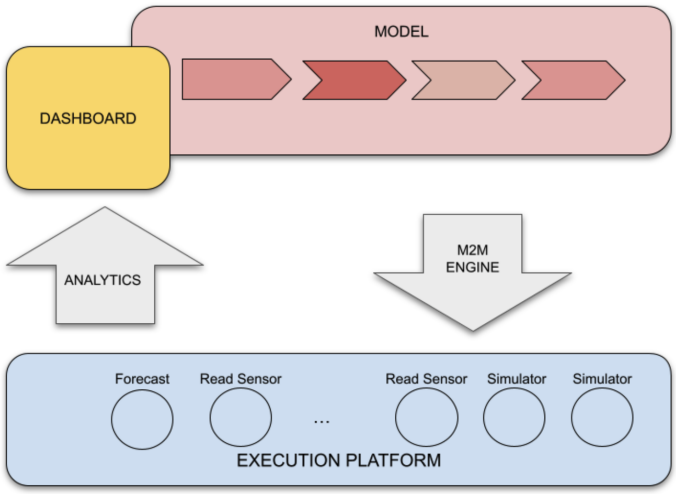

Figure 8.1: Multi-model project direction

Automation and AI-assisted modeling

Another promising possible improvements concerns the automation of model creation and calibration. Text-based model generation through natural language processing (NLP) or large language models (LLM) could enable analysts to describe a process in plain language and automatically generate a corresponding ARMS model. Likewise, data-driven parameter estimation and intelligent tuning mechanisms could reduce manual configuration effort and improve accuracy. Exploring how AI can complement human expertise in model generation, scenario exploration, and performance optimization will be a major focus of future research.

Usability and human-centered evaluation

Finally, although usability was a central goal, further human-centered studies are needed to systematically evaluate how analysts interact with ARMS over time. Future iterations of the platform will benefit from longitudinal usability testing, cognitive walkthroughs, and participatory design workshops to refine interaction flows, visualization dashboards, and onboarding materials. These studies will help ensure that ARMS remains both technically robust and intuitively usable for non-programming analysts.

Summary

Future developments will thus proceed along three complementary paths: (i) conceptual refinement of the meta-model to capture additional manufacturing dynamics; (ii) technical expansion towards multi-level and data-integrated simulation; and (iii) methodological enhancement through AI support and human-centered evaluation. Together, these efforts will strengthen ARMS as a robust, intelligent, and accessible environment for tactical and strategic simulation in manufacturing.

8.4 Implications for Tactical Manufacturing Simulation

This thesis advances business–manufacturing process simulation by operationalizing a BPMN-extended, hybrid activity–resource meta-model and the ARMS platform. Together, they let analysts model and simulate processes that explicitly accounting for resource constraints, material flows, and inventories, while retaining BPMN's intuitive control-flow for rapid model construction.

By fusing BPM's accessibility with manufacturing semantics in a single modeling environment, ARMS democratizes tactical what-if analysis: tasks that previously required code-heavy, specialist simulators become routine decision-support activities for planners and analysts.

The hybrid paradigm also establishes a concrete foundation for next-generation manufacturing simulators, encouraging tools that support fast, low-effort scenario exploration and iterative refinement. In doing so, it shows that simulation can be both powerful and accessible, lowering barriers to model creation, accelerating insight, and enabling better tactical decisions without reliance on scarce simulation expertise. This work contributes to more human-centered digital manufacturing tools where usability and analytical depth coexist to drive continuous improvement and operational excellence.

Appendix A

Formal Ecore Meta-model Specification

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="arms"
       nsURI="https://jjocram.com" nsPrefix="">
4    <eClassifiers xsi:type="ecore:EClass" name="Activity">
5      <eStructuralFeatures xsi:type="ecore:EAttribute" name="priority"
       eType="ecore:EDataType
       http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
6      <eStructuralFeatures xsi:type="ecore:EReference"
       name="compatibilities" upperBound="-1"
7          eType="#//Compatibility" containment="true"/>
8    </eClassifiers>
9    <eClassifiers xsi:type="ecore:EClass" name="ProductFamily">
10     <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
       eType="ecore:EDataType
       http://www.eclipse.org/emf/2002/Ecore#//EString"/>
11     <eStructuralFeatures xsi:type="ecore:EReference" name="subFamilies"
       upperBound="-1"
12         eType="#//ProductFamily" containment="true"/>
13     <eStructuralFeatures xsi:type="ecore:EReference" name="consumedBy"
       upperBound="-1"
14         eType="#//Transformation"/>
15     <eStructuralFeatures xsi:type="ecore:EReference" name="producedBy"
       upperBound="-1"
16         eType="#//Transformation"/>
17   </eClassifiers>
18   <eClassifiers xsi:type="ecore:EClass" name="Executor">
19     <eStructuralFeatures xsi:type="ecore:EAttribute" name="quantity"
       eType="ecore:EDataType
       http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
20     <eStructuralFeatures xsi:type="ecore:EReference" name="capabilities"
       upperBound="-1"
21         eType="#//Compatibility" containment="true"/>
```

```
22      <eStructuralFeatures xsi:type="ecore:EReference" name="history"
        upperBound="-1"
23          eType="#//Assignment"/>
24      <eStructuralFeatures xsi:type="ecore:EReference" name="currentWork"
        upperBound="-1"
25          eType="#//WorkItem"/>
26    </eClassifiers>
27    <eClassifiers xsi:type="ecore:EClass" name="Accessory">
28      <eStructuralFeatures xsi:type="ecore:EAttribute" name="quantity"
        eType="ecore:EDataType
        http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
29    </eClassifiers>
30    <eClassifiers xsi:type="ecore:EClass" name="Inventory">
31      <eStructuralFeatures xsi:type="ecore:EAttribute" name="quantity"
        eType="ecore:EDataType
        http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
32      <eStructuralFeatures xsi:type="ecore:EReference" name="products"
        upperBound="-1"
33          eType="#//Product" containment="true"/>
34      <eStructuralFeatures xsi:type="ecore:EReference"
        name="categorizedBy" lowerBound="1"
35          eType="#//ProductFamily"/>
36    </eClassifiers>
37    <eClassifiers xsi:type="ecore:EClass" name="Compatibility">
38      <eStructuralFeatures xsi:type="ecore:EAttribute" name="duration"
        eType="ecore:EDataType
        http://www.eclipse.org/emf/2002/Ecore#//EFloat"/>
39      <eStructuralFeatures xsi:type="ecore:EReference" name="activity"
        lowerBound="1"
40          eType="#//Activity"/>
41      <eStructuralFeatures xsi:type="ecore:EReference" name="relevantFor"
        lowerBound="1"
42          eType="#//ProductFamily"/>
43      <eStructuralFeatures xsi:type="ecore:EReference" name="supportedBy"
        lowerBound="1"
44          eType="#//Executor"/>
45      <eStructuralFeatures xsi:type="ecore:EReference" name="requires"
        upperBound="-1"
46          eType="#//Accessory"/>
47    </eClassifiers>
48    <eClassifiers xsi:type="ecore:EClass" name="Transformation">
49      <eStructuralFeatures xsi:type="ecore:EReference" name="executedVia"
        lowerBound="1"
50          eType="#//Activity"/>
51      <eStructuralFeatures xsi:type="ecore:EReference" name="consume"
        upperBound="-1"
52          eType="#//ProductFamily"/>
53      <eStructuralFeatures xsi:type="ecore:EReference" name="produce"
        upperBound="-1"
54          eType="#//ProductFamily"/>
55      <eStructuralFeatures xsi:type="ecore:EReference" name="composedBy"
        upperBound="-1"
56          eType="#//WorkItem"/>
57    </eClassifiers>
58    <eClassifiers xsi:type="ecore:EClass" name="Product">
59      <eStructuralFeatures xsi:type="ecore:EReference" name="type"
        lowerBound="1" eType="#//ProductFamily"/>
```

```
60      <eStructuralFeatures xsi:type="ecore:EReference" name="storedIn"
        lowerBound="1"
61          eType="#//Inventory"/>
62    </eClassifiers>
63    <eClassifiers xsi:type="ecore:EClass" name="WorkItem">
64      <eStructuralFeatures xsi:type="ecore:EReference" name="activity"
        lowerBound="1"
65          eType="#//Activity"/>
66      <eStructuralFeatures xsi:type="ecore:EReference"
        name="transformation" lowerBound="1"
67          eType="#//Transformation"/>
68      <eStructuralFeatures xsi:type="ecore:EReference" name="assignment"
        eType="#//Assignment"
69          containment="true"/>
70    </eClassifiers>
71    <eClassifiers xsi:type="ecore:EClass" name="Assignment">
72      <eStructuralFeatures xsi:type="ecore:EReference"
        name="assignedThrough" lowerBound="1"
73          eType="#//Executor"/>
74      <eStructuralFeatures xsi:type="ecore:EReference" name="workOn"
        upperBound="-1"
75          eType="#//Product"/>
76    </eClassifiers>
77  </ecore:EPackage>
```

Bibliography

[1] Wil M. P. van der Aalst. "Business Process Management: A Comprehensive Survey". In: *ISRN Software Engineering* 2013 (Feb. 2013), pp. 1–37. ISSN: 2090-7680. DOI: 10.1155/2013/507984.

[2] Ihsane Abouzid and Rajaa Saidi. "Proposal of BPMN extensions for modelling manufacturing processes". In: *2019 5th International Conference on Optimization and Applications (ICOA)*. 2019, pp. 1–6. DOI: 10.1109/ICOA.2019.8727651.

[3] Bugra Alkan et al. "Complexity in manufacturing systems and its measures: a literature review". In: *European J. of Industrial Engineering* 12.1 (2018), p. 116. ISSN: 1751-5262. DOI: 10.1504/ejie.2018.089883.

[4] Satyajith Amaran et al. "Simulation optimization: a review of algorithms and applications". In: *4OR* 12.4 (Nov. 2014), pp. 301–333. ISSN: 1614-2411. DOI: 10.1007/s10288-014-0275-2.

[5] Robert Newton Anthony. *Planning and Control Systems: A Framework for Analysis*. Massachusetts, United States: Boston : Division of Research, Graduate School of Business Administration, Harvard University, 1965. ISBN: 0875840477. URL: https://searchworks.stanford.edu/view/10046478.

[6] Nelly Bencomo et al. *Models@run.time: foundations, applications, and roadmaps*. Vol. 8378. Springer, 2014. DOI: 10.1007/978-3-319-08915-7.

[7] Paolo Bocciarelli and Andrea D'Ambrogio. "A BPMN extension for modeling non functional properties of business processes". In: *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*. TMS-DEVS '11. Boston, Massachusetts: Society for Computer Simulation International, 2011, pp. 160–168.

[8] Paolo Bocciarelli, Andrea D'Ambrogio, and Gerd Wagner. "Resource Modeling in Business Process Simulation". In: *Winter Simulation Conference, WSC 2022, Singapore, December 11-14, 2022*. IEEE, 2022, pp. 1296–1310. DOI: 10.1109/WSC57314.2022.10015259.

[9] Paolo Bocciarelli et al. "A BPMN extension for modeling Cyber-Physical-Production-Systems in the context of Industry 4.0". In: *14th IEEE International Conference on Networking, Sensing and Control, ICNSC 2017, Calabria, Italy, May 16-18, 2017*. Ed. by Giancarlo Fortino et al. IEEE, 2017, pp. 599–604. DOI: 10.1109/ICNSC.2017.8000159.

[10] Paolo Bocciarelli et al. "A BPMN Extension to Enable the Explicit Modeling of Task Resources". In: *Proceedings of the 2nd INCOSE Italia Conference on Systems Engineering, Turin, Italy, November 14-16, 2016*. Ed. by Eugenio Brusa et al. Vol. 1728. CEUR Workshop Proceedings. CEUR-WS.org, 2016, pp. 40–47. URL: https://ceur-ws.org/Vol-1728/paper5.pdf.

[11] Sally C. Brailsford et al. "Hybrid simulation modelling in operational research: A state-of-the-art review". In: *European Journal of Operational Research* 278.3 (Nov. 2019), pp. 721–737. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2018.10.025.

[12] Marco Brambilla, Piero Fraternali, and Carmen Vaca. "BPMN and Design Patterns for Engineering Social BPM Solutions". In: *Business Process Management Workshops*. Springer Berlin Heidelberg, 2012, pp. 219–230. ISBN: 9783642281082. DOI: 10.1007/978-3-642-28108-2_22.

[13] Richard Braun and Werner Esswein. "Classification of Domain-Specific BPMN Extensions". In: *The Practice of Enterprise Modeling*. Springer Berlin Heidelberg, 2014, pp. 42–57. ISBN: 9783662455012. DOI: 10.1007/978-3-662-45501-2_4.

[14] Jan vom Brocke, Alan Hevner, and Alexander Maedche. "Introduction to Design Science Research". In: *Design Science Research. Cases*. Springer International Publishing, 2020, pp. 1–13. ISBN: 9783030467814. DOI: 10.1007/978-3-030-46781-4_1.

[15] James Byrne, Cathal Heavey, and P.J. Byrne. "A review of Web-based simulation and supporting tools". In: *Simulation Modelling Practice and Theory* 18.3 (2010), pp. 253–276. ISSN: 1569-190X. DOI: 10.1016/j.simpat.2009.09.013. URL: https://www.sciencedirect.com/science/article/pii/S1569190X0900149X.

[16] Davide Cafasso et al. "Framework for Selecting Manufacturing Simulation Software in Industry 4.0 Environment". In: *Sustainability* 12.15 (July 2020), p. 5909. ISSN: 2071-1050. DOI: 10.3390/su12155909.

[17] J.S. Carson. "Introduction to modeling and simulation". In: *Proceedings of the Winter Simulation Conference, 2005.* 2005. DOI: 10.1109/WSC.2005.1574235.

[18] Armando Cartenì and Stefano de Luca. "Tactical and strategic planning for a container terminal: Modelling issues within a discrete event simulation approach". In: *Simulation Modelling Practice and Theory* 21.1 (2012), pp. 123–145. ISSN: 1569-190X. DOI: 10.1016/j.simpat.2011.10.

005. URL: https://www.sciencedirect.com/science/article/pii/S1569190X11001730.

[19] Nancy Cartwright. *How the laws of physics lie*. OUP Oxford, 1983.

[20] Guanghsu Chang and William Peterson. "Modeling and Analysis of Flexible Manufacturing Systems: A Simulation Study". In: *2015 ASEE Annual Conference and Exposition Proceedings*. ASEE Conferences, 2015, pp. 26.1162.1–26.1162.19.

[21] David Chapela-Campa et al. "A framework for measuring the quality of business process simulation models". In: *Information Systems* 127 (Jan. 2025), p. 102447. ISSN: 0306-4379. DOI: 10.1016/j.is.2024.102447.

[22] G Dagkakis and C Heavey. "A review of open source discrete event simulation software for operations research". In: *Journal of Simulation* 10.3 (Aug. 2016), pp. 193–206. ISSN: 1747-7786. DOI: 10.1057/jos.2015.9.

[23] Thomas H Davenport. *Process innovation: reengineering work through information technology*. Harvard business press, 1993.

[24] R. B. Deal, Averill M. Law, and W. David Kelton. "Simulation Modeling and Analysis". In: *Technometrics* 36.4 (Nov. 1994), p. 429. ISSN: 0040-1706. DOI: 10.2307/1269971.

[25] Angelo Di Iorio, Marco Ferrati, and Davide Rossi. "ARMS: Activity-Resource Modelling Simulator". In: *Simulation Tools and Techniques*. Ed. by Angel A. Juan et al. Cham: Springer Nature Switzerland, 2025, pp. 167–185. ISBN: 978-3-031-87345-4.

[26] Alexandre Dolgui, Dmitry Ivanov, and Boris Sokolov. "Ripple effect in the supply chain: an analysis and recent literature". In: *International Journal of Production Research* 56.1–2 (Oct. 2017), pp. 414–430. ISSN: 1366-588X. DOI: 10.1080/00207543.2017.1387680.

[27] Michael J Drevna and Cynthia J Kasales. "Introduction to ARENA". In: *Proceedings of Winter Simulation Conference*. IEEE, 1994, pp. 431–436.

[28] Jonnro Erasmus et al. "The HORSE Project: The Application of Business Process Management for Flexibility in Smart Manufacturing". In: *Applied Sciences* 10.12 (2020). ISSN: 2076-3417. DOI: 10.3390/app10124145. URL: https://www.mdpi.com/2076-3417/10/12/4145.

[29] Jonnro Erasmus et al. "Using business process models for the specification of manufacturing operations". In: *Comput. Ind.* 123 (2020), p. 103297. DOI: 10.1016/J.COMPIND.2020.103297.

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

[31] Luca Fumagalli et al. "Framework for simulation software selection". In: *Journal of Simulation* 13.4 (Apr. 2019), pp. 286–303. ISSN: 1747-7786. DOI: 10.1080/17477778.2019.1598782.

[32] A. García-Domínguez, Mariano Marcos, and I. Medina. "A comparison of BPMN 2.0 with other notations for manufacturing processes". In: *AIP Conference Proceedings*. Vol. 1431. Apr. 2012, pp. 593–600. DOI: 10.1063/1.4707613.

[33] Eliyahu M Goldratt et al. *Theory of constraints*. North River Croton-on-Hudson, 1990.

[34] David Goldsman and Paul Goldsman. "Discrete-event simulation". In: *Modeling and Simulation in the Systems Engineering Life Cycle: Core Concepts and Accompanying Lectures*. Springer, 2015, pp. 103–109.

[35] Matteo Golfarelli, Stefano Rizzi, and Andrea Proli. "Designing what-if analysis: towards a methodology". In: *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*. CIKM06. ACM, Nov. 2006, pp. 51–58. DOI: 10.1145/1183512.1183523.

[36] Paul Goodall, Richard Sharpe, and Andrew West. "A data-driven simulation to support remanufacturing operations". In: *Computers in Industry* 105 (2019), pp. 48–60. ISSN: 0166-3615. DOI: 10.1016/j.compind.2018.11.001. URL: https://www.sciencedirect.com/science/article/pii/S0166361518303191.

[37] Ruddy Guerrero et al. "Simulation Model for Wire Harness Design in the Car Production Line Optimization Using the SimPy Library". In: *Sustainability* 14.12 (June 2022), p. 7212. ISSN: 2071-1050. DOI: 10.3390/su14127212.

[38] Randolph W. Hall. "Queueing Methods: For Services and Manufacturing". In: *Queueing Methods: For Services and Manufacturing*. 1991. URL: https://api.semanticscholar.org/CorpusID:57133109.

[39] Ruud van der Ham. "salabim: discrete event simulation and animation in Python". In: *Journal of Open Source Software* 3.27 (July 2018), p. 767. ISSN: 2475-9066. DOI: 10.21105/joss.00767.

[40] Stefan Hanenberg et al. "An empirical study on the impact of static typing on software maintainability". In: *Empirical Software Engineering* 19.5 (Dec. 2013), pp. 1335–1382. ISSN: 1573-7616. DOI: 10.1007/s10664-013-9289-1.

[41] Germán Herrera-Vidal et al. "Measuring Complexity in Manufacturing: Integrating Entropic Methods, Programming and Simulation". In: *Entropy* 27.1 (Jan. 2025), p. 50. ISSN: 1099-4300. DOI: 10.3390/e27010050.

[42] Alexander Herzog. "Simulation mit dem Warteschlangensimulator". In: *Angenommen zur Veröffentlichung* (2021).

[43] B Hollocks. "Forty years of discrete-event simulation—a personal reflection". In: *Journal of The Operational Research Society - J OPER RES SOC* 57 (Dec. 2006), pp. 1383–1399. DOI: 10.1057/palgrave.jors.2602128.

[44] Wallace J Hopp and Mark L Spearman. *Factory physics*. Waveland Press, 2011.

[45] Kristina Höse et al. "Manufacturing Flexibility through Industry 4.0 Technological Concepts—Impact and Assessment". In: *Global Journal of Flexible Systems Management* 24.2 (Apr. 2023), pp. 271–289. ISSN: 0974-0198. DOI: 10.1007/s40171-023-00339-y.

[46] Stefan Jablonski and Christoph Bussler. *Workflow management - modeling concepts, architecture and implementation*. International Thomson, 1996. ISBN: 978-1-85032-222-1.

[47] Mohsen Jahangirian et al. "Simulation in manufacturing and business: A review". In: *European Journal of Operational Research* 203.1 (May 2010), pp. 1–13. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2009.06.004.

[48] Eeva Järvenpää et al. "The development of an ontology for describing the capabilities of manufacturing resources". In: *J. Intell. Manuf.* 30.2 (2019), pp. 959–978. DOI: 10.1007/S10845-018-1427-6.

[49] Philip Joschko et al. "Business process simulation with IYOPRO und DESMO-J". In: *Proceedings of the International Workshop on Applied Modeling & Simulation, Rome, Italy*. 2012, pp. 125–132.

[50] Magdalena Jurczyk-Bunkowska. "Tactical manufacturing capacity planning based on discrete event simulation and throughput accounting: A case study of medium sized production enterprise". In: *Advances in Production Engineering & Management* 16.3 (2021), pp. 335–347.

[51] Deogratias Kibira et al. "Integrating data analytics and simulation methods to support manufacturing decision making". In: *2015 Winter Simulation Conference (WSC)*. IEEE, Dec. 2015, pp. 2100–2111. DOI: 10.1109/wsc.2015.7408324.

[52] Christopher Koliba et al. "Assessing strategic, tactical, and operational decision-making and risk in a livestock production chain through experimental simulation platforms". In: *Frontiers in Veterinary Science* 9 (Oct. 2022). ISSN: 2297-1769. DOI: 10.3389/fvets.2022.962788.

[53] Kateryna Kovbasiuk et al. "ANALYSIS OF THE SELECTED SIMULATION SOFTWARE PACKAGES: A STUDY". In: *Acta Tecnología* 7.4 (Dec. 2021), pp. 111–120. ISSN: 2453-675X. DOI: 10.22306/atec.v7i4.120.

[54] Damian Krenczyk, Reggie Davidrajuh, and Bozena Skolud. "Comparing Two Methodologies for Modeling and Simulation of Discrete-Event Based Automated Warehouses Systems". In: *Advances in Manufacturing II* (2019). Ed. by Adam Hamrol, Agnieszka Kujawińska, and Manuel Francisco Suarez Barraza, pp. 161–175.

[55] Vinay Kulkarni, Souvik Barat, and Tony Clark. "Towards adaptive enterprises using digital twins". In: *2019 winter simulation conference (WSC)*. IEEE. 2019, pp. 60–74.

[56] Karen M Kuntz et al. "Best Practices for Decision and Simulation Modeling". In: *Decision and Simulation Modeling in Systematic Reviews*. 2013. URL: https://api.semanticscholar.org/CorpusID:68332832.

[57] Averill M Law, W David Kelton, and W David Kelton. *Simulation modeling and analysis*. Vol. 3. Mcgraw-hill New York, 2007.

[58] Sanghoon Lee, Hyunbo Cho, and Mooyoung Jung. "A conceptual framework for the generation of simulation models from process plans and resource configuration". In: *International Journal of Production Research* 38.4 (2000), pp. 811–828.

[59] Yung-Tsun Tina Lee, Frank Riddick, and Björn Johan Ingemar Johansson. "Core Manufacturing Simulation Data - a manufacturing simulation integration standard: overview and case studies". In: *Int. J. Comput. Integr. Manuf.* 24.8 (2011), pp. 689–709. DOI: 10.1080/0951192X.2011.574154.

[60] Kashif Mahmood et al. "Performance Analysis of a Flexible Manufacturing System (FMS)". In: *Procedia CIRP* 63 (2017), pp. 424–429. ISSN: 2212-8271. DOI: 10.1016/j.procir.2017.03.123.

[61] Marvin Carl May et al. "Ontology-based production simulation with ontologysim". In: *Applied Sciences* 12.3 (2022), p. 1608.

[62] Dimitris Mourtzis, Michael Doukas, and Dimitra Bernidaki. "Simulation in manufacturing: Review and challenges". In: *Procedia Cirp* 25 (2014), pp. 213–229.

[63] Richard Müller and Andreas Rogge-Solti. "BPMN for Healthcare Processes". In: *Central-European Workshop on Services and their Composition*. 2011. URL: https://api.semanticscholar.org/CorpusID:14255935.

[64] Ashkan Negahban and Jeffrey S. Smith. "Simulation for manufacturing system design and operation: Literature review and analysis". In: *Journal of Manufacturing Systems* 33.2 (Apr. 2014), pp. 241–261. ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2013.12.007.

[65] OMG. *Business Process Model and Notation (BPMN)*. 2011.

[66] OMG. *Meta Object Facility (MOF)*. 2016.

[67] José Miguel Pérez-Álvarez et al. "Tactical Business-Process-Decision Support based on KPIs Monitoring and Validation". In: *Computers in Industry* 102 (2018), pp. 23–39. ISSN: 0166-3615. DOI: 10.1016/j.compind.2018.08.001. URL: https://www.sciencedirect.com/science/article/pii/S0166361517307819.

[68] Renata Petrevska Nechkoska. "Introduction to Tactical Management Research". In: *Tactical Management in Complexity*. Springer International Publishing, Aug. 2019, pp. 1–26. ISBN: 9783030228040. DOI: 10.1007/978-3-030-22804-0_1.

[69] C.A. Petri. "Nets, time and space". In: *Theoretical Computer Science* 153.1–2 (Jan. 1996), pp. 3–48. ISSN: 0304-3975. DOI: 10.1016/0304-3975(95)00116-6.

[70] Melanie Polderdijk et al. "A Visualization of Human Physical Risks in Manufacturing Processes Using BPMN". In: *Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*. Ed. by Ernest Teniente and Matthias Weidlich. Vol. 308. Lecture Notes in Business Information Processing. Springer, 2017, pp. 732–743. DOI: 10.1007/978-3-319-74030-0_58.

[71] Luise Pufahl and Mathias Weske. "Batch Activities in Process Modeling and Execution". In: *Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings*. Ed. by Samik Basu et al. Vol. 8274. Lecture Notes in Computer Science. Springer, 2013, pp. 283–297. DOI: 10.1007/978-3-642-45005-1_20.

[72] Luise Pufahl, Tsun Yin Wong, and Mathias Weske. "Design of an Extensible BPMN Process Simulator". In: *Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*. Ed. by Ernest Teniente and Matthias Weidlich. Vol. 308. Lecture Notes in Business Information Processing. Springer, 2017, pp. 782–795. DOI: 10.1007/978-3-319-74030-0_62.

[73] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: https://arxiv.org/abs/1908.10084.

[74] Kristina Rosenthal, Benjamin Ternes, and Stefan Strecker. "Business Process Simulation: A Systematic Literature Review". In: *European Conference on Information Systems*. 2018. URL: https://api.semanticscholar.org/CorpusID:56148450.

[75] Nick Russell, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. *Workflow Patterns: The Definitive Guide*. MIT Press, 2016. ISBN: 9780262029827. URL: https://mitpress.mit.edu/books/workflow-patterns.

[76] Nandakishore Santhi, Stephan Eidenbenz, and Jason Liu. "The Simian concept: Parallel Discrete Event Simulation with interpreted languages and just-in-time compilation". In: *2015 Winter Simulation Conference (WSC)*. 2015, pp. 3013–3024. DOI: 10.1109/WSC.2015.7408405.

[77] Robert G. Sargent. "Verification and validation of simulation models". In: *Winter Simulation Conference*. WSC '09. Austin, Texas: Winter Simulation Conference, 2009, pp. 162–176. ISBN: 9781424457717.

[78] G. Schmidt and Wilbert E. Wilhelm. "Strategic, tactical and operational decisions in multi-national logistics networks: A review and discussion of modelling issues". In: *International Journal of Production Research* 38.7 (2000), pp. 1501–1523. DOI: 10.1080/002075400188690.

[79] Seleim, A. Azab, and T. AlGeddawy. "Simulation Methods for Changeable Manufacturing". In: *Procedia CIRP* 3 (2012), pp. 179–184. ISSN: 2212-8271. DOI: 10.1016/j.procir.2012.07.032.

[80] Clay Spinuzzi. "The Methodology of Participatory Design". In: *Technical Communication* 52.2 (2005), pp. 163–174. ISSN: 0049-3155. URL: https://www.ingentaconnect.com/content/stc/tc/2005/00000052/00000002/art00005.

[81] Jiachao Tang et al. "A Comprehensive Review of Theories, Methods, and Techniques for Bottleneck Identification and Management in Manufacturing Systems". In: *Applied Sciences* 14.17 (Aug. 2024), p. 7712. ISSN: 2076-3417. DOI: 10.3390/app14177712.

[82] Jan-Frederik Uhlenkamp et al. "Digital Twins: A Maturity Model for Their Classification and Evaluation". In: *IEEE Access* 10 (2022), pp. 69605–69635. DOI: 10.1109/ACCESS.2022.3186353.

[83] Rajesh Verma, Ashu Gupta, and Kawaljeet Singh. "Simulation software evaluation and selection: a comprehensive framework". In: *J. Automation & Systems Engineering* 2 (2008), pp. 221–234.

[84] Christian Weckenborg et al. "Flexibility in manufacturing system design: A review of recent approaches from Operations Research". In: *European Journal of Operational Research* 315.2 (2024), pp. 413–441. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2023.08.050. URL: https://www.sciencedirect.com/science/article/pii/S0377221723006781.

[85] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Berlin Heidelberg, 2024. ISBN: 9783662695180. DOI: 10.1007/978-3-662-69518-0.

[86] WfMC. *BPSim*. 2016. URL: https://www.bpsim.org.

[87] Tommy E White. "Resolution of the Detection of Bottlenecks in Job Shop Enterprises using Discrete-Event Simulation and Lean Six Sigma". In: *Academic Journal of Engineering Studies* 4.1 (May 2025). ISSN: 2694-4421. DOI: 10.31031/aes.2025.04.000576.

[88] Karim Zarour et al. "A systematic literature review on BPMN extensions". In: *Bus. Process. Manag. J.* 26.6 (2020), pp. 1473–1503. DOI: 10.1108/BPMJ-01-2019-0040.

[89] Durk-Jouke van der Zee and Jack G. A. J. van der Vorst. "Guiding principles for conceptual model creation in manufacturing simulation". In: *Proceedings of the Winter Simulation Conference, WSC 2007, Washington, DC, USA, December 9-12, 2007*. Ed. by Shane G. Henderson et al. WSC, 2007, pp. 776–784. DOI: 10.1109/WSC.2007.4419673.

[90] Bernard P. Zeigler and Sankait Vahie. "DEVS formalism and methodology: unity of conception/diversity of application". In: *Proceedings of the 25th conference on Winter simulation - WSC '93*. WSC '93. ACM Press, 1993, pp. 573–579. DOI: 10.1145/256563.256724.

[91] Dmitry Zinoviev. *Discrete Event Simulation: It's Easy with SimPy!* 2024. arXiv: 2405.01562 [cs.MS]. URL: https://arxiv.org/abs/2405.01562.

[92] Sema Zor, David Schumm, and Frank Leymann. "A Proposal of BPMN Extensions for the Manufacturing Domain". In: *Proceedings of the 44th CIRP International Conference on Manufacturing Systems*. 2011.

Finanziato dall'Unione europea – Next Generation EU, Missione 4, Componente 2, Investimento 3.3 (D.M. 117/2023) CUP J33C22001400009