TNO innovation for life

Towards a virtual factory for high-mix low-volume production systems

TNO Public ⟩ TNO 2025 R12766

14 January 2026

ICT, Strategy & Policy
www.tno.nl
+31 88 866 50 00
info@tno.nl

TNO 2025 R12766 – 14 January 2026

Towards a virtual factory for high-mix low-volume production systems

Author(s)	Bram van der Sanden Jacques Verriet
Classification report	TNO Public
Title	TNO Public
Report text	TNO Public
Number of pages	24 (excl. front and back cover)
Number of appendices	0
Programme name	AIMS
Project name	AIMS 2025
Project number	060.62913/01.01

Disclaimer
In the creation of this document, Microsoft 365 Copilot was used to improve the use of English. In all instances where Copilot was used, the authors thoroughly reviewed the results to ensure the end result is accurate and free from errors, thus allowing the authors to take responsibility for the content.

Acknowledgement
The research is carried out as part of the AIMS program under the responsibility of TNO-ESI in cooperation with Canon Production Printing. The research activities are co-funded by Holland High Tech | TKI HSTM via the PPP Innovation Scheme (PPP-I) for public-private partnerships.

Contents

Summary

This report explores the development of a virtual factory approach tailored for high-mix low-volume (HMLV) production systems, dealing with a wide variety of products produced in small batches. The virtual factory provides system architects with the means to reason about performance of the high-tech equipment systems they design in different scenarios. Two simulation approaches are evaluated. The first approach involves commercial tools (AnyLogic and FlexSim), which encode production recipes within process flows. The second approach involves a proprietary, hierarchical simulation concept implemented in POOSL, which accepts recipes as input but requires manual priority management.

Building on these insights, we propose a new virtual factory concept comprising two main phases: recipe selection and recipe scheduling. The concept allows users to configure factory scenarios, select among multiple production recipes per product, and optimize schedules according to various criteria such as timing, cost, and resource utilization. An incremental scheduling concept is proposed to efficiently explore the scheduling space. The concept is designed to be extensible, supporting future integration of reinforcement learning for adaptive scheduling strategies. The implementation of the incremental scheduling concept is planned for 2026.

1 Introduction

By pushing the boundaries of physics, companies in the high-tech equipment industry have been improving the performance of their equipment. These systems can reach high levels of performance, with a higher quality of the products produced, increased productivity, and reduced waste and energy consumption. High-tech equipment does, however, not operate in isolation: these machines are integrated into systems of systems [1], often production systems. The performance that these systems reach in practice depends on the context in which they operate. For high-tech equipment systems, this is often a production system. For instance, professional printers operate in a printshop and wafer scanners in a wafer fab.

The AIMS programme is a collaboration between TNO-ESI and Canon Production Printing studying performance engineering for high-mix low-volume (HMLV) production systems. These are manufacturing systems that produce a large variety of products in short series. AIMS' focus is on providing system architects a means to reason about the performance of their high-tech equipment system in the context of a (HMLV) production system.

In 2023 and early 2024, AIMS focused on the (timing) performance of specific scenarios in fully automated HMLV production lines, i.e., mechanical production systems that operate without human involvement [2] [3]. Starting in 2024, the scope was broadened to production systems in which human operators are active, e.g., for transporting materials between production lines. An AIMS 2024 report identified the following characteristics of HMLV production systems [4]:

- Multiple jobs will be processed at the same time, competing for shared resources like operators and equipment, and each job might have their own routing through the production system.
- The different operations inside a job may have precedence relations. Different jobs may contain different operations and precedence relations between its operations. The operations are not necessarily sequential, meaning that multiple operations of the same job could run in parallel.
- Setup times and cycle times of any operation may vary per job. Setup times for operations may also depend on the job sequence.
- Due to low-volume orders, production is based on shorter runs, with setup times playing a key role.

1.1 Research questions

AIMS 2025 focuses on reasoning about the performance of HMLV production systems, from the viewpoint of the architects of the high-tech equipment systems in the production system. The architects are responsible for creating the right system architecture that ensures that the system in the customer context operates in an efficient and effective manner. To achieve this, the architect needs to think about dimensioning of the system (e.g., how many buffers are needed and what buffer sizes), and how the system is integrated in the workflow of the customer (e.g., inline or nearline).

AIMS 2025's main research question is the following:

- **RQ0:** How to allow an architect to reason about the performance of a system in a high-mix low-volume (HMLV) factory context?

Two refining questions have been formulated to answer the main research question:

- **RQ1:** How to create a virtual factory model for architects to specify and analyse factory scenarios?
- **RQ2:** How to support architects in optimizing the use of available factory resources?

1.2 Requirements

The intended users of our methodology are architects of high-tech equipment. These are no experts in the field of (performance) modelling and optimization. This means that our (model-based) methodology should be usable by people with limited knowledge of performance modelling. Moreover, there is a large variety with respect to equipment configurations and order sets in production systems. Our methodology should offer the flexibility to specify these variations and quickly find solutions that effectively allocate an order set on the available resources. This leads to the following main requirements to address the research questions introduced in Section 1.1:

- **REQ1:** The user, e.g., a system architect, should be able to quickly specify a *factory scenario* by configuring the virtual factory model with the available resources (e.g., equipment and operators) and an order set.
- **REQ2:** After a user has specified a factory scenario, the virtual factory model should find a high-quality schedule of the scenario's factory recipes in reasonable time.

We will use the term *factory recipe* for a way to produce products as ordered by a customer of a production system. This involves sequences of production steps and their allocation to the available resources. There may be multiple factory recipes resulting in the same product. This could involve the same sequences of production steps, but a different allocation to the available resources. In a production system, there may be multiple machines that provide the same functionality. For instance, in a printshop, there may be multiple printers that could print the content of a booklet. This would involve multiple factory recipes that produce the same product, but with different costs, e.g., amount of waste and energy usage.

Different sequences of production steps could also lead to the same product. An example involves the available materials, especially if these support ganging/nesting[1] or if they can be trimmed/cut. For instance, in the printing domain, one can produce an A4 leaflet by printing the content directly on A4 sheets. It is however also possible to print on slightly larger sheets, e.g., SRA4 sheets, and trimming the sheets to A4 dimensions. Moreover, one could print multiple leaflets on larger sheets, e.g., A3 sheets and cut the sheets into multiple A4 leaflets. These different recipes may all have different costs because of the variation of the operations and materials.

In the printing example, trimming leads to waste materials, which could be undesirable from a sustainability point of view. However, from a latency point of view, it may be beneficial to avoid changing the material to avoid equipment setup times. As there are many different materials and hence also many factory recipes leading to the same product, one would like to allow the user to reason about multiple costs and balance them. The user should limit the potentially large set of possible factory recipes to a small set of reasonable recipes.

We would like to include this recipe flexibility and multiple objectives in our virtual factory concept. For this, we formulate the following additional (advanced) requirements:

- **REQ3:** Per product in the order set, the user should specify the allowed factory recipes to produce the ordered product.

[1] https://en.wikipedia.org/wiki/Nesting_(process)

- **REQ4:** The virtual factory user should be able to reason about the trade-offs between multiple optimization criteria.

The focus of this report is on the main requirements REQ1 and REQ2. Requirements REQ3 and REQ4 are advanced requirements to be addressed later, but the concepts considered in the report should not hamper them.

1.3 Outline

Chapter 2 introduces the HMLV production systems as considered in this report. Chapter 3 gives an overview of related work. In Chapters 4 and 5, we evaluate the use of simulation as a basis for a virtual factory. Chapter 4 considers the usability of commercial simulation tools by modelling a production line and the corresponding workflows from the professional printing domain. Chapter 5 considers a proprietary approach using open-source simulation software, which is flexible with respect to production line equipment and workflows. Chapter 6 proposes a new virtual factory concept to address the limitations of the approaches presented in Chapters 4 and 5 with respect to usability for the system architect.

2 HMLV production systems

In this report, we will assume that a factory's equipment is built up as shown in **Figure 2.1**. A factory consists of (fully automated) production lines with their input and output buffers. We assume that the production lines in a factory have limited internal storage capacity. Each production line consists out of one or multiple equipment modules that perform operations. To move material from one module to the next module, a line's modules are connected by fully automated transportation, e.g., a transport belt.

Different production lines are connected by transports between their buffers; such transport is done by human operators or AGVs. One production line's output buffer may also be another production line's input buffer; then transport is handled mechanically.

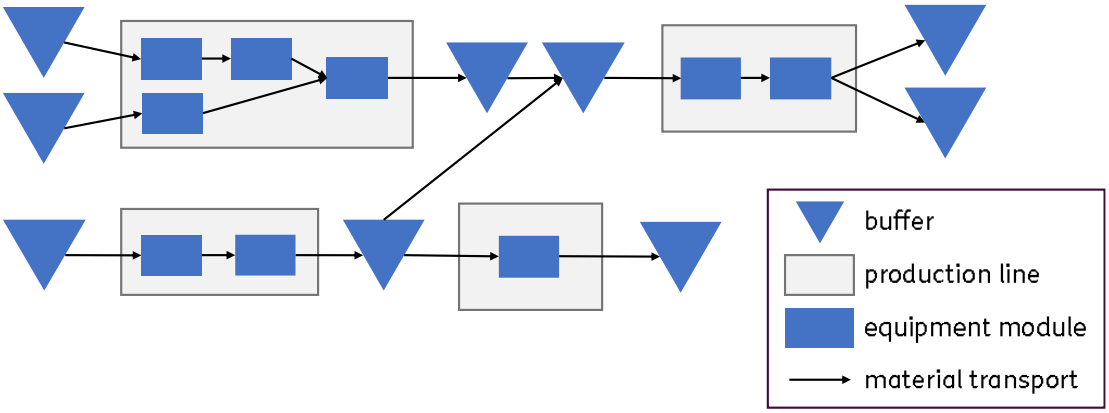

Figure 2.1: Factory equipment.

We will call the work to be done by the (HMLV) production system a *job mix*. A job mix is defined in terms of factory recipes and line recipes. A *line recipe* is a directed acyclic graph of operations that can be executed by of a production line's modules. Arcs between a line recipe's operations represent the precedence relation between the operations. The operations are linked to the modules in a production line. As a production line's modules have limited storage capacity, we assume that a line recipe starts by retrieving materials from the line's input buffer and end with storing materials in the line's output buffers.

An example of a line recipe is shown in **Figure 2.2**. This recipe consists of six operations, number 1 through 6. If operations in a line recipe are allocated to different modules, then their precedence relation must follow the topology of the equipment. This is the case for operations 1 and 2, 2 and 5, 4 and 5, and 5 and 6. For simplicity, we assume that line recipes do not have any sequencing freedom. This means that if two operations are allocated to the same module, like operations 3 and 4, then there must be a precedence relation between them. In the example in **Figure 2.2**, operation 3 precedes operation 4. As line recipes start by retrieving materials from input buffers and end by storing materials in output buffers, operations 1 and 3 must be retrieval operations, and operation 6 must be a storage operation.

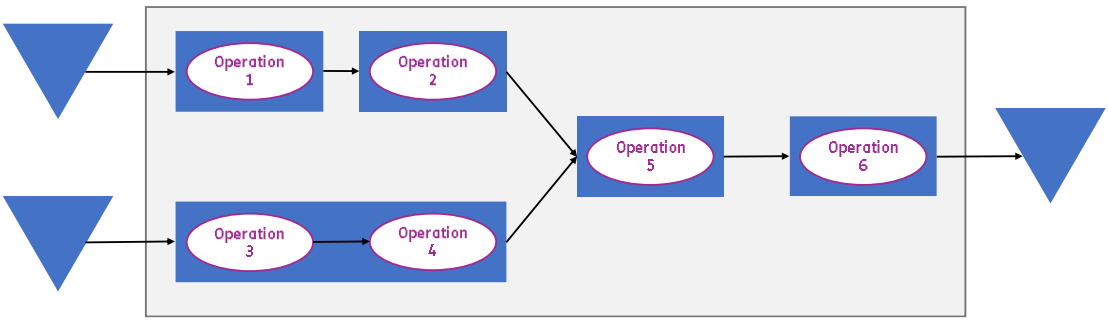

Figure 2.2: Line recipe consisting of operations allocated to an production line's modules.

The execution of a line recipe by an equipment line involves three types of delays:

- The execution of an operation by an equipment module has an *operation duration*, which depends on the materials that the operations need to handle. Large materials typically require longer operation durations than small materials.
- Before a module can perform an operation, it may need to adapt to the operation's materials. This requires a (sequence-dependent) *setup time*. This setup could involve adjusted to the materials' dimensions.
- If subsequent operations are performed by different modules, then a *transport time* is needed between the operations.

A *factory recipe* is a directed acyclic graph of line recipes. The arcs between the line recipes represent precedence constraints between the line recipes. These may be realized by transportation from one production line's output buffer to another line's input buffer. The factory recipe example in **Figure 2.3** involves three line recipes. Line recipe 1 consists of five operations Op1.1, Op1.2, Op1.3, Op1.4, and Op1.5. It precedes line recipe 2, which has operations Op2.1 and Op2.2 and line recipe 3, which has operations Op3.1 and Op3.2.

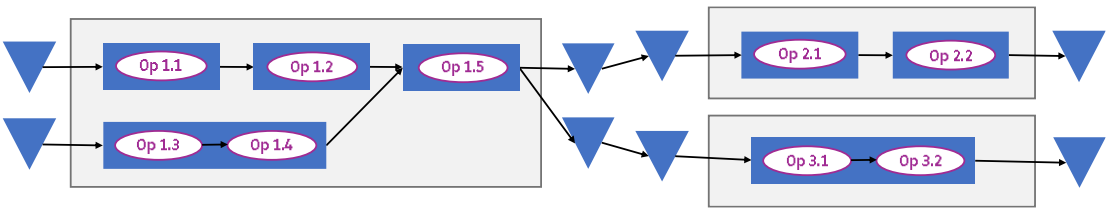

Figure 2.3: Factory recipe consisting of multiple line recipes.

This report presents approaches to schedule factory recipes, and their constituting line recipes and operations, to a factory's equipment.

3 Related work

This chapter discusses existing work related to performance analysis for HMLV production systems.

3.1 HMLV production system scheduling

Just Plan It [5] is a commercial tool for scheduling of high-mix low-volume production systems. The tool schedules jobs as soon as possible based on a priority list. Functionality is available to derive which priorities ensure that a job finishes before their due date. Jobs can also be scheduled just-in-time given their due date. A user does not have to order the priority list himself/herself, as Just Plan It offers functionality to sort the work to be done. The tool offers several common scheduling heuristics, such as earliest deadline first and least slack time first. The user can partially override the heuristics by fixing the location of specific jobs in the priority list. Just Plan It does not seem to consider transport between machines.

Many scheduling algorithms assume that material will eventually arrive at the machine that should perform the operation and consequently abstract from the factory topology and material handling. Capability-based planning and scheduling [6] has been presented as solution that does not abstract from the material and computes a production schedule that does consider the material flow. For all production plans, the scheduling computes how operations requiring capabilities can be allocated onto machines providing these capabilities. The scheduling also considers that material can be transported between subsequent steps in a production plan. Keddis et al. [6] optimize the production plans' schedules with respect to minimum duration. These individually optimized schedules are then scheduled sequentially on the available resources. The approach does not consider batching or buffer sizes. In our setting, we assume that transport between any two production lines is possible, and batching and limited buffer sizes are a key concern to derive a feasible schedule.

3.2 Factory simulation

Simulation offers a versatile way to capture dynamic system behaviour. There are several types of simulation, the most common one being discrete event simulation (DES), agent-based simulation (ABS) and system dynamics simulation (SDS). DES and ABS are relevant for factory-level simulation, DES can describe a factory's material flow and ABS its distributed intelligence [7]. SDS is typically applied on a smaller scope, e.g., for the simulation of a physical (production) step.

Simulation can be used for factory design as well as factory operation [8]. The former involves a large degree of uncertainty about a factory's (future) workload. This can be addressed by simulating many scenarios and calculating the best overall operational performance [9]. A source of uncertainty for factory operation is the human operator, which is less predictable than the equipment, especially with respect to timing [10].

There are many simulation tools, both open source and commercial. Overviews of simulation tools can be found online [11] [12]. Dias et al. [13] compare 19 discrete event simulation tools with respect to their popularity. There are several articles comparing simulation tool functionality in a factory setting. Attajer et al. [14] compare AnyLogic and FlexSim for a

production system with distributed control requirements. They conclude that AnyLogic is better suited to capture product flows than FlexSim. Paape et al. [7] compare multiple simulation tools for production systems. They conclude that AnyLogic [15] is the most promising simulation tool for a combination of discrete event simulation and agent-based simulation. Mourtzis et al. [16] compare AnyLogic [15], Arena [17], FlexSim [18], Siemens Tecnomatix Plant Simulation [19] and Witness [20]. In their evaluation, AnyLogic, FlexSim and Siemens Tecnomatix Plant Simulation score well.

Simulation is typically used for the analysis of a single scenario. By varying its inputs, a design space can be explored. This is called simulation optimization [21]. Some simulation tools offer simulation optimization. Examples include AnyLogic and FlexSim.

3.3 Workflow modelling

BPMN [22] is a well-known formalism to describe business workflows / processes, and sometimes also used to describe manufacturing processes [23]. It provides the concepts of resources and resource roles that can be referred to by activities. A limitation is that only a single resource can be assigned to an activity, meaning that BPMN cannot describe one-to-many or many-to-many relationships between activities and resources [24]. Furthermore, BPMN lacks the primitives to describe traveling times of products in a manufacturing process [23].

Process mining is a data-driven method to automatically discover business workflows from event logs. It has also been applied in the field of manufacturing [25], where object-centric process mining (OCPM) [26] is seen as promising extension to cover the interplay between resources, processes, products, and customer orders. OCPM discovers object-centric Petri nets (OCPNs) [27], where places and transitions are colored based on the related objects.

Hooman et al. [28] describe an MBSE method to model workflows and analyse productivity by annotating the workflow with quantitative information and resources being used. They use Capella [29] to demonstrate their method: they use Capella's functional chains to specify workflows and specify quantification information using its Property Values Management Tools (PVMT) addon.

3.4 Recipes and recipe management

In highly automated manufacturing environments, recipes play a critical role in defining the precise parameters for each production process to ensure quality and consistency of the end products, as well as optimizing the production efficiency. Parameters in a recipe include material quantities, temperature settings, and durations of processing steps. An example of a highly-automated manufacturing domain is the semiconductor industry, which uses the "SEMI E139 – Specification for Recipe and Parameter Management (RaP)" standard [30] to define how factory control systems interact with manufacturing equipment to manage processing specifications, including equipment recipes and parameters.

Recipe Management Systems (RMS) are used to automate the handling of production recipes in manufacturing environments. It complements Manufacturing Execution Systems (MES) by managing recipe bodies, version control, and equipment-level recipe selection, which are often

not covered by MES. Various commercial tools exist in the field of semiconductor industry, e.g., by Systema[2] and ZNT,[3] but also in other domains like the food industry.[4]

[2] https://www.systema.com/digital-transformation/recipe-management-systems
[3] https://www.znt-richter.com/en/recipe-management-system-rms
[4] https://www.sw.siemens.com/en-US/digital-thread/enterprise-recipe-management/

4 Simulation tool exploration

Because of its versatility, we aim to use simulation as the basis for a virtual factory model. Based on the comparison of simulation tools in the literature (see Chapter 3), we selected AnyLogic fand FlexSim for further investigation. This involved applying these tools to a use case from the professional printing domain. This use case involved two simple production systems for creating booklets. Both systems have two equipment lines: (1) a printing line, which takes blank sheets of paper and creates book blocks, and (2) a booklet making line, which creates a booklet by folding, stitching and trimming the book blocks. In one system configuration, the equipment lines are directly connected, meaning that the output buffer of the printing line is the input buffer of the booklet making line (see Figure 4.1). In the other configuration, there is a transport between the output buffer of the printing line and the input buffer of the booklet making line (see Figure 4.2).

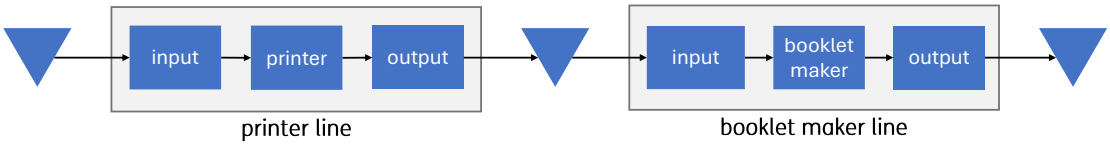

Figure 4.1: Printer and booklet maker line sharing a buffer (adapted from [4]).

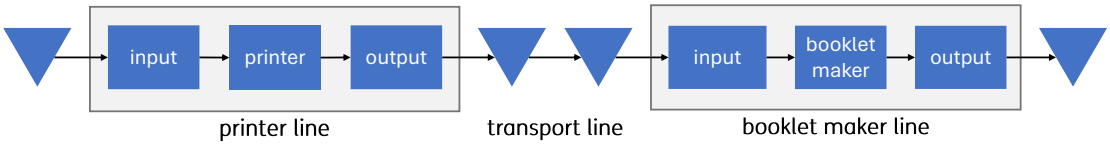

Figure 4.2: Printer line and a booklet maker line connected via transport (adapted from [4]).

Our initial assessment involved two student projects creating models in AnyLogic [31] and FlexSim [32] and comparing the models with respect to expressiveness and usability. The AnyLogic model was based on process flows capturing the recipes to produce booklets; the FlexSim model did not use process flow but focused on the flow of material between machines. The students concluded that AnyLogic outperforms FlexSim with respect to expressiveness, whereas FlexSim's usability is better than AnyLogic's [31] [32].

To get an expert view on the usability of AnyLogic and FlexSim, the AIMS project consulted external AnyLogic and FlexSim experts to independently create models for the printing use case. Interestingly, the consultants took a similar modelling approach: the recipe to create booklets was captured in the tools' process flow. A process flow captures the sequence of steps and the synchronization between them. For instance, synchronization may involve claiming of shared resources, such as materials, equipment or operators. The distinction between directly and indirectly connected equipment lines was captured by the process flow's conditional steps. These conditionals allow starting a special workflow for the transport of materials between the equipment lines. With about a day of modelling effort, both consultants were able to create a simulation model that captures the behaviour of both variants of the use case. The created models offer some degree of parameterization, e.g., with respect to the number of available operators.

While building the models, the experts used their extensive knowledge of the semantics of the building blocks that AnyLogic and FlexSim offer. If no suited block was available, custom scripts were written to achieve the desired (synchronization) behaviour. This makes the tools less usable for occasional users, like the architects that AIMS targets as the users of the virtual factory.

Both consultants applied the process flow functionality of the respective simulation tools. The resulting AnyLogic model was very abstract, as the process flow does not reflect the flow of material. The AnyLogic consultant indicated that creating a (material flow) visualization takes approximately the same amount of time as creating the process flow. The FlexSim model was less abstract than the AnyLogic model because it came with a material flow visualization. This confirms the students' observation of the better usability of FlexSim [31] [32].

4.1 Evaluation

We evaluate the simulation models by assessing whether they satisfy the (main) requirements introduced in Chapter 1. The simulation models partially satisfy requirement REQ1. The simulation models made by the consultants allow quick variation of the job mix: the number and the size of the jobs could be varied. However, this does not hold for the production recipe as it is encoded in the models' process flow. This encoding of the production recipes in the model is a potential usability challenge, at least for the occasional user. HMLV production systems have a variety of products and hence also a variety of production recipes. If there is only a small number of recipes, then these could all be encoded as separate process flows. One would then require a higher-level process flow coordinating the (active) process flows. The feasibility of a production recipe depends on the available equipment in the production system. Equipment changes may therefore lead to new production recipes, but they may also lead to production recipes becoming infeasible. Requiring an expert user to update the set of recipes for the occasional user is a tool usability issue.

The consultants' simulation models only consider a single sequence of production recipes. Their models do not address requirement REQ2. The students, however, did consider different sequences of production recipes [31] [32]. They used the tools' optimization functionality, i.e., FlexSim's Experimenter and AnyLogic's OptQuest optimization engine, to find optimum recipe sequences. They also explored the number of operators in the system using these exploration functions. This means that FlexSim and AnyLogic at least partially satisfy requirement REQ2.

The external consultants indicated that their customers typically hire them to provide an answer to a specific question (and its variations). They indicated that their customers were mostly interested in the outcome of their models. The models themselves were mostly used to validate the models' outcome. When companies have many questions of a similar nature, there is an incentive to develop a company-specific simulation concept. NS (Dutch Railways) and Vanderlande are such companies. Both have a simulation department, which offers modelling and simulation services to the company. NS uses commercial simulation tools, including AnyLogic and FlexSim. In the past, Vanderlande used Automod [33] as their main simulation tool. As this tool will be discontinued, Vanderlande's simulation department has developed a proprietary simulation (and emulation) environment.

5 Hierarchical simulation concept

The tool exploration explained in Chapter 4 revealed that the commercial simulation concepts lack flexibility with respect to the equipment and the corresponding workflows: these are hardcoded in the simulation model. To address this lack of flexibility, we developed a proprietary simulation concept in POOSL [34]. In this simulation concept, production recipes are inputs to the simulation model instead of hardcoded in the model.

The simulation concept assumes a factory context as introduced in Chapter 2 (see **Figure 2.3**). Hence a factory is built up from production lines and material buffers. The production lines consist of equipment modules with little to no storage capacity. A line recipe corresponds to a specific production line. We assume that the decision of which production line(s) to use to produce a product have already been made. We also assume that there are no rush jobs to be handled and that all recipes have an equal priority.

The simulation has a hierarchical structure matching the hierarchical structure of a production system. It involves three types of schedulers: (1) one factory scheduler, (2) a line scheduler for each production line and (3) a module scheduler for production module. This structure is visualized in Figure 5.1 for the production lines shown in **Figure 4.1** and **Figure 4.2**. Because these only differ slightly, their scheduler hierarchy is identical.

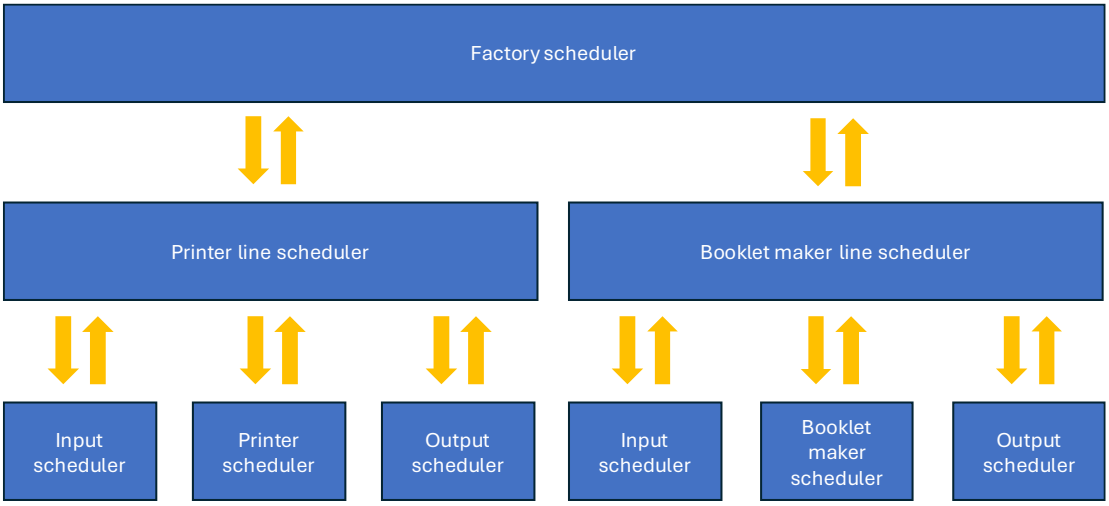

Figure 5.1: Scheduler hierarchy in simulation concept

The scheduling on factory level is done by a *factory scheduler*. It maintains an overview of all unscheduled lines recipes. It iteratively selects a line recipe, for which all preceding line recipes have been completed, and schedules it on the corresponding production line. This is done by communicating the line recipe to the line scheduler of the corresponding production line.

A *line scheduler* operates in a similar manner as the factory scheduler, but on an operation basis. It maintains an overview of all unscheduled operations of a line recipe. The line scheduler

iteratively selects one of the operations, for which all preceding operations have been completed, and schedules it by communicating the operation to the corresponding equipment module's module scheduler. The line scheduler informs the module scheduler when materials for an operation will arrive.

A *module scheduler* schedules an operation as early as possible after the already scheduled operations. The starting time of the operation depends on the communicated arrival time, the completion times of the preceding operations and the module's setup time. When an operation is finished, the module scheduler reports the operation's completion time to the line scheduler.

After the line scheduler receives an operation's completion notification, it determines which operations can be scheduled next. It continues until all operations have been completed. If all operations of a line recipe have executed, the line recipe's completion is communicated to the factory scheduler. This scheduler updates the line recipes that can be scheduled and selects one. The factory scheduler continues until all line recipes have been scheduled.

5.1 Evaluation

As we did in Chapter 4 for the models built using the commercial simulators, we assess whether the hierarchical simulation concept satisfy (main) requirements REQ1 and REQ2 introduced in Chapter 1.

Compared to the simulations presented in Chapter 4, the hierarchical simulator presented in this chapter is more configurable, because the workflow is not hardcoded in the simulation. Both the factory recipes and the line recipes are described using directed acyclic graphs that are input for the hierarchical simulator. So it satisfies requirement REQ1.

With respect to requirement REQ2, the hierarchical scheduler concept also has a serious limitation. We use priority lists and the basis for the selection mechanism of both the factory scheduler and the line schedulers. When work elements, either a line recipe or an operation, can be scheduled, the schedulers will select the work element highest on this list. It is the model creator's responsibility to fill the priority list in the best possible manner. This is not a simple task. A model user should foresee the availability of all resources, and which work elements are the most urgent and use this information to compile an appropriate priority list. REQ2 can be realized by having the simulation evaluate all schedulable work elements and then select the most promising one. Unfortunately, this is not an easy change to the hierarchical simulation concept, as it is very operational, i.e., it is targeted as executing the work elements.

Another limitation with respect to requirement REQ2 comes from the choice of simulation tool. POOSL's performance proved to be insufficient; its data structures have inefficient implementations. This results in simulation of a scenario with a few thousand operations taking more than a minute on a standard laptop. As the scheduling space is huge for scenarios with many line recipes, searching this space would be infeasible. This could, to some extent, be addressed by only scheduling on factory and line level, but it would still necessitate the concept's user, e.g., the system architect, to manually explore the scheduling space.

6 Virtual factory concept

In Chapters 4 and 5, we presented our efforts in developing a virtual factory. The goal of this virtual factory is reasoning about the influence of individual production machines on the performance of HMLV production systems in which they are deployed. Both approaches had its weaknesses with respect to usability. The simulations discussed in Chapter 4 fails to satisfy requirement REQ1: it does not allow easy variation of a job mix; this required changes to the simulation model. The hierarchical simulation discussed in Chapter 5 does not satisfy requirement REQ2: it requires the model user to manually explore the scheduling space to find a good allocation of a job mix onto the available equipment. In this chapter, we introduce our conceptual solution for a refined virtual factory that allows for easy variation of the job mix and can automatically find high-quality job schedules for a given job mix onto the available equipment in the factory.

In this chapter, we will propose an alternative virtual factory that will be developed in AIMS 2026. The concept is explained in Section 6.1 and an implementation based on LSAT in Section 6.2.

6.1 Virtual factory workflow

Based on the (main and advanced) requirements identified in Chapter 1, we propose a virtual factory concept with two high-level phases: (1) a recipe selection phase and (2) a recipe scheduling phase. The workflow is shown in Figure 6.1.

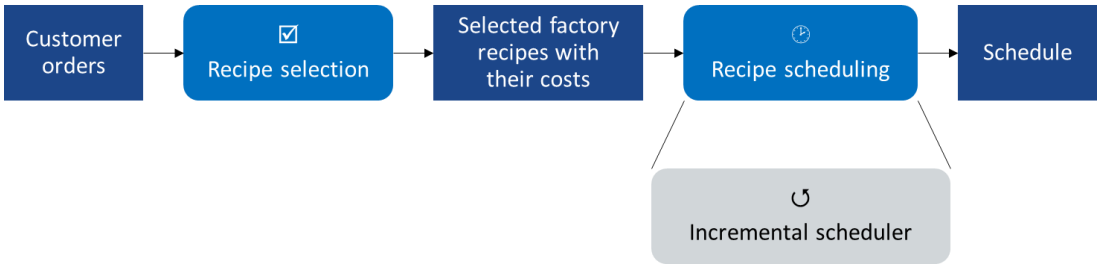

Figure 6.1: Virtual factory workflow.

The *recipe selection phase* starts from a set of products as ordered by a production system's customers. As explained in Section 1.2, there may be many ways to create the products order by the customer(s). Not all of these are feasible from a cost perspective. The recipe selection phase identifies the allowed factory recipes, typically only a few, and their corresponding costs in terms of the optimization criteria (e.g., for aspects like usage of resources, equipment costs, and energy consumption). The selected recipes and their costs are input for the *recipe scheduling phase*. This phase builds a schedule that can be executed by the production lines to produce the desired products requested by the customer(s).

As explained in Section 1.2, the constructed schedule should ensure that the production lines are used to produce the order products in an effective manner, because we do not want to burden the virtual factory user with exploring the huge scheduling space. Exhaustive exploration of this scheduling space is not feasible (in a reasonable amount of time). Hence, we require a heuristic approach. We propose an incremental scheduling approach, inspired by

decision diagrams [35] and the just-plan-it planning and scheduling software [5]. The incremental nature of such a scheduling approach allows fast scheduling of factory and line recipes.

The incremental scheduling approach can be explained using Figure 6.2. The example involves two customer orders. For both orders, one factory recipe was selected (FR1 and FR2). Both factory recipes consist of two line recipes: FR1 consists of line recipes A and B and FR2 of line recipes C and D. Figure 6.2 shows a partial schedule in which line recipes A and B have already been scheduled. In the next iteration, the partial schedule is extended with either line recipe B or line recipe D. In the example, line recipe B was added to the partial schedule.

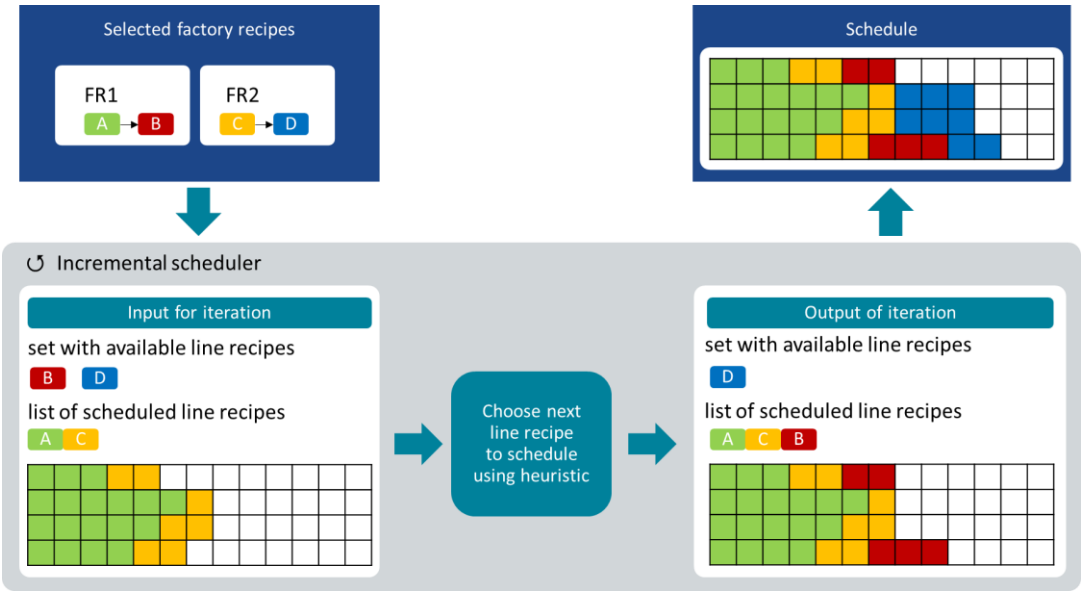

Figure 6.2: Incremental scheduling approach.

The core element of the incremental scheduler is the *selection heuristic*, which must select and allocate the next line recipe. Many scheduling heuristics have been proposed in the literature [36]. For the virtual factory, we will develop a heuristic that considers both the effective usage of factory resources, i.e., the execution of operations, as well as the overhead from transportation and setup. Initially, we will use manually specified heuristics, drawing inspiration from established scheduling rules found in literature [36]. Later, we will explore the usage of *reinforcement learning* (RL) to learn decision rules that could tailor the scheduling strategy to aspects like volume of the customer orders, due dates, and the product mix to be manufactured. The decision rules derived from the RL policy, to decide which actions to take in different situations, should ideally be interpretable to obtain insights in what scheduling strategy works well for specific use cases.

In the explanation illustrated using **Figure 6.2**, we have not yet considered transportation. In a production system, transportation is needed to transport materials between production lines: intermediate products must be taken from one line's output buffer, transported, and stored in another line's input buffer. The incremental scheduler needs to schedule such transportations as well. For this, it should consider the location of the materials and the location of the transportation resources, i.e., the operators and automated guided vehicles (AGVs). This means that transportations can be both idle, i.e., not transporting materials, and non-idle, i.e., transporting materials. Ideally, the scheduler should consider batching of transportation tasks: if multiple materials need to be transported from one line's output buffer to another line's output

buffer, the scheduler could schedule a single transportation task for all materials. This requires considering the capacities of the material buffers and the transportation resources.

6.2 Virtual factory realization

We aim to build the new virtual factory concept on top of LSAT [37]. LSAT is a performance engineering tool for flexible manufacturing systems that provides both a modeling framework and efficient analytical capabilities. In LSAT, a system is described in terms of a platform and an application [38]. The platform describes a system's equipment in terms of resources and their peripherals, and the operations that the peripherals can perform. The platform also describes the timing of the peripherals' operations. An LSAT application is defined in terms of activities and (logistic) scenarios. Activities are directed acyclic graphs of peripheral operations and scenarios are compositions of activities.

An LSAT activity can be translated into a (max, +) matrix [39]. The timing of an LSAT scenario can be analyzed very efficiently by multiplying the matrices of its activities. We intend to use the latter in our virtual factory concept. Line recipes (and transports) can be modelled as LSAT activities. LSAT's matrix computation will be used by the incremental scheduler to evaluate (the timing of) all possible line recipes and use the obtained information to extend the partial schedule with one of the line recipes.

Rationale of choosing LSAT

We have chosen LSAT as modeling and analysis tool for the following reasons:

- **Fit in terms of domain modeling:** LSAT's platform and application structure align closely with our virtual factory representation of factory resources, operations, and recipes.
- **Computational efficiency:** LSAT's use of (max,+) algebra enables fast timing analysis compared to general-purpose simulation. This efficiency is critical for our incremental scheduling approach, which must evaluate multiple candidate recipes under high-mix, low-volume (HMLV) constraints.
- **Direct support for HMLV objectives:** LSAT provides quick determination of resource usage and timing—primary objectives for our scheduler. While these are not the only objectives (others include operator/equipment costs, material waste, and consumables), LSAT gives us a strong analytical foundation to build upon and can be easily extended to model additional aspects like cost, energy consumption, and material waste related to executing operations.
- **Integration and maintainability:** LSAT is in active development at TNO-ESI with a related project developing new functionality that we can leverage in this project. Conversely, we can contribute generic results of the virtual factory concept back to LSAT. Its open development model reduces vendor lock-in and ensures long-term maintainability of the virtual factory concept.

7 Conclusion

This report addresses the challenges and opportunities in developing a virtual factory concept for high-mix low-volume (HMLV) production systems. The report aims to answer the following main research question RQO and subordinate questions RQ1 and RQ2:

- **RQ0:** How to allow an architect to reason about the performance of a system in a high-mix low-volume (HMLV) factory context?
- **RQ1:** How to create a virtual factory model for architects to specify and analyse factory scenarios?
- **RQ2:** How to support architects in optimizing the use of available factory resources?

The report's focus was on simulation-based approaches to address these questions because of the inherent flexibility of simulation. To assess the approaches, we defined two main requirements (REQ1 and REQ2) and two advanced requirements (REQ3 and REQ4):

- **REQ1:** The user, e.g., the system architect, should be able to quickly specify a factory scenario by configuring the virtual factory model with the available resources (e.g., equipment and operators) and an order set.
- **REQ2:** After a user has specified a factory scenario, the virtual factory model should find a high-quality schedule of the scenario's factory recipes in reasonable time.
- **REQ3:** Per product in the order set, the user should specify the allowed factory recipes to produce the ordered product.
- **REQ4:** The virtual factory user should be able to reason about the trade-offs between multiple optimization criteria.

Unfortunately, existing commercial simulation tools (AnyLogic and FlexSim) fall short with respect to capturing equipment flexibility. This missing flexibility was addressed by a proprietary simulation (in POOSL), which allows easy variation of equipment, workload and the workload's allocation. However, this simulation concept also proved to be insufficiently usable: the simulation's user, typically a system architect, is responsible for (optimally) allocating the workload. **Table 7.1** shows the conclusions of our tool evaluation.

The explored concepts sufficiently answered research question RQ1. To address research question RQ2, the allocation optimization should be included in the virtual factory approach. For this, we propose a yet-to-be-developed virtual factory concept comprising two main phases: *recipe selection* and *recipe scheduling*. The concept allows architects to configure factory scenarios, select among multiple production recipes per product (the recipe selection phase), and optimize schedules (in the recipe scheduling phase) according to various criteria such as timing, cost, and resource utilization. We aim to build the new virtual factory concept on top of LSAT given its fit in terms of domain modeling, its computational efficiency, direct support for HMLV objectives and integration and maintainability support.

Table 7.1: Summary of how well the tools meet requirements REQ1 and REQ2.

Tool	REQ1 Factory scenario specification	REQ2 Scheduling automation
AnyLogic	*Moderate*: Job mix can be varied easily, but production recipes are	*Partial*: Optimization possible using the OptQuest optimization engine, but only

Tool	REQ1 Factory scenario specification	REQ2 Scheduling automation
	encoded in the models' process flow and require expert modelling.	for encoded process flows; not scalable for many recipes.
FlexSim	*Moderate*: Better usability than AnyLogic as it also provided a material flow visualization with limited modelling effort, but similar hardcoding of recipes.	*Partial*: FlexSim's Experimenter provides optimization functionality, but only for the encoded process flows.
POOSL	*Good*: Production recipes are model inputs, not hardcoded. Factory recipes, line recipes, and the factory layout can be varied easily.	*Poor*: Scheduling relies on manual specification of priority lists. The user must manually explore the scheduling space. We also encountered performance issues during model execution limiting scalability.
LSAT	*Good*: Provides the flexibility to vary the platform and production recipes. Modelling concepts closely match the virtual factory concepts.	*Good*: Efficient timing analysis to analyse line recipes and extensible to add incremental scheduling. Support for automation.

An incremental scheduler is proposed which uses a heuristic that efficiently explores the scheduling space, balancing multiple objectives such as timing, costs, and resource utilization. The approach is designed to be extensible, with future work aimed at integrating advanced optimisation techniques (e.g., reinforcement learning) and addressing practical requirements like material handling, buffer management, and disruption handling (e.g., to handle machine breakdowns or rush jobs).

Both the recipe selection and the recipe scheduling of the proposed virtual factory concept will be developed in the context of AIMS 2026.

8 References

[1] T. Hendriks and S. Acur, “2024 R10452 Vision and Outlook for Systems Architecting and Systems Engineering in the High-Tech Equipment Industry,” TNO, Eindhoven, 2024.

[2] J. Verriet, “2023 R12451 Compositional Performance Prediction for Tightly Coupled Manufacturing Systems,” TNO, Eindhoven, 2023.

[3] J. Verriet, “2025 R11609 Production Line Performance Optimisation,” TNO, Eindhoven, 2025.

[4] B. van der Sanden and J. Verriet, “2025 R10304 Performance engineering of high-mix low-volume production systems,” TNO, Eindhoven, 2025.

[5] just plan it, “Production Scheduling Software and Tools for HMLV shops,” 2025. [Online]. Available: https://www.just-plan-it.com/. [Accessed October 2025].

[6] N. Keddis, G. Kainz and A. Zoitl, “Capability-based Planning and Scheduling for Adaptable Manufacturing Systems,” in *IEEE Emerging Technology and Factory Automation (ETFA)*, 2014.

[7] N. Paape, J. van Eekelen and M. Reniers, “Review of simulation software for cyber-physical production systems with intelligent distributed production control,” *International Journal of Computer Integrated Manufacturing,* vol. 37, no. 5, pp. 589-611, 2024.

[8] A. Negahban and J. S. Smith, “Simulation for manufacturing system design and operation: Literature review and analysis,” *Journal of Manufacturing Systems,* vol. 33, no. 2, pp. 241-261, 2014.

[9] I. Jithavech and K. Kumar Krishnan, “A simulation-based approach for risk assessment of facility layout designs under stochastic product demands,” *The International Journal of Advanced Manufacturing Technology,* vol. 29, pp. 27-40, 2010.

[10] T. Baines, S. Mason, P.-O. Siebers and J. Ladbrook, “Humans: the missing link in manufacturing simulation?,” *Simulation Modelling Practice and Theory,* vol. 12, no. 7-8, pp. 515-526, 2004.

[11] Wikipedia, “List of discrete event simulation software,” 2025. [Online]. Available: https://en.wikipedia.org/wiki/List_of_discrete_event_simulation_software. [Accessed November 2025].

[12] Wikipedia, “List of computer simulation software,” 2025. [Online]. Available: https://en.wikipedia.org/wiki/List_of_computer_simulation_software. [Accessed November 2025].

[13] L. M. da Silva Dias, A. A. C. Vieira, G. A. B. Pereira and J. A. Oliveira, “Discrete simulation software ranking — A top list of the worldwide most popular and used tools,” in *2016 Winter Simulation Conference (WSC)*, Washington, DC, 2016.

[14] A. Attajer, S. Darmoul, S. Chaabane, F. Riane and Y. Sallez, “Benchmarking Simulation Software Capabilities Against Distributed Control Requirements: FlexSim vs AnyLogic,” in *Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future (SOHOMA)*, Paris, 2020.

[15] The Anylogic Company, “Anylogic Simulation Software,” 2025. [Online]. Available: https://www.anylogic.com/. [Accessed October 2025].

[16] D. Mourtzis, M. Doukas and D. Bernidaki, “Simulation in Manufacturing: Review and Challenges,” *Procedia CIRP,* vol. 25, pp. 213-229, 2014.

[17] Rockwell Automation, “Arena Simulation Software,” 2025. [Online]. Available: https://www.rockwellautomation.com/en-us/products/software/arena-simulation.html. [Accessed October 2025].

[18] FlexSim Software Products, Inc., “3D Simulation Modeling and Analysis Software,” 2025. [Online]. Available: https://www.flexsim.com/. [Accessed October 2025].

[19] Siemens Digital Industries Software, “Tecnomatix Plant Simulation,” 2025. [Online]. Available: https://plm.sw.siemens.com/en-US/tecnomatix/products/plant-simulation-software/. [Accessed October 2025].

[20] Haskoning, “Witness Simulation Modelling Software,” 2025. [Online]. Available: https://www.haskoning.com/en/twinn/products/witness. [Accessed October 2025].

[21] A. Ghasemi, F. Farajzadeh, C. Heavey, J. Fowler and C. T. Papadopoulos, “Simulation optimization applied to production scheduling in the era of industry 4.0: A review and future roadmap,” *Journal of Industrial Information Integration,* vol. 39, 2024.

[22] Object Management Group, *Business Process Model and Notation (BPMN) - version 2.0.2,* OMG, 2013.

[23] J. Erasmus, I. Vanderfeesten, K. Traganos and P. Grefen, “Using business process models for the specification of manufacturing operations,” *Computers in Industry,* vol. 123, 2020.

[24] G. Wagner, Business Process Modeling and Simulation with DPMN, Brandenburg: Sim4Edu, 2024.

[25] J. Villwock Gomes de Oliveira, E. Alves Portela Santos and S. Pereira Detro, “Uncovering the potential and pitfalls of Process Mining in manufacturing,” *Procedia CIRP,* vol. 132, pp. 19-24, 2025.

[26] W. Van der Aalst, “Object-Centric Process Mining: Unraveling the Fabric of Real Processes,” *Mathematics,* vol. 11, no. 12, 2023.

[27] W. van der Aalst and A. Berti, “Discovering Object-Centric Petri Nets,” *Fundamenta Informaticae,* vol. 175, no. 1-4, 2020.

[28] J. Hooman, K. Kanters, V. Alexandr and J. Verriet, “MBSE-Based Design Space Exploration for Productivity Improvement Using Workflow Models,” in *Proceedings of the 2023 Conference on Systems Engineering Research*, 2024.

[29] Eclipse Foundation, “Capella Open Source MBSE Tool,” [Online]. Available: https://mbse-capella.org/. [Accessed December 2025].

[30] SEMI, “SEMI E139 - Specification for Recipe and Parameter Management (RaP),” SEMI, 2011.

[31] J. van Maren, “Simulating HMLV printing systems in AnyLogic,” Eindhoven University of Technology, 2025.

[32] O. Vermeulen, “Simulating High-Mix Low-Volume Manufacturing Systems in FlexSim,” Eindhoven University of Technology, 2025.

[33] Applied Materials, “SmartFactory Simulation AutoMod,” 2025. [Online]. Available: https://appliedsmartfactory.com/semiconductor/supply-chain-solutions/simulation-automod/. [Accessed October 2025].

[34] Eclipse Foundation, “POOSL,” 2025. [Online]. Available: https://www.poosl.org/. [Accessed October 2025].

[35] W.-J. van Hoeve, “An Introduction to Decision Diagrams for Optimization,” *Tutorials in Operations Research,* vol. 2024, pp. 117-145, 2024.

[36] S. S. Panwalkar and W. Iskander, “A Survey of Scheduling Rules,” *Operations Research,* vol. 25, no. 1, pp. 45-61, 1977.

[37] Eclipse Foundation, “Eclipse LSAT,” 2025. [Online]. Available: https://eclipse.dev/lsat/. [Accessed October 2025].

[38] B. van der Sanden, Y. Blankenstein, R. Schiffelers and J. Voeten, “LSAT: Specification and Analysis of Product Logistics in Flexible Manufacturing Systems,” in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, Lyon, 2021.

[39] B. van der Sanden, J. Bastos, J. Voeten, M. Geilen, M. Reniers and T. Basten, “Compositional specification of functionality and timing of manufacturing systems,” in *2016 Forum on Specification and Design Languages (FDL)*, Bremen, 2016.

ICT, Strategy & Policy

High Tech Campus 25
5656 AE Eindhoven
www.tno.nl

TNO innovation for life