ARQII Publication
http://arqiipubl.com/ams
APPLICATIONS OF MODELLING AND SIMULATION
eISSN 2600-8084
VOL 10, 2026, 184–195

A Model-Driven Transformation Framework for Business Process Simulation and Process-Level KPI Computation

Mariane El Kassis[1], Charbel Kady[1,*], François Trousset[1], Nicolas Daclin[1] and Gregory Zacharewicz[1]

[1]SyCoIA - Systèmes Complexes et Intelligence Artificielle, IMT Mines Alès, 6 Avenue de Clavières, 30100 Alès, France

[*]Corresponding author: charbel.kady@mines-ales.fr

Submitted 17 April 2026; Revised 26 May 2026; Accepted 01 June 2026; Available online 25 June 2026.

Abstract: Business process simulation requires process-level performance indicators that can be computed traceably from parameterized process models. This paper extends a previously defined dual-phase transformation chain from Business Process Model and Notation (BPMN) models enriched with Business Process Simulation Specification (BPSim) parameters to executable Discrete Event System Specification (DEVS) simulation models through an Intermediate Interaction Model (I2M). The extension introduces additional I2M DEVS library components to support exclusive control-flow execution and structured process-level observation, together with four transformation rules linking BPMN constructs and BPSim annotations to the generated simulation model. A Key Performance Indicator (KPI) computation and aggregation methodology is also defined to transform execution-level simulation events into process-level indicators through explicit aggregation rules. The approach organizes indicators into four performance dimensions: time, cost, resource, and quality. To demonstrate the extensibility of the framework, representative indicators including service-level agreement (SLA) compliance and rework-related metrics are implemented without modifying existing transformation rules. The extended framework is applied to a synthetic order-fulfillment process under multiple what-if scenarios involving targeted parameter changes. The results show that the approach supports traceable KPI computation and structured performance analysis under controlled simulation settings.

Keywords: Business process simulation; BPMN; BPSim; DEVS; Interoperability; Key performance indicators; Model-driven engineering; Model transformation; SLA.

1. INTRODUCTION

Process optimization requires the comparison of alternative process configurations against measurable performance criteria [1,2]. When such alternatives cannot be tested directly on operational systems, discrete-event simulation provides a controlled means of evaluating their effects before deployment. For simulation results to support decision-making, however, process-level indicators must remain traceable to the parameters that govern process behavior [3]. Business Process Model and Notation (BPMN) [4] is widely used to model business processes, while Business Process Simulation Specification (BPSim) [5] enriches BPMN models with simulation parameters such as durations, resources, costs, and routing probabilities. However, BPMN models enriched with BPSim parameters are not directly executable. In addition, neither BPMN nor BPSim explicitly defines process-level indicators as modeling constructs. To address these limitations, previous work introduced a dual-phase transformation chain that converts BPMN+BPSim models into executable Discrete Event System Specification (DEVS) simulation models through an Intermediate Interaction Model (I2M) [6–9]. This architecture preserves traceability between declared process parameters and generated simulation artifacts. However, three limitations remain: support for exclusive routing based on BPSim probability declarations is incomplete; Key Performance Indicator (KPI) computation often depends on external post-processing rather than explicit simulation-level observation and aggregation; and existing approaches provide limited support for structuring process-level indicators across multiple performance dimensions. Building on this foundation, this paper extends the existing transformation framework by introducing new I2M DEVS components for exclusive control-flow execution and structured process-level observation, together with a traceable KPI computation and aggregation methodology that derives process-level indicators from execution events through explicit aggregation rules organized across four performance dimensions. The approach is evaluated on a manufacturing order-fulfillment process under one baseline and four what-if scenarios involving controlled parameter changes.

The main contributions are:

- an extension of the I2M DEVS library to support exclusive routing, loop-back tracking, and structured process observation;
- four transformation rules integrating these components into the dual-phase BPMN+BPSim to DEVS chain;

- a traceable KPI computation and aggregation methodology based on explicit execution-event aggregation rules;
- representative implementations of extensible process-level indicators, including service-level agreement (SLA) compliance and rework metrics.

The remainder of the paper is organized as follows. Section 2 reviews the related work on business process simulation, KPI computation, and model-driven transformation approaches. Section 3 presents the proposed methodological extensions, including the transformation rules, library additions, and KPI aggregation principles. Section 4 introduces the case study and describes the experimental design. Section 5 presents the simulation results obtained across the considered scenarios. Section 6 discusses the findings, the computational behavior of the framework, and the main limitations of the present evaluation. Section 7 concludes the paper and outlines future work.

2. RELATED WORK

This section positions the present work with respect to two main research directions: business process simulation and KPI computation on the one hand, and model-driven transformation of BPMN models into executable simulation formalisms on the other. It then introduces the technical basis and specific positioning of the present contribution.

2.1 Business Process Simulation and KPI Computation

Business process simulation uses executable models to estimate process performance under controlled conditions and to compare alternative configurations before implementation in real operations [10, 11]. Commonly observed indicators include cycle time, throughput, waiting time, resource utilization, and cost. More generally, KPIs in business process simulation are commonly organized into four dimensions [10]: time, cost, resource, and quality. Time indicators include cycle time, throughput, and lead time; cost indicators include execution and total process cost; resource indicators include utilization and queue-related measures; and quality indicators include defect, rework, and compliance measures.

Several tools support such analyses. BPMN-oriented environments such as BIMP [12] and Bizagi Modeler [13] provide direct simulation from process models, while general-purpose discrete-event simulation platforms such as Arena [14], AnyLogic [15, 16], and SIMUL8 [17] offer broader modeling flexibility. However, general-purpose platforms often require manual construction or adaptation of simulation models, which weakens traceability between the process specification and the executable model.

A recurring limitation is that KPI sets are usually predefined by the simulation engine, while the aggregation logic used to compute them remains internal to the tool. Standard outputs are therefore readily available, especially for time- and cost-related measures and, to a lesser extent, for resource utilization. By contrast, domain-specific and quality-related indicators often require additional observation mechanisms. Waiting-time decomposition depends on explicit tracking of resource requests and acquisitions. SLA compliance requires per-entity deadline tracking. Rework indicators require loop-sensitive observation of process entities across repeated executions. As a result, such indicators often depend on external post-processing or tool-specific customization rather than on explicit and traceable computation within the simulation model.

Process mining approaches can derive similar indicators from execution logs [2], but they operate on observed process histories rather than on parameterized simulation models. They therefore do not directly support traceable what-if analysis over modified process configurations.

2.2 Model-Driven Transformations for Simulation

Model-Driven Engineering (MDE) aims to generate executable simulation models automatically from process specifications, thereby reducing manual modeling effort and preserving traceability between source and target models. Several target formalisms have been explored. BPMN-to-Petri-Net transformations have been used for verification and performance analysis [18, 19]. These approaches are well suited to control-flow analysis, but they offer more limited support for rich resource behavior and stochastic task execution at process level.

Within the DEVS formalism, several contributions have addressed the mapping of BPMN constructs to executable simulation models. Cetinkaya *et al.* [20] proposed an early BPMN to DEVS mapping, later extended by Bazoun *et al.* [21] for broader BPMN 2.0 coverage. Bocciarelli *et al.* [22] introduced automated transformations with code generation capabilities, while D'Ambrogio and Zacharewicz [23] refined DEVS-based simulation architectures with resource allocation and coordination mechanisms. Alshareef and Sarjoughian [24, 25] further developed hierarchical activity-based specifications for control-flow modeling in P-DEVS.

These works mainly address the structural transformation from process constructs to simulation primitives. By contrast, the question of how process-level KPIs should be observed, computed, and related to the declared BPMN+BPSim parameters is less explicitly treated. In most cases, KPI computation remains a post-simulation concern.

2.3 Technical Basis of the Present Work

The present work builds on a previously defined dual-phase transformation architecture from BPMN models enriched with BPSim parameters to executable DEVS simulation models [6–9]. BPMN provides the structural description of the process, while BPSim complements it with simulation-oriented parameters such as durations, costs, resources, routing probabilities, and scenario-level settings. Since BPMN models enriched with BPSim parameters do not define executable semantics by themselves, an executable simulation formalism is required.

The target formalism used here is Parallel DEVS (P-DEVS) [26], executed in DEVS-Suite [27, 28]. To avoid a direct and monolithic BPMN-to-DEVS transformation, previous work introduced an Intermediate Interaction Model (I2M). The main role of I2M is to provide an intermediate semantic layer that helps resolve semantic ambiguities between BPMN process constructs, simulation annotations, and their executable representation in DEVS. In the first phase, BPMN elements together with their BPSim annotations are transformed into I2M structures through Xtend-based mapping rules guided by Generic Interaction Models (GIMs). In the second phase, I2M elements are translated into executable P-DEVS models through a dedicated library while preserving the same interaction topology. Two properties of this architecture are particularly important for the present paper. First, the I2M DEVS library is modular, so new execution components can be added without altering existing ones. Second, the transformation rules operate on the GIM-based intermediate structures, which supports extension while preserving previously validated mappings.

Before the extensions proposed here, the framework already supported time-, cost-, and resource-related indicators. Compared with the prior versions in [6–9], which focused on establishing the BPMN+BPSim-to-DEVS dual-phase transformation and on its application to distributed and collaborative settings, the present paper introduces three distinct contributions that did not exist in any of the earlier works: explicit support for probability-based exclusive routing in the I2M DEVS library, a process-level observation component coupled to the process terminal event, and a structured KPI aggregation methodology with rules attached to declared BPSim parameters. In particular, exclusive routing based on BPSim probability declarations was not directly supported, and process-level KPI aggregation relied on ad hoc post-processing rather than on a dedicated observation mechanism. These limitations motivate the methodological and structural extensions introduced in the next section.

2.4 Positioning of the Present Work

The present work extends this line of research in two complementary ways. First, it integrates KPI observation and aggregation within the simulation framework itself through an explicit observation and aggregation mechanism based on defined aggregation rules, rather than treating KPI computation as a purely external analysis step. Second, it extends the I2M DEVS library with components for exclusive routing and process-level observation, and defines explicit transformation rules linking BPMN constructs and BPSim annotations to these new components. This supports extensible KPI definition through additive extensions while preserving previously validated rules.

3. METHODOLOGY

This section presents the methodological extensions introduced into the BPMN+BPSim to I2M to DEVS transformation framework. These extensions pursue two complementary objectives: enriching the transformation chain with additional execution semantics through extensions of the I2M metamodel and its associated DEVS library, and formalizing a traceable methodology for computing and aggregating process-level performance indicators directly from simulation execution traces.

As illustrated in Figure 1, the extended workflow follows a structured sequence of modeling, transformation, execution, and analysis stages. The starting point is a BPMN model enriched with BPSim parameters capturing the simulation-related information required to represent process behavior. In the first transformation phase, this enriched BPMN model is transformed into an I2M through the selection and instantiation of GIMs. These GIMs define the internal structure of each process element and specify how simulation-relevant behaviors, including resource allocation, execution duration, routing logic, and observation points, are represented.

The resulting I2M model is then extended with the additional elements introduced in this work, notably components supporting exclusive routing behavior and structured process-level observation. In the second transformation phase, the enriched I2M structure is translated into executable Parallel DEVS (P-DEVS) models through mappings to predefined elements of the I2M DEVS library, and the generated simulation model is executed in the DEVS-Suite environment.

During execution, simulation components generate observation-level data through dedicated observation mechanisms. These execution traces are processed using explicitly defined aggregation rules to derive process-level performance indicators. Because each aggregation rule is associated with specific simulation mechanisms and parameter sources, the resulting indicators remain traceable to their originating BPSim definitions.

3.1 Performance Dimensions and Traceability Structure

Consistent with classifications commonly used in the business process management literature [10], this work organizes process-level indicators into four performance dimensions: time for process duration and completion dynamics, cost for monetary execution impact, resource for resource usage and contention, and quality for conformance, rework, and service objectives.

Rather than treating KPI computation as an external post-simulation activity, the proposed framework integrates it directly into the simulation architecture. During execution, relevant events and local measurements are captured by simulation components and exposed for process-level observation. Process-level KPIs are then derived by applying explicit aggregation rules to the collected execution data.

This methodology relies on a three-level traceability structure:

1. **BPSim parameter**, representing the declared simulation input that influences behavior;
2. **DEVS observation mechanism**, representing the execution component that records or exposes the relevant data;
3. **Aggregation rule**, representing the transformation of the collected execution data into a process-level indicator.

This structure makes it possible to relate each reported indicator both to its simulation source and to its aggregation logic.

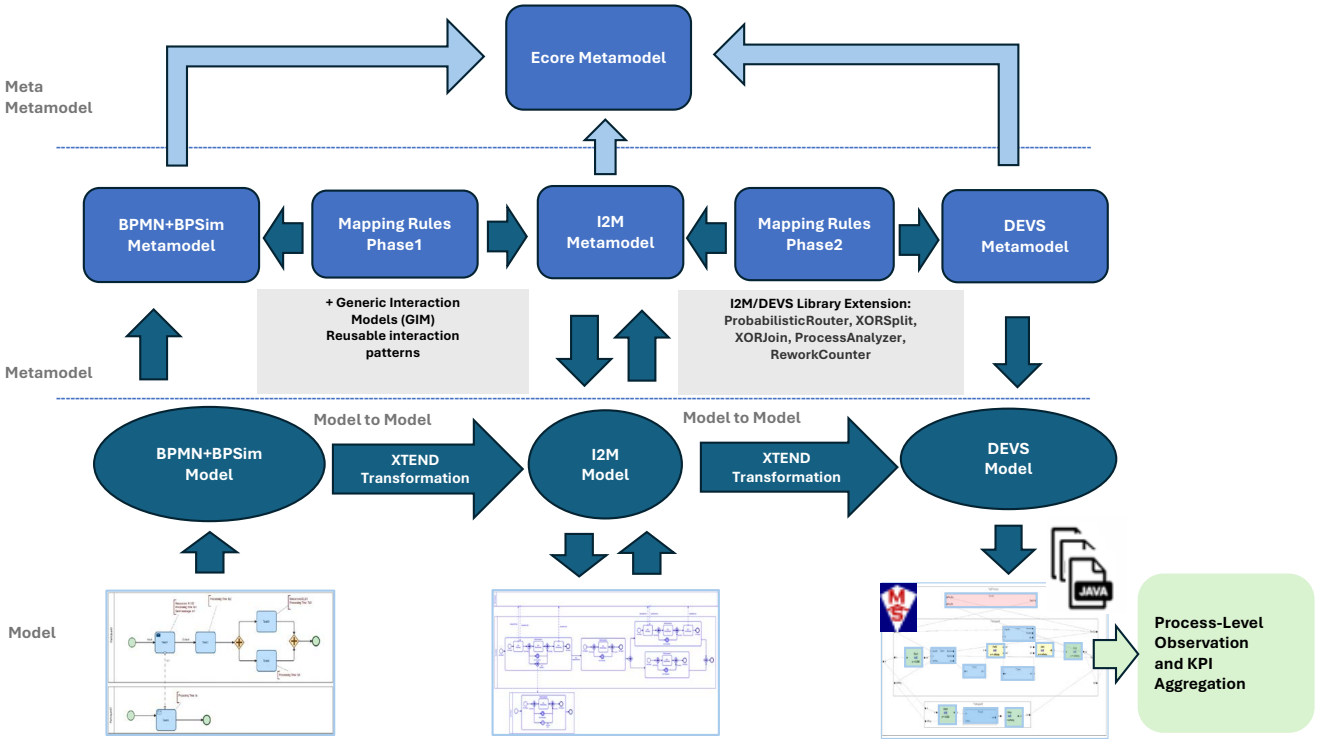

Figure 1. Extended methodological workflow from BPMN+BPSim enrichment to KPI computation and aggregation.

It also makes explicit that different indicators require different aggregation strategies. For example, cycle-time indicators are derived from completed entities, utilization is computed from accumulated service times and resource capacities, SLA compliance is obtained as the proportion of completed entities satisfying a deadline, and rework-related indicators are derived from loop counters carried by entities.

Because all indicators are derived from the same simulation trace, the framework also supports interpretation across performance dimensions. A modification applied to one parameter category may primarily affect one KPI dimension while producing secondary effects in another. For instance, a reduction in resources may primarily affect resource-related behavior while also increasing deadline violations and therefore influencing quality-related indicators. The methodology is intentionally extensible. It is not restricted to a fixed set of predefined indicators. Instead, new KPIs can be incorporated by defining the required observation points and aggregation rules while preserving the existing transformation structure.

3.2 Transformation Rules and Library Extensions

The extended methodology builds upon the dual-phase transformation architecture introduced in Section 2. In Phase 1, BPMN models enriched with BPSim parameters are transformed into I2M structures through GIM-guided Xtend mapping rules. In Phase 2, each generated I2M element is mapped one-to-one to a P-DEVS atomic model from the I2M DEVS library and assembled into an executable coupled model.

This work extends the methodology in both phases. In Phase 1, four new transformation rules generate additional I2M elements from specific BPMN constructs and BPSim annotations. In Phase 2, the I2M DEVS library is extended with the corresponding components required to execute these elements. Table 1 summarizes the Phase 1 rules. Each rule specifies the BPMN source element, the triggering condition, and the generated I2M element. In Phase 2, each generated I2M element is mapped to its corresponding P-DEVS atomic model following the existing library convention.

Table 1. Phase 1 transformation rules for the new I2M elements introduced in this work.

Rule	BPMN source	Triggering condition	Generated I2M element
R1	ExclusiveGateway (diverging)	`ControlParameters.Probability` declared on outgoing SequenceFlows	ProbabilisticRouter
R2	ExclusiveGateway (converging)	Multiple incoming SequenceFlows, single outgoing SequenceFlow	XORJoin
R3	SequenceFlow (feedback path)	`PropertyParameters` with `LoopCounter` attribute declared on the SequenceFlow	ReworkCounter
R4	EndEvent (process-level)	Terminal event of the process	Stop + ProcessAnalyzer

The following paragraphs describe Rules R1 to R4 and the behavior of their corresponding DEVS components.

R1: Probability-Based Exclusive Routing: In Phase 1, Rule R1 applies to a diverging BPMN ExclusiveGateway whose outgoing SequenceFlows carry `ControlParameters.Probability`. The rule generates an `i2m:ProbabilisticRouter` parameterized with the declared probability vector. In Phase 2, the corresponding DEVS component, **ProbabilisticRouter**, receives an incoming entity, requests a random value from the shared random stream, and forwards the entity to exactly one of N outgoing branches according to the cumulative probability intervals derived from $p_1, \ldots, p_N$. This preserves the probability-based exclusive gateway semantics defined at the BPMN level during DEVS execution.

R2: Exclusive Merge: In Phase 1, Rule R2 applies to a converging BPMN ExclusiveGateway with multiple incoming SequenceFlows and a single outgoing SequenceFlow. The rule generates an `i2m:XORJoin` with one input port per incoming flow. In Phase 2, the resulting DEVS component, **XORJoin**, accepts an entity arriving from any input branch and immediately forwards it to its single output without synchronization. This behavior is distinct from the parallel Join used inside the TaskContainer GIM, which waits for all branches before releasing the entity.

R3: Loop-Back Tracking: In Phase 1, Rule R3 applies to a SequenceFlow annotated with the BPSim attribute `LoopCounter`, declared through `PropertyParameters`. The rule generates an `i2m:ReworkCounter` on that flow. The counter-type identifier, for example `PRODUCTION` or `MATERIAL`, is derived from the value of the `LoopCounter` attribute.

This rule relies on the BPSim extension mechanism, which allows user-defined properties to be attached to annotated model elements, including SequenceFlows. By making loop tracking annotation-driven rather than structure-driven, the modeler retains explicit control over which feedback paths are instrumented.

In Phase 2, the resulting DEVS component, **ReworkCounter**, increments an integer counter carried by the passing *ProcessEntity* and forwards the entity without introducing simulation delay. Each entity therefore accumulates the number of times it has traversed each instrumented feedback path during its execution lifecycle.

R4: Process-Level Observation: In Phase 1, Rule R4 applies to the process-level terminal EndEvent. The rule generates both an `i2m:Stop` and an `i2m:ProcessAnalyzer` coupled to it. In Phase 2, the `ProcessAnalyzer` is parameterized with references to all `GetRes` elements present in the I2M model. This requires a collection pass over the I2M structure, which is feasible since the full I2M XMI is available at that stage.

The resulting DEVS component, **ProcessAnalyzer**, receives each completed *ProcessEntity* and applies KPI-specific aggregation rules to compute process-level indicators. These include cycle-time statistics, throughput, SLA compliance, and rework-related metrics. The **ProcessAnalyzer** also accesses execution data recorded in the *GlobalParameters* singleton during simulation, such as accumulated service times and cost values, which are required for resource utilization and cost reporting. Accordingly, entity-level indicators are derived from completed *ProcessEntity* instances received by the **ProcessAnalyzer**, whereas cost- and resource-related indicators additionally rely on execution data exposed through observation components and recorded in *GlobalParameters*.

In addition to the components produced by Rules R1 to R4, the I2M DEVS library includes the **XORSplit**, which supports attribute-based exclusive routing. Rather than selecting an outgoing branch from predefined probabilities, it routes an entity according to the value of a numerical attribute relative to a threshold. This component is not triggered by the case study presented in this paper, which uses only probability-based routing at exclusive gateways. It is retained in the library as a general-purpose mechanism for processes in which routing decisions depend on runtime entity properties.

The transformation rules described above require the source BPMN model to satisfy two structural conditions. First, exclusive merge points must be represented as explicit converging ExclusiveGateways rather than as implicit multi-incoming flows into a task. This ensures that Rule R2 is triggered by a well-defined BPMN element. Second, feedback SequenceFlows that require loop counting must carry a `LoopCounter` attribute declared through `PropertyParameters`. Feedback paths without this annotation are treated as regular SequenceFlows and do not produce a `ReworkCounter`. These conditions do not restrict the expressiveness of the BPMN model. They reflect standard modeling practices in which control-flow semantics are made explicit at the source level rather than inferred during transformation.

3.3 KPI Computation and Aggregation Methodology

The computation of process-level performance indicators in the proposed framework is based on a structured observation and aggregation methodology integrated directly into the simulation execution architecture. Rather than computing indicators through external post-processing, KPI values are derived from execution-level observations using explicitly defined aggregation rules associated with simulation components.

The objective of this methodology is not to impose a fixed set of predefined indicators, but rather to provide a systematic mechanism through which new KPIs can be defined, computed, and aggregated while preserving traceability to their originating process definitions. To preserve interpretability, each indicator is linked to the declared simulation inputs that influence behavior, to the observation mechanisms that expose the relevant execution data, and to the aggregation rule used to derive the final process-level value.

This structure ensures that each reported KPI remains explicitly linked both to its originating process definitions and to the simulation mechanisms through which it is computed. It also clarifies that different indicators may require distinct aggregation strategies depending on their semantic interpretation.

The proposed methodology supports multiple aggregation methods depending on the nature of the observed data. Rather than applying a single uniform aggregation mechanism, indicator computation is adapted to the semantic requirements of each KPI. Three representative aggregation families are supported:

- **Proportion-based aggregation**, in which the indicator value represents the proportion of entities satisfying a given condition relative to the total number of completed entities;
- **Count-based aggregation**, in which the indicator value represents the number of occurrences of specific events accumulated during execution;
- **Average-based aggregation**, in which the indicator value represents the mean of entity-level measurements collected during execution.

These aggregation families provide a generic foundation that can be adapted to a wide variety of performance indicators without modifying the underlying transformation architecture.

During simulation execution, observation data is collected through dedicated DEVS components associated with relevant process events. Local measurements generated by individual components are then made available to the *ProcessAnalyzer*, which is responsible for computing final process-level indicators. It receives completed process entities and applies predefined aggregation rules to compute final KPI values. Because aggregation is performed within the simulation architecture rather than through external post-processing, the methodology preserves consistency between simulation behavior and reported performance outcomes.

A fundamental design objective of the proposed methodology is extensibility. The framework does not restrict the analysis to a predefined set of indicators. Instead, new KPIs can be introduced through additive extensions that define:

- the required observation points within the simulation structure;
- the aggregation rule used to compute the final indicator value;
- the performance dimension associated with the new indicator.

This extensibility ensures that the methodology remains adaptable to domain-specific requirements while preserving compatibility with previously defined indicators. To demonstrate it concretely, representative KPIs are implemented in the remainder of this work, namely SLA compliance (illustrating proportion-based aggregation) and rework-related indicators (illustrating count-based and average-based aggregation). These representative KPIs serve as methodological instances showing how aggregation logic can be defined and integrated into the simulation framework; additional indicators can be incorporated using the same methodological structure.

3.4 Rework Metrics

Rework-related performance indicators quantify the frequency and intensity of corrective loops occurring during process execution. In manufacturing and service processes, rework represents repeated execution of tasks required to correct defects or incomplete outcomes. Such behaviors directly affect process efficiency, resource usage, and quality performance.

Two complementary indicators are defined to capture rework behavior:

- the **rework rate**, representing the proportion of completed entities that experienced at least one rework traversal;
- the **mean production rework count**, representing the average number of rework traversals per completed entity.

Let N denote the total number of completed entities, and let N_r denote the number of completed entities that traversed at least one rework loop. The rework rate is defined as:

$$R_{\text{rework}} = \frac{N_r}{N} \tag{1}$$

Let r_i denote the number of rework traversals associated with completed entity i. The mean production rework count is defined as:

$$\bar{r}_{\text{all}} = \frac{\sum_{i=1}^{N} r_i}{N} \tag{2}$$

These two indicators capture complementary perspectives on rework behavior: the first measures occurrence frequency, whereas the second reflects repetition intensity.

Although the methodology supports loop tracking on any instrumented feedback path, the indicators reported in this paper focus on production rework. The material replenishment loop is also instrumented through a separate `LoopCounter` annotation but is not included in the reported metrics, as production rework is selected as the representative quality-oriented indicator for this case study.

BPSim Source and Routing Dependencies: Rework behavior originates from loop-back structures governed by exclusive gateway routing probabilities defined within BPSim `ControlParameters`. The feedback SequenceFlows associated with these structures carry `PropertyParameters.LoopCounter` annotations, which trigger the generation of ReworkCounter components during Phase 1 transformation (rule R3 in Table 1).

In the considered process model, rework occurs when inspection results fail to satisfy quality requirements. The probability assigned to the rework branch determines how frequently entities are redirected toward corrective activities. These routing probabilities therefore constitute the primary parameter source influencing rework-related behavior. Variations in their values directly affect the number of loop traversals observed during simulation.

Observation Mechanism: Rework-related observations are captured by a dedicated atomic component, the *ReworkCounter*, positioned along loop-back SequenceFlows associated with corrective execution paths. Each time a *ProcessEntity* traverses such a path, the counter carried by the entity is incremented before the entity is forwarded to the next execution stage. In this way,

individual rework occurrences are recorded at the entity level, repeated loop traversals are accumulated as integer counts, and rework information remains attached to the entity throughout its execution lifecycle.

Aggregation Rule: At process completion, the *ProcessAnalyzer* retrieves the rework counter associated with each completed entity and derives two indicators from it. The rework rate is computed as the proportion of entities whose rework counter is greater than zero, while the mean production rework count is computed by summing all recorded rework counts and dividing by the total number of completed entities. These aggregation rules show how different indicators can be derived from the same observation mechanism through distinct aggregation strategies. They also demonstrate that event occurrences can be recorded incrementally during execution, that entity-level counters can capture repeated behavior, and that process-level metrics can be produced through explicit aggregation of accumulated values. Together with SLA compliance, rework-related indicators illustrate complementary aggregation strategies for structured performance analysis across multiple dimensions.

3.5 SLA Compliance

Service-Level Agreement (SLA) compliance measures the proportion of completed process instances whose total cycle time satisfies a predefined execution deadline. This indicator evaluates whether process execution meets time-related service objectives and is therefore classified within the quality dimension of the proposed performance structure.

Let N denote the total number of completed process instances and c_i the cycle time associated with the i-th completed entity. Let D denote the SLA deadline defined at the scenario level. The SLA compliance rate is computed as:

$$R_{\text{SLA}} = \frac{|\{i \mid c_i \leq D\}|}{N} \tag{3}$$

This formulation represents a proportion-based aggregation method in which each completed entity contributes a binary evaluation depending on whether the deadline constraint is satisfied.

BPSim Source and Parameter Mapping : The deadline value D is introduced as a scenario-level parameter within the BPSim configuration. Unlike task-level parameters such as processing time or resource allocation, this parameter represents a global performance constraint applied uniformly across all process instances. During simulation initialization, the deadline value is assigned to each generated *ProcessEntity* as an attribute. This ensures that deadline verification can be performed independently for each entity at the completion stage.

Observation Mechanism and Aggregation Rule: The observation mechanism associated with SLA computation is triggered when a *ProcessEntity* reaches the process termination point. At this stage, the *ProcessAnalyzer* computes the total cycle time of the entity by subtracting its recorded start time from its completion time. The computed cycle time is then compared with the declared deadline value. The result of this comparison is stored as a binary indicator associated with the completed entity:

- value 1 if $c_i \leq D$ (deadline satisfied);
- value 0 if $c_i > D$ (deadline violated).

These binary values constitute the intermediate observations used for aggregation.

Aggregation Rule: The final SLA compliance value is obtained through proportion-based aggregation applied to the set of completed entities. This aggregation is performed at the end of simulation execution, ensuring that the reported indicator reflects the complete population of processed entities rather than a partial sample. SLA compliance serves as a representative example of a proportion-based KPI derived from entity-level observations. Its implementation demonstrates that a scenario-level parameter can define a global performance objective, that entity-level measurements can be evaluated individually, and that process-level indicators can be computed through explicit aggregation logic integrated into the simulation framework. Although SLA compliance is used here as an illustrative example, the same aggregation structure can be applied to other condition-based indicators without modifying the transformation workflow.

4. CASE STUDY

This section presents the manufacturing order-fulfillment process used to evaluate the extended transformation chain and KPI computation methodology. The objective is methodological demonstration rather than operational calibration. One baseline scenario (S0) and four what-if scenarios (S1–S4) are defined to evaluate the effect of controlled parameter variations.

The case study considers a manufacturing order-fulfillment workflow containing seven operational tasks, four exclusive gateways, and two loop-back paths, illustrated in Figure 2. The process begins with order reception and validation and ends with order dispatch after successful production and inspection. The forward flow consists of order validation, material checking, production scheduling, production execution, inspection, and dispatch. If materials are unavailable, the process enters a replenishment loop. If inspection fails, the produced item is redirected to production through a rework loop. These routing mechanisms enable evaluation of both time-related and quality-related indicators.

The BPMN model is enriched with BPSim parameters defining processing times, resource usage, costs, and routing probabilities. Task-level parameters are summarized in Table 2. All processing times follow triangular distributions. Task T4 has the longest execution time and highest cost, making it the dominant production activity. Resource capacities introduce potential waiting times when demand exceeds availability. Global simulation settings are identical across scenarios unless modified: time unit = h, cost unit = EUR, 30 independent replications, seed range = 123456–123485, 1000 entities per replication, fixed inter-arrival time = 1.0 h, and baseline SLA deadline = 12.0 h.

Five scenarios are defined using a one-factor-at-a-time design to isolate the effect of individual parameter changes. Table 3 summarizes the modifications relative to the baseline configuration. Scenario S1 reduces production capacity by decreasing

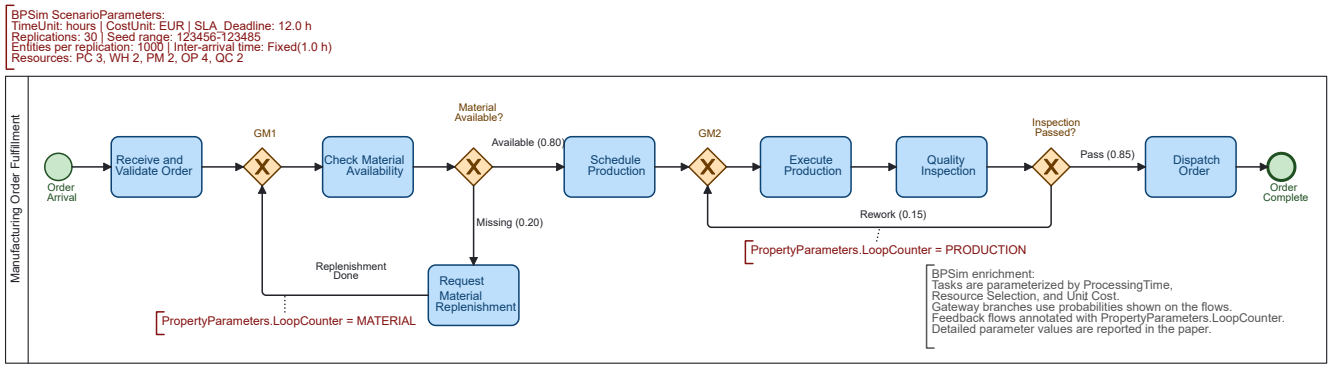

Figure 2. BPMN model of the manufacturing order-fulfillment process with scenario-level BPSim enrichment. Task-level parameters are summarized in Table 2.

Table 2. BPSim parameters for the order-fulfillment process. Processing times are in hours; costs are in EUR per invocation. $\mathrm{Tri}(a,b,c)$ denotes a triangular distribution with minimum a, mode b, and maximum c.

Element	**Processing time**	**Resource**	**Pool size**	**Cost**
T1 Receive/Validate	Tri(0.1,0.2,0.4)	PC	3	10
T2 Check Material	Tri(0.05,0.1,0.2)	WH	2	5
T3 Schedule Production	Tri(0.2,0.4,0.8)	PM	2	15
T4 Execute Production	Tri(1.0,2.0,4.0)	OP	4	50
T5 Quality Inspect.	Tri(0.3,0.5,1.0)	QC	2	20
T6 Material Replenish.	Tri(0.2,0.4,0.8)	WH	2	8
T7 Dispatch Order	Tri(0.1,0.2,0.3)	WH	2	8
G1 Material Available	$P = 0.80/0.20$			
G2 Inspection Passed	$P = 0.85/0.15$			

Table 3. Scenario definitions relative to the baseline configuration.

Scenario	Modified parameter	Baseline	Modified	Category
S0	—	—	—	Baseline
S1	OP pool size	4	3	Resource
S2	T4 duration	Tri(1,2,4)	Tri(0.5,2,6)	Time
S3	SLA deadline	12 h	8 h	Scenario
S4	G2 probability	0.85/0.15	0.70/0.30	Control

the operator pool associated with T4. Scenario S2 increases duration variability in the same task. Scenario S3 tightens the SLA deadline without modifying execution dynamics. Scenario S4 increases inspection failure probability, resulting in higher expected rework frequency. All scenarios use identical global settings, ensuring consistent comparison across runs.

5. SIMULATION RESULTS

All scenarios summarized in Table 3 were executed using the DEVS-Suite simulation environment generated from the extended BPMN+BPSim to I2M to DEVS transformation chain. Each scenario was executed over $R = 30$ independent replications, indexed by integer seeds in the range $[123456, 123485]$, while all other simulation parameters were kept identical across replications and across scenarios. The number of replications follows standard discrete-event simulation practice [29], in which multiple replications combined with confidence intervals are used to account for stochastic variability.

For each KPI, the reported value is the sample mean over the 30 replications, and the reported half-width corresponds to the 95% confidence interval around that mean. Each simulation generated 1000 process entities using a fixed inter-arrival time of 1.0 h. Because simulation execution was bounded by a predefined time horizon, a small number of entities remained in transit at the end of each run. The number of completed entities observed in each scenario was constant across replications and is summarized as follows:

- S0 (Baseline): 995 completed entities
- S1 (Resource reduction): 996 completed entities
- S2 (Increased duration variability): 994 completed entities
- S3 (SLA tightening): 995 completed entities
- S4 (Reduced inspection pass rate): 995 completed entities

The small variation in completed entity counts between scenarios reflects expected dynamics associated with time-bounded simulation horizons rather than instability in the model. In all scenarios, more than 99% of generated entities completed execution, indicating consistent simulation behavior. All scenarios preserved identical baseline settings except for the targeted modifications summarized in Table 3, ensuring that observed differences can be attributed to the intended parameter changes.

Resource contention in this framework is managed through a centralized Broker model that serializes all resource requests through a single FIFO queue. This design introduces head-of-line blocking effects that increase absolute cycle-time values, while preserving meaningful relative differences between scenarios for comparative analysis.

Because S0 and S3 differ only in the SLA deadline, which is a scenario-level parameter consumed at the analyzer stage and not during execution, all execution-level KPIs (time, cost, resource, and rework indicators) are expected to be identical for these two scenarios. The 30-replication experiments confirm this property exactly: across all 30 paired seeds, S0 and S3 produce identical values for every execution-level KPI, and the only difference between them concerns the SLA compliance rate.

5.1 Time-Dimension Results

Cycle-time statistics and throughput values were computed from the completed entities observed in each scenario. These indicators were obtained through average-based aggregation of the per-entity cycle times recorded by the *ProcessAnalyzer*, while throughput was derived from the number of completed entities. Table 4 summarizes the resulting time-dimension KPIs across all scenarios.

Table 4. Time-dimension KPI results across scenarios. Values are means over 30 replications; ± values denote 95% confidence interval half-widths.

KPI	S0	S1	S2	S3	S4
Mean cycle time (h)	870.7 ± 9.8	874.6 ± 10.4	1161.5 ± 13.0	870.7 ± 9.8	1159.6 ± 15.1
Median cycle time (h)	831.8 ± 10.4	836.0 ± 10.5	1116.3 ± 13.2	831.8 ± 10.4	1060.0 ± 17.6
P95 cycle time (h)	1916.9 ± 19.4	1923.7 ± 20.2	2486.8 ± 24.8	1916.9 ± 19.4	2511.0 ± 30.4
Stdev cycle time (h)	549.8 ± 6.7	550.9 ± 6.9	721.8 ± 8.5	549.8 ± 6.7	760.7 ± 8.0
Throughput (ent/h)	0.364 ± 0.002	0.364 ± 0.002	0.300 ± 0.002	0.364 ± 0.002	0.299 ± 0.002

Scenario S3 produces time-dimension results identical to S0, confirming that modification of the SLA deadline does not influence execution dynamics. Scenario S1 produces a slight increase in mean cycle time relative to the baseline, but the corresponding 95% confidence intervals overlap, indicating that the effect of the OP pool reduction on cycle time is within the stochastic variability observed across replications; the operational impact of this reduction is instead captured by the marked increase in OP utilization reported in Table 7. Scenarios S2 and S4 produce clearly separated increases in cycle-time indicators (non-overlapping confidence intervals with S0), reflecting increased duration variability and additional rework loops respectively. These results illustrate how operational and quality-related changes propagate into time-based indicators and how the use of confidence intervals helps distinguish meaningful effects from stochastic fluctuation.

5.2 Cost-Dimension Results

Cost indicators are derived from the per-invocation unit costs declared in BPSim CostParameters and accumulated during execution by the ConstantCost components within each TaskContainer. Table 5 presents the direct cost and per-entity cost values for each scenario.

Table 5. Cost-dimension KPI results across scenarios. Values are means over 30 replications; ± values denote 95% confidence interval half-widths.

KPI	S0	S1	S2	S3	S4
C_{direct} (EUR)	$123\,308 \pm 473$	$123\,406 \pm 497$	$123\,117 \pm 491$	$123\,308 \pm 473$	$140\,976 \pm 548$
$C_{\text{per entity}}$ (EUR)	123.93 ± 0.48	123.90 ± 0.50	123.86 ± 0.49	123.93 ± 0.48	141.68 ± 0.55

Scenarios S0 and S3 produce identical cost values, confirming independence between SLA thresholds and execution-related costs. Scenarios S1 and S2 remain within the confidence interval of the baseline, indicating that the targeted resource and duration changes do not significantly affect total cost. Scenario S4 produces a substantial and clearly separated increase associated with additional task invocations caused by higher rework frequency. Table 6 provides the per-element cost breakdown across scenarios.

5.3 Resource-Dimension Results

Resource utilization is computed as the ratio of accumulated service time to available pool capacity. Table 7 presents utilization values for each resource pool.

Scenario S1 produces a clearly separated utilization increase in the OP resource pool (from 24.97% to 33.30%, with non-overlapping confidence intervals), consistent with the reduction in available units. Scenario S2 reduces utilization across most pools, reflecting the wider duration distribution and the consequent change in instantaneous load. Scenario S4 shows utilization values close to baseline for most pools, with the additional rework workload absorbed primarily by the OP and QC capacity slack. Identical values between S0 and S3 confirm that SLA constraints do not affect execution-level resource usage.

Table 6. Per-element cost breakdown across scenarios (EUR), obtained from a representative replication (seed 123456). Aggregate cost values in Table 5 are based on 30 replications and are consistent with this representative breakdown.

Element	**S0**	**S1**	**S2**	**S3**	**S4**
T1 Receive/Validate	10 000	10 000	10 000	10 000	10 000
T2 Check Material	6 210	6 255	6 195	6 210	6 285
T3 Schedule Prod.	14 970	14 970	14 970	14 970	14 970
T4 Execute Prod.	59 200	59 300	58 800	59 200	70 050
T5 Quality Inspect.	23 680	23 720	23 500	23 680	28 020
T6 Mat. Replenishment	1 944	2 016	1 920	1 944	2 064
T7 Dispatch Order	7 960	7 968	7 952	7 960	7 960

Table 7. Resource utilization (%) across scenarios. Values are means over 30 replications; ± values denote 95% confidence interval half-widths.

Resource pool	**S0**	**S1**	**S2**	**S3**	**S4**
PC (3 units)	2.85 ± 0.02	2.84 ± 0.02	2.35 ± 0.02	2.85 ± 0.02	2.34 ± 0.02
WH (2 units)	8.41 ± 0.07	8.41 ± 0.08	6.95 ± 0.07	8.41 ± 0.07	6.93 ± 0.07
PM (2 units)	8.49 ± 0.06	8.49 ± 0.06	7.03 ± 0.05	8.49 ± 0.06	7.00 ± 0.06
OP (4/3 units)	$24.97 \pm < 0.01$	$33.30 \pm < 0.01$	$24.98 \pm < 0.01$	$24.97 \pm < 0.01$	$24.98 \pm < 0.01$
QC (2 units)	12.86 ± 0.06	12.86 ± 0.06	10.57 ± 0.08	12.86 ± 0.06	12.83 ± 0.06

5.4 SLA Compliance Results

This subsection presents the results obtained for the SLA compliance indicator introduced in Section 3.5. Table 8 summarizes the observed SLA compliance values across scenarios.

Table 8. SLA compliance results across scenarios. Values are means over 30 replications; ± values denote 95% confidence interval half-widths.

Scenario	SLA deadline	SLA compliance rate (%)
S0 (Baseline)	12.0 h	0.707 ± 0.060
S1 (Resource reduction)	12.0 h	0.676 ± 0.055
S2 (Increased variability)	12.0 h	0.537 ± 0.064
S3 (SLA tightening)	8.0 h	0.459 ± 0.049
S4 (Reduced pass rate)	12.0 h	0.580 ± 0.066

Compliance values remain low across all scenarios due to the strict deadline relative to the observed cycle-time distribution. Scenario S3 demonstrates that tightening the deadline directly affects compliance without modifying execution dynamics. Variations between the remaining scenarios reflect the influence of operational changes combined with stochastic variability; the relatively wider confidence intervals for this indicator are a direct consequence of the small absolute values involved.

5.5 Rework Metrics Results

This subsection presents the results obtained for the rework-related indicators introduced in Section 3.4. Table 9 summarizes the observed rework-related values.

Scenarios S1 and S2 produce values whose confidence intervals overlap with the baseline, confirming that rework behavior is primarily governed by routing probabilities rather than by upstream resource or duration parameters. Scenario S3 reproduces the baseline values exactly, confirming independence between SLA thresholds and rework behavior. Scenario S4 produces a large and clearly separated increase (non-overlapping confidence interval with S0), reflecting the direct impact of the modified inspection-pass probability.

5.6 Cross-Dimensional Analysis and Consistency Checks

Cross-dimensional comparison confirms consistent relationships between parameter changes and observed KPI behavior. Scenarios sharing identical operational settings produce identical execution-level indicators, while targeted parameter modifications generate changes confined to the expected dimensions. Direct routing modifications primarily affect rework behavior, whereas resource and duration changes influence time and utilization indicators. These observations collectively support the internal consistency and traceability of the aggregation methodology across all evaluated performance dimensions.

6. DISCUSSION

The experimental results indicate that the proposed aggregation framework produces indicators consistent with expected process behavior. The observed dimensional independence between operational parameters and evaluation thresholds, the alignment between measured values and theoretical expectations, and the proportional response of resource utilization to capacity changes together support the correctness of the aggregation mechanisms.

Table 9. Rework-related KPI results across scenarios. Values are means over 30 replications; ± values denote 95% confidence interval half-widths.

Scenario	Rework rate (%)	Mean rework count
S0 (Baseline)	15.18 ± 0.56	0.1773 ± 0.0066
S1 (Resource reduction)	15.18 ± 0.54	0.1780 ± 0.0070
S2 (Duration variability)	14.92 ± 0.48	0.1743 ± 0.0066
S3 (SLA tightening)	15.18 ± 0.56	0.1773 ± 0.0066
S4 (Reduced pass rate)	30.12 ± 0.53	0.4303 ± 0.0080

From a methodological perspective, embedding aggregation logic directly within the transformation workflow enables systematic traceability between process structure and performance measurement. The one-parameter-per-scenario design supports clear association between KPI variations and individual model components, while the modular structure of aggregation rules enables extensibility without requiring modification of the core transformation architecture.

The framework also demonstrates practical relevance for industrial process modeling. The ability to relate KPI values to explicit parameter changes supports informed decision-making and enables structured evaluation of trade-offs between operational efficiency and quality requirements. The observed sensitivity of quality-related metrics to routing probabilities and of deadline-based indicators to operational conditions illustrates how the framework captures cross-dimensional process effects within a unified simulation model.

Several limitations should be acknowledged. First, the case study is based on a single synthetic process, and additional applications across different domains would strengthen generalizability. Second, the experiments use deterministic inter-arrival patterns and a fixed number of entities; more realistic arrival processes could provide additional insight into system dynamics. Third, while the framework supports a wide range of indicators, the present evaluation focuses on representative KPI categories. Fourth, the centralized Broker model serializes all resource requests through a single FIFO queue, which amplifies queueing delays and produces absolute cycle times that are higher than typical industrial observations. Although relative differences between scenarios remain meaningful, alternative resource allocation strategies would improve realism.

Concerning computational behavior, the full 30-replication batch across the five scenarios (150 simulation runs in total) executed in approximately 52.6 minutes on a standard workstation, with a mean wall-clock time per run of 21.04 ± 0.24 seconds (95% confidence interval). The wall-clock cost is dominated by the simulation itself rather than by the I2M layer: the dual-phase transformation chain produces I2M and DEVS artifacts at compile time, and each I2M element is mapped one-to-one to a P-DEVS atomic component used directly by the simulator. Consequently, the I2M layer does not introduce a runtime indirection: the number of atomic models grows linearly with the number of tasks, gateways, and instrumented flows in the BPMN source, so simulation cost scales with the size of the generated DEVS coupled model in the same way as a hand-written DEVS model of equivalent structural size. For substantially larger processes containing hundreds of tasks, the dominant cost would arise from the simulator and the resource broker rather than from the I2M abstraction. Empirical evaluation of the framework on larger industrial processes is part of ongoing work.

Overall, the modular structure of the aggregation rules suggests that the framework can be extended to support additional indicators, such as waiting-time distributions, cost-efficiency measures, and reliability-related metrics, across a wide range of process modeling contexts.

7. CONCLUSION

This paper presented a structured framework for KPI aggregation within BPMN+BPSim-based simulation workflows using a dual-phase model transformation approach. The proposed methodology integrates KPI observation and aggregation mechanisms into the transformation chain from BPMN+BPSim models to executable DEVS simulations through the intermediate interaction model (I2M). Four transformation rules were introduced to extend the existing mapping with support for exclusive routing, feedback flows, and process-level observation, without modifying previously validated transformations.

A central contribution of this work is the formalization of KPI aggregation rules as explicit model-level constructs linked to process execution events. Embedding aggregation logic within the transformation workflow ensures traceability between parameter-level configurations and process-level performance indicators, supporting consistent interpretation of simulation outcomes under controlled parameter variations.

The methodology was evaluated using a manufacturing order-fulfillment process executed across five scenarios involving controlled parameter modifications affecting resource capacity, processing-time variability, service-level constraints, and routing probabilities. The experiments demonstrated stable and interpretable behavior across multiple KPI dimensions and confirmed the applicability of the proposed aggregation framework.

The modular structure of the aggregation rules supports extensibility, allowing new performance indicators to be introduced without modifying the core transformation architecture. This capability provides a foundation for structured performance evaluation and decision-support analysis in model-driven simulation environments.

Future work will extend the framework with additional indicators and more granular metrics, and will include larger and more complex case studies to further validate scalability and generalization across application domains.

ACKNOWLEDGEMENT AND FUNDING

The authors receive no financial support for the research, authorship, and publication of this article.

DECLARATION OF CONFLICTING INTERESTS

The authors declare no potential conflicts of interest with respect to the research and publication of this article.

REFERENCES

[1] A. Van Looy and A. Shafagatova, Business process performance measurement: a structured literature review of indicators, measures and metrics, *SpringerPlus*, 5, 2016, 1797.
[2] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Springer, 2011.
[3] K. Rosenthal, B. Ternes, and S. Strecker, Business process simulation on procedural graphical process models: Structuring overview and paths for future research, *Business & Information Systems Engineering*, 63, 2021, 569–602.
[4] O. M. G. (OMG), *Business Process Model and Notation (BPMN) Version 2.0*, 2011.
[5] Workflow Management Coalition, Business process simulation specification version 2.0. https://www.wfmc.org/standards/bpsim, 2016.
[6] M. E. Kassis, F. Trousset, G. Zacharewicz, and N. Daclin, Bridging the gap between business process and simulation: transformation from bpmn to devs, *IFAC-PapersOnLine*, 56, 2023, 11888–11893.
[7] M. El Kassis, F. Trousset, G. Zacharewicz, and N. Daclin, Incremental transformation of BPSim-enriched BPMN models into DEVS, *Proceedings of the 2023 Winter Simulation Conference*, San Antonio, TX, USA, 2023, 2542–2553.
[8] M. El Kassis, P. Bocciarelli, F. Trousset, N. Daclin, A. D'Ambrogio, and G. Zacharewicz, An hla-based automated approach for the interoperable simulation of collaborative business processes, *Simulation Modelling Practice and Theory*, 135, 2024, 102977.
[9] M. El Kassis, *Transforming BPMN models enriched with BPSim parameters into DEVS : An Incremental Approach Enhancing Business Processes through Simulation*, Ph.D. Thesis, IMT Mines Alès, France, 2024.
[10] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management*, 2nd ed., Springer, 2018.
[11] S. Robinson, *Simulation: The Practice of Model Development and Use, 2nd edition*, 2nd ed., Palgrave Macmillan, 2014.
[12] QBP Simulator, BIMP – business process simulator. https://bimp.cs.ut.ee/, 2015.
[13] Bizagi, Simulation in bizagi. https://help.bizagi.com/platform/en/simulation_in_bizagi.htm, 2022 (accessed April 16, 2026).
[14] Rockwell Automation, Arena simulation software. https://www.rockwellautomation.com/en-us/products/software/arena-simulation.html, 2026 (accessed April 16, 2026).
[15] AnyLogic, Business process simulation software. https://www.anylogic.com/business-processes/, 2026 (accessed April 16, 2026).
[16] A. Borshchev, *The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6*, AnyLogic North America, 2013.
[17] SIMUL8, Business process management | simulation software. https://www.simul8.com/applications/business-process-management/, 2026 (accessed April 16, 2026).
[18] W. M. Van der Aalst, Challenges in business process management: Verification of business processes using petri nets, *Bulletin of the EATCS*, 80, 2003, 174–199.
[19] U. Mutarraf, K. Barkaoui, Z. Li, N. Wu, and T. Qu, Transformation of business process model and notation models onto Petri nets and their analysis, *Advances in Mechanical Engineering*, 10, 2018, 1–21.
[20] D. Cetinkaya, A. Verbraeck, and M. D. Seck, Mdd4ms: a model driven development framework for modeling and simulation, *Proceedings of the 2011 Summer Computer Simulation Conference*, Vista, CA, 2011, 113–121.
[21] H. Bazoun, Y. Bouanan, G. Zacharewicz, H. Boyer, and Y. Ducq, Business process simulation: transformation of bpmn 2.0 to devs models, *SCS/ACM/IEEE Symposium on Theory of Modeling and Simulation*, Tampa, USA, 2014.
[22] P. Bocciarelli, A. D'ambrogio, A. Giglio, E. Paglia, and D. Gianni, A transformation approach to enact the design-time simulation of bpmn models, *2014 IEEE 23rd International WETICE Conference*, Parma, Italy, 2014, 199–204.
[23] A. D'Ambrogio and G. Zacharewicz, Resource-based modeling and simulation of business processes, *Proceedings of the Summer Computer Simulation Conference*, San Diego, CA, USA, 2016.
[24] A. Alshareef, H. S. Sarjoughian, and B. Zarrin, An approach for activity-based devs model specification, *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*, Pasadena, CA, USA, 2016, 1–8.
[25] A. Alshareef and H. S. Sarjoughian, Hierarchical activity-based models for control flows in parallel discrete event system specification simulation models, *IEEE Access*, 9, 2021, 80970–80985.
[26] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of Modeling and Simulation*, Academic Press, 2000.
[27] H. S. Sarjoughian and A. M. Markid, Emf-devs modeling, *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium*, San Diego, CA, USA, 2012.
[28] B. P. Zeigler and H. Sarjoughian, Devs-suite simulator. http://acims.asu.edu/software/devs-suite/, 2010 (accessed January 15, 2026).
[29] A. M. Law, *Simulation Modeling and Analysis*, 5th ed., McGraw-Hill, 2015.