A Real-world Case Study of Process and Data Driven Predictive Analytics for Manufacturing Workflows

Petros Papapanagiotou
School of Informatics
University of Edinburgh
pe.p@ed.ac.uk

James Vaughan
School of Informatics
University of Edinburgh
jvaughan@ed.ac.uk

Filip Smola
School of Informatics
University of Edinburgh
f.smola@sms.ed.ac.uk

Jacques Fleuriot
School of Informatics
University of Edinburgh
jdf@ed.ac.uk

Abstract

We present a novel application of business process modelling and simulation of manufacturing workflows. Using formal methods, we produce correct-by-construction executable models that can be simulated in an interleaved way. The simulation draws advanced analytics from live IoT monitoring as well as an ERP system to provide predictive business intelligence. We describe our process and resource modelling efforts in the context of a collaborative project with two manufacturing partners. We evaluate our results based on the improvement of the scheduling accuracy for real production flows.

1. Introduction

Modern trends in agile manufacturing [1] put the focus on the production of small to medium size batches of high-quality machined and fabricated components. Manufacturing flows consist of sequences of discrete operations that are customised and tailored for each job. The large variability in the job specification and the involved custom operations poses an operational challenge in terms of monitoring and optimising the workload and throughput of each department as well as the entire factory. Processes both at the worker level (machine operations, movement, queuing, etc.) and at the management level (communication, scheduling, task allocation etc.) rely heavily on human decision making and regular team and cross-department communication and collaboration. Teething problems from new designs, incomplete specifications, unknown variables, changing customer requirements, and arising issues such as unexpected delays, faulty materials, required re-works, etc. exacerbate the unpredictability of the manufacturing flows. Managers are forced to make decisions under incomplete and uncertain information as they do not have a clear view of the state of the factory.

Such circumstances call for new live monitoring and predictive tools that can help manage manufacturing flows in real-time, inform decision makers, and coordinate workers within and across departments. We present a novel approach that combines rigorous, AI-based workflow modelling and management, with data-driven monitoring, and simulation-based scheduling. It is backed by a logical theory that allows for formally verified, correct-by-construction workflow models. The resulting workflow management and simulation system `WorkflowFM` [2] is highly practical, with applications in the healthcare domain [3], and has been deployed and tested at two manufacturing sites, thus spanning the full spectrum from theory to practice.

We present our experience applying a logic-based framework in practice, in close collaboration with two industrial partners. We specifically focus on a key feature of our approach, using Business Process Simulation (BPS) as a tool for predictive Business Intelligence, and particularly scheduling. Our tool excels at the simulation of multiple dynamic interleaved workflows while focusing on resource management. We adopt a broad advanced analytics view, where low level data from IoT sensors and an Enterprise Resource Planning (ERP) system are fed into the simulation automatically.

Although BPS has clear benefits on its own in the context of Business Intelligence [4], we believe our approach is a step towards incorporating new perspectives of industrial processes within BPS. For instance, recent work in this conference track and elsewhere shows how connecting IoT objects in team processes improves team effectiveness [5], and how a Social Factory where machines and people interact as users in a network can improve problem solving [6]. Our system leverages live IoT data, human-in-the-loop simulation [7], and decision support [8].

2. Background

Our project involved a collaboration between two academic institutions, namely FBK CREATE-NET[1]

[1] `https://create-net.fbk.eu/`

URI: https://hdl.handle.net/10125/70734
978-0-9981331-4-0

and The University of Edinburgh, two technology companies, namely Reply[2] and ThinkInside[3], and two manufacturing partners as described next, for two years. The goal was to develop a real-time monitoring and scheduling system, using state-of-the-art knowledge representation and reasoning techniques, to provide key insights, guide decision support and facilitate coordination. Our solution consists of three components:

1. `WorkflowFM`: an AI-based workflow modelling and management framework.
2. Real-time data provision through a combination of mining from the ERP system and IoT sensors.
3. A combined Cloud and Fog computing infrastructure for scalable deployment.

Due to space constraints, this paper focuses on the workflow management tool `WorkflowFM`, including its simulation capabilities. We describe this next, followed by an overview of the settings we encountered at our two manufacturing partners.

2.1. Rigorous Workflow Management

Workflow-based solutions for the analysis and optimisation of manufacturing business processes already exist in the context of Business Process Management (BPM) and Business Process Simulation (BPS) [9, 10, 11]. BPM models typically rely on either BPMN [12] or Petri Nets [13]. BPMN is a descriptive language without formal semantics. The correctness of BPMN models needs to be established through mappings to formal languages and post-hoc verification and validation. Petri Nets have rich semantics that allow verification [14], but their features and properties as executable artefacts are relatively limited (for instance in terms of data-driven parameterisation [11]).

We adopt a novel approach in process modelling that relies on logic-based knowledge representation and reasoning techniques [2, 15]. Specifically, we use a resource-aware logic, known as linear logic [16], to model processes based on their input and output resources including variations or exceptions. Automated reasoning is then used to put together the defined processes and form complex workflows based on logical inference [15]. This results in rigorous workflow models with mathematical guarantees of correctness with respect to (a) the consistency of the structure and resource flow across the model and (b) a systematic accounting of resources (thanks to the linearity of the logic), i.e. it is automatically ensured that no resources appear out of nowhere or disappear when unused. We suggest this is a fruitful marriage of formal verification and BPM, leading to models that have a degree of correctness by design.

In addition to that, we exploit a long established and recently evolved theory for formally verified concurrent programs [17, 18]. The theory allows the conversion of linear logic proofs to process algebras or session types. In essence, our practical implementation of this theory allows us to obtain correct-by-construction artefacts from our workflow models, which can be executed concurrently and asynchronously, with guaranteed freedom from deadlocks and race conditions. This results in an event-driven system that keeps track of the state of the workflow for each job and can react to events that happen in the factory in real time based on the intended processes captured in the model. In addition, the same artefacts can be executed concurrently in simulation, in a way that allows them to interleave naturally and without errors. This enables the simulation of patterns that only emerge through the interaction of jobs that are being processed on the manufacturing floor at the same time (see Section 3.2 for an example).

The rest of this paper focuses on the practical advantages obtained by using such a framework, starting with process analysis and modelling, all the way to quantifiable benefits. It is worth noting, however, that our logic-based models can be developed through a visual, diagrammatic tool, as demonstrated in the figures in the next section. The associated correct-by-construction executable code is extracted automatically. The technical and logical details are effectively hidden from the user, and thus the benefits mentioned above can be realised without the need for expertise in formal verification.

2.2. Academic & Technology Partners

Our academic and technology partners in this project contributed complementary assets and expertise.

ThinkInside contributed its innovative IoT location tracking technology to provide real-time data on the active production flows. We particularly exploited geofencing data showing when each flow would enter or exit a particular area or machine.

FBK CREATE-NET contributed a state-of-the-art Cloud and Fog platform where our workflow solution was deployed. This provided scalability, robustness, and fault tolerance, and enabled the integration with the IoT sensors and data streams.

Reply helped establish the relationship between the

[2] `https://www.reply.com`
[3] `https://thinkin.io/`

industry and academic partners as well as address the business challenges of converting our academic prototypes (TRL 4) to an operational system (TRL 6-9).

The composition of the consortium for this project reveals our strong drive to exploit our latest research achievements in formally verified workflow management and simulation to provide value for manufacturers in real-world settings. We believe the effort required to achieve this, particularly in software based solutions, is often underestimated, due to unforeseen gaps between the research assumptions and the actual challenges and the need for intuitive interfaces that are usable by non-technology experts.

2.3. Manufacturing Partners

In order to explore the benefits of a formal workflow approach in practice, we established a collaboration with two discrete manufacturing [19] SMEs:

Partner A is a small manufacturer of luxury and bespoke pens with a long tradition. They typically produce small batches of up to 100 pieces and cater to highly customised orders. We deployed our system in the metal department of their small factory, involving 15 areas of different human operated machines. Each machine is capable of a particular operation, such as milling, pressing, rifling, washing, polishing, etc., allowing different possible configurations depending on the requirements. Orders fall under one of 14 possible Standard Operating Workflows (see Section 3.3).

Flows are organised in production lots, which are essentially crates of materials (typically pen tubing) to be processed. A sheet of order and operation details is attached on each production lot. Operations are scheduled manually by the department head after close of each day. Scheduling decisions are primarily made based on the priority, urgency, and value of the order.

The key value they sought in our solution was the ability to a) monitor the state of their active production flows, b) detect the source of delays and minimise the overhead of communication when issues arise, and c) have better insights for informed decision making, including exploration of "what-if" scenarios.

Partner B is a medium-sized manufacturer of metal components and structures. They frequently serve as the initial step in their customers production line and prioritise reducing lead-times and offering a reliable service. They are organised into a handful of departments, each with similar machines that frequently operate in concert with one another, such as the "2D Operations" department which contains laser cutters, bending machines, and stamps. Our solution is deployed on both of their two sites, tracking their average 600 production jobs per month across all departments.

The nature of their work induces logistical challenges for management which our simulation tool is well placed to address. Jobs, whether "make-to-order" or "make-to-stock", are initially planned out by general management in the ERP system with respect to the involved assemblies and operations (see Section 3.4), including an estimate of the time required for each operation. The "Due Date" is set for each job based on the actual shipping deadline and an artificial buffer of up to several weeks to cater for potential delays and conflicts. The start time is finally calculated in a manually curated Excel formula that takes into consideration the "Due Date", the estimated operation times, manual prioritisation and tweaking, and standard working hours. A "Job Card" is printed for each job, including all the details of the work plan, specifications and a drawing of the part, and attached to the corresponding part or production lot.

Jobs within a department are organised in a "Departure Board" screen, which lists all the jobs currently assigned to that department in ascending order of the "Due Date". Ideally departure boards would be sufficient and machinists could assign themselves the jobs which appear there. However, interviewing the Team Leaders revealed that, whilst the departure boards were useful as a starting point, it was usually necessary to manually assign jobs to machinists for each shift, as a majority of jobs required that constraints of subsequent departments be taken into account for them to ship on time. For this reason, Team Leads hold meetings every few days to go through the list of jobs to be done to update each other on progress, re-assign completed work, and forecast any potential bottlenecks. The floor manager may also directly organise the highest-priority work with the relevant department as needed.

Based on these details, there is obvious need for better, real-time monitoring of production flows and tools to support decision making, such that take into consideration cross-departmental constraints, work loads, and unexpected events and delays. We believe the work described in this paper provides a framework to address this need directly.

Although our work focuses on the operational details of our two partners, we believe their practices and operating patterns and conditions are common across many SMEs of discrete and agile manufacturing. Exploring, analysing, and understanding the details and particular challenges of such organisations are key steps in our process and should not be detached from the technological solution. We devoted a sizeable amount of time and effort to this purpose, conducting interviews across different hierarchical levels in each partner,

shadowing daily operations to experience the challenges first-hand, reviewing standard operating procedures and guidelines, and exploring the management tools already in place. This gave us a detailed understanding of the challenges faced from different perspectives and ensured that we address key gaps. This is particularly important, as often general management, middle management, and machinists have very different views and needs, which can be hard to communicate or elaborate.

2.4. Methodology

Our methodology is aimed at exploiting our rigorous workflow management technology to collect analytics about manufacturing flows and inform a predictive simulation environment.

First, we **Model** the production flows in the factory as workflows. The results of this first step are discussed in Section 3. Second, we **Deploy** the modelled workflows as executable artefacts based on the rigorous theory at the core of our technology. Third, we **Track** the operations using the IoT sensors and the information logged in the ERP system. The data is processed and mapped into the workflow models. This leads into the **Monitor** step, where workflow analytics are extracted and displayed to the user in real time. This includes a timeline of events for each job up to their current status, currently running operations, machine utilisation statistics, delay and queuing analytics, notifications of deviations, etc. In the final step, we use the extracted workflow analytics to **Predict** information including scheduling, duration, delays, and costs of unfinished and new production flows by simulating the corresponding workflows. The simulation is performed in a virtual environment that models the factory assets and resources as described in Section 4. The simulation tool is described in Section 5, whereas the results are evaluated based on the accuracy of the predicted duration for each job as described in Section 6.

3. Production Flow Modelling

The main aim of our process models is to describe the operations that occur in a *production flow*. The granularity of the model can vary depending on the needs and size of the manufacturer. For instance, for Partner A we focused on a particular department of the factory that was viewed as a bottleneck due to key operations happening there, whilst for Partner B end-to-end models were deemed more valuable in terms of optimising cross-departmental collaboration.

In any case, we view a production flow as an instance of a particular set of operations that need to be applied to fulfil a particular job, whether it is a *make-to-order* or a *make-to-stock* type of job. We design our models drawing from various sources, such as standard operating procedures (SOPs), operation planning retrieved through an ERP system, policies and guidelines established by management, and observation of informal but standard practices on the floor.

Based on these sources, we developed two types of workflow models:

1. *Standard Operating Workflows (SOWs)*: These are concrete process models describing the processes that *must* be followed, as normally determined in a SOP document.

2. *Assembly Workflows*: These are dynamic models mined from an ERP system and based on the assembly planning performed in advance on a per-job basis by management.

In the next sections we describe some of the patterns of operations that we encountered and incorporated in our models.

3.1. The Reserve-Move-Perform Pattern

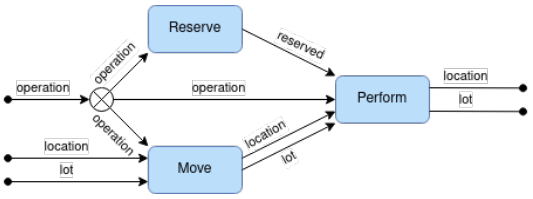
Throughout the production flows examined in this work, we observed a common pattern representing how operations that use a machine on a physical object are performed:

1. *Reserve*: First, a worker makes the decision to work on a particular job on a machine. This effectively reserves the machine for that particular operation.

2. *Move*: The production lot is moved from its previous location to the position of the machine. This includes any buffering that needs to happen until the machine is reserved for the particular job.

3. *Perform*: Once the machine is available for use and the object is in the right position, the worker can perform the operation.

The workflow diagram for this pattern is shown in Figure 1. It is worth re-iterating here that all our workflow diagrams correspond to formal, fully-validated logical models. This is showcased in the same figure, where we include the corresponding linear logic proof that verifies the correct flow of resources across the workflow. The automatically generated, verified process calculus term below the proof constitutes a specification for asynchronous execution based on message passing between named channels. As previously mentioned, both the logical and process calculus models are hidden from the user.



$$\dfrac{\dfrac{}{\vdash op^{\perp},\ res}\ Reserve \qquad \dfrac{}{\vdash res^{\perp},\ op^{\perp},\ loc^{\perp},\ lot^{\perp},\ loc \otimes lot}\ Perform}{\dfrac{\vdash op^{\perp},\ op^{\perp},\ loc^{\perp},\ lot^{\perp},\ loc \otimes lot}{\vdash op^{\perp},\ op^{\perp},\ loc^{\perp} \parr lot^{\perp},\ loc \otimes lot}\ \parr}\ Cut \tag{1}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash op^{\perp},\ loc^{\perp},\ lot^{\perp},\ loc \otimes lot}\ Move \quad (1)}{\vdash op^{\perp},\ op^{\perp},\ op^{\perp},\ loc^{\perp}, lot^{\perp},\ loc \otimes lot}\ Cut}{\vdash op^{\perp},\ op^{\perp} \parr op^{\perp},\ loc^{\perp}, lot^{\perp},\ loc \otimes lot}\ \parr}{\vdash op^{\perp} \parr op^{\perp} \parr op^{\perp},\ loc^{\perp}, lot^{\perp},\ loc \otimes lot}\ \parr \tag{2}$$

$$\dfrac{\dfrac{}{\vdash op^{\perp},\ op \otimes op \otimes op}\ CopyOp \quad (2)}{\vdash op^{\perp},\ loc^{\perp}, lot^{\perp},\ loc \otimes lot}\ Cut \tag{3}$$

$$ReserveMovePerform(cLot, cLoc, cOp, cOut) =$$
$$(\nu\ z_9)((z_9(cReserveOp, z_7).z_7(cMoveOp, cPerformOp).$$
$$(\nu\ z_5)((z_5(cPerformLoc, cPerformLot).$$
$$(\nu\ z_2)((\mathbf{Perform}(z_2, cPerformOp, cPerformLoc,$$
$$cPerformLot, cOut) \mid\mid \mathbf{Reserve}(cReserveOp, z_2))) \mid\mid$$
$$\mathbf{Move}(cMoveOp, cLot, cLoc, z_5))) \mid\mid CopyOp(cOp, z_9)))$$

Figure 1. Reserve-Move-Perform Diagram, Linear Logic Proof, and π-calculus Specification

The edges in this diagram represent the different resources associated with the particular workflow instance. More specifically, `operation` is the operation to be performed, which includes what machine is required and its location, `location` is the current location of the production lot, `lot` is a description of the production lot including a reference to its corresponding job, and `reserved` is a successful machine reservation.

3.2. Buffering

Understanding the subtleties of how people work and collaborate in practice is, in our view, a crucial step towards creating accurate and useful workflow models. Errors and suboptimal processes often occur not at the level captured in an SOP document or ERP entry, but rather in the daily, informal, practical human decisions.

One particular example of process that is often neglected in optimisation but can have major effects in the processing of a production flow is *buffering*. This involves the temporary storage of a production lot when we need to pause its operations, for example due to a machine being used on another job (queuing) or needing to be configured or maintained before the next operation, or another production lot in the assembly being delayed.

An example can be seen in Figure 2. The first two panels show individual production flows in a floor with three machines. The third panel shows both production flows running at the same time. Both the red and yellow production lots need to use Machine A, so the red lot is delayed. This leads to the blue lot being buffered as it waits for the delayed red lot before it can continue.

Typical business process models tend to focus on the key operational steps and do not model seemingly unimportant intermediate steps such as buffering (see example in [20]). BPS models commonly assume queuing has no operational effect and queued jobs can start immediately when the resource is released, putting focus on the queuing and prioritisation policy instead [4]. Moreover, BPS systems often neglect interactions between different, interleaved workflows or process instances [9, 4], such as buffering.

The reality we observed is very different. Buffering typically involves moving the production lot to temporary storage, which could either be a designated buffering area, or as informal as any empty space in the factory. A production lot may be buffered anywhere from a few minutes to several weeks until it is ready to be moved to the next position in the production flow. Based on this, we have identified three common examples of how buffering can affect a production flow.

First, moving production lots takes time and effort. It may, for instance, require specialised lifting equipment or trained moving personnel that need to be available for the move. Delays in moving a lot can cause cascading ones in other operations that depend on it.

Second, production lots may be misplaced in their buffering positions, especially if there is no specific designated temporary storage space in the factory. Locating the needed production lot may also cause cascading delays in the process.

Third, decisions relating to buffering can also be sub-optimal and may need to rely on both intra- and inter-departmental communication and experience. For instance, a production lot may be located in a particular machine where an operation has just concluded. It may be the case that no other jobs are queued for that machine until the time the production lot is due to be moved to the next position. In that case, moving it to a buffering position may be a waste of time and effort.

These examples demonstrate the importance of incorporating buffering processes in any workflow model. In our models, we accomplish this by extending the Reserve-Move-Perform pattern from Figure 1 as shown in Figure 3. We have specifically modelled 2 occurrences of buffering: one when the next position is busy with another job and one when a machine is being configured before it can handle the current job.

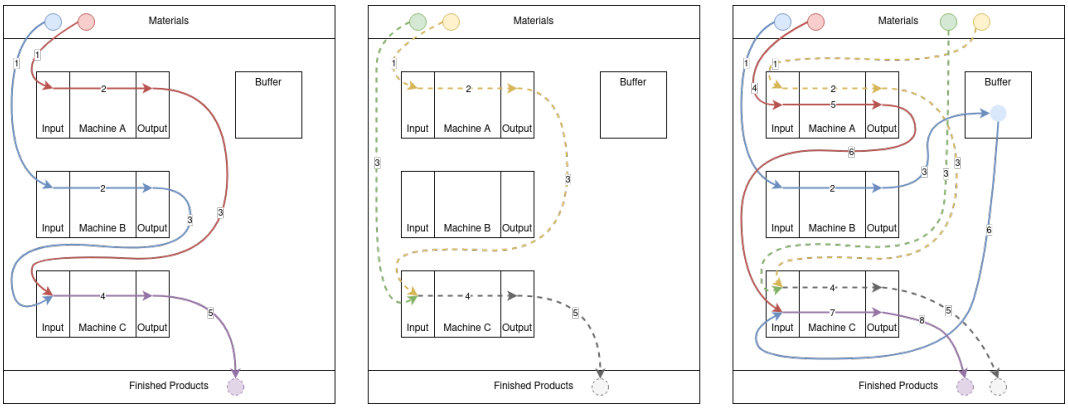


Figure 2. Buffering observed when interleaving 2 production flows: the blue lot must wait for the delayed red lot.

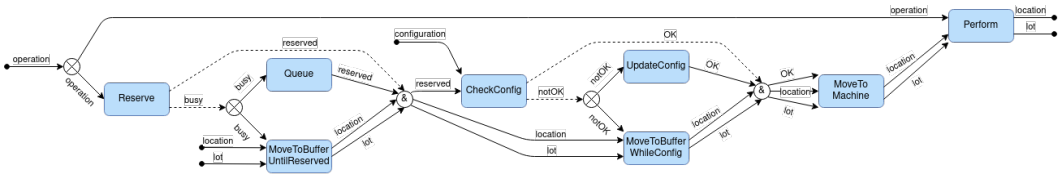


Figure 3. RMP with Buffering

3.3. Standard Operating Workflows (SOWs)

We define a SOW to be a workflow model of a predefined sequence of manufacturing operations that accomplishes a particular production task. These models are particularly suited for manufacturing flows that follow specific, rigid patterns of operations, as opposed to a highly dynamic environment where each job requires a different operation sequence. In this setting, jobs can be customised through the various parameters and configurations for each operation.

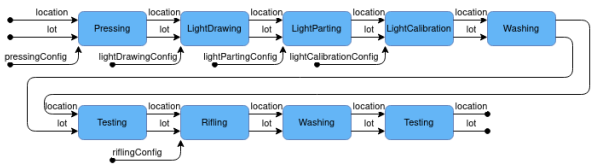


Figure 4. Standard Operating Workflow

An example of a SOW model is shown in Figure 4, detailing Partner A's clip manufacturing process as a sequence of operations shown in blue, each a collapsed version of the RMP pattern.

3.4. Assembly Workflows

In agile manufacturing, each job may require its own custom production flow. In such a setting, production flows are typically organised into *assemblies*, where production lots and materials are pulled together and have a sequence of operations performed on them. Each production lot in an assembly could in turn also be a product of another assembly. In that sense, each assembly may contain *subassemblies* representing how some of the components are assembled before being combined. This forms an assembly tree (see Figure 5), where each node is an assembly with its sequence of operations and its children are its subassemblies. The leaves of the tree are assemblies which only use raw resources and have no subassemblies.

The planning of operations and assemblies involved in a job is typically done by the management team and recorded in an ERP system (in our case Epicor - `https://www.epicor.com`). We draw from that information to construct workflow models of assemblies based on the general model shown in Figure 5. Each workflow takes an assembly job as an input and extracts its components: a) the initial location of assembled resources, b) initial production lot descriptor, c) list of operations to be performed, and d) list of assembly jobs corresponding to direct subassemblies.

Once the main assembly job is split into its components, we spawn a new instance of the assembly workflow for each subassembly, thus recursively building the assembly tree. Once all subassemblies of an assembly are complete, they are assembled at a location and given a unified lot descriptor from the job. The sequence of operations is then performed on the unified lot, resulting in an updated lot descriptor and location, as well as an (empty) list of remaining operations which

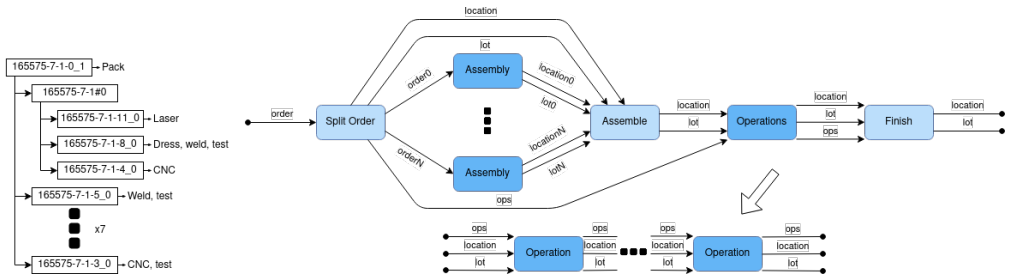


Figure 5. Example Assembly Tree and Model Diagram

is then consumed by the last process.

Each operation is performed in sequence, using the Reserve-Move-Perform pattern we described previously. Operations are chained in a sequence to form the `Operations` process in Figure 5.

Using this model, we are able to capture a large range of flows in a real industrial setting, using data exported from the ERP system. The data was processed to convert it from a flat structure to the assembly tree that matches the structure of our workflow model.

4. Resource Modelling

In order to produce an accurate simulation of the production flows, we need to create a simulation environment that models the operational resources of the factory, including assets such as machines and equipment, and human resources.

Machines and equipment are modelled based on their numbers and their features. Each machine is typically able to handle a specific *type of operation*, such as "milling", "welding", or "laser cutting". The exact operation depends on the specification of the job and may require a particular *configuration* of the machine. It is also worth noting that two machines of the same type may differ in what operations they can handle based on their available configurations. For instance, a newer "laser cutting" machine may be able to perform more precise cutting or cut smaller parts.

Typically each machine can handle one operation at a time, which is reflected in the simulation. However, some cases do not fall under the same "one job per machine" assumption.

One such case that we encountered is *nesting*. This involves the appropriate combination of multiple cutting tasks on the same sheet of metal [21] to minimize raw material waste from unused metal. This effectively means that multiple jobs are handled by the same, single operation and in the same machine. Moreover, a single job may be spread across multiple operations (nests).

Although nesting software tools that optimise this process do exist, the prioritisation and scheduling of the jobs to be nested remains largely a human task. Since a nested operation starts at the same time for all involved jobs, this introduces an important trade-off in the scheduling decision, as the nesting process can cause delays to some of the jobs. This makes incorporating nesting in the simulation model a significant challenge.

Human resources are modelled not only based on their capabilities, but also on their working hours, including allocated shifts and excluding weekends, holidays, and leaves of absence. One can further increase the sophistication of the simulation model by incorporating the skill level of each worker in terms of speed of operations, and job handovers between shifts.

5. Simulation

BPS [22] is a valuable business intelligence tool for workflow decision support. For instance, it can be used to explore what-if scenarios and estimate the effect of different decisions, including operation scheduling, in a virtual, risk-free environment.

Our simulator relies on discrete event based simulation [23] over individual, interleaved workflow instances. In short, each simulated case consists of a workflow instance that unfolds over (virtual) time, generating events (such as tasks starting or finishing) in discrete time. All cases are simulated concurrently so that they can make use of the same resources and observe and react to the current state at runtime.

In this particular case, we have set up a simulation environment for manufacturing workflows. This includes the following user configurable elements:

- *Operation Types:* These capture groups of similar operations, such as "milling" or "laser".
- *Locations:* Machine locations are used to determine if a production lot needs to be moved.
- *Machine Groups:* The user can define a group of machines that can handle a particular operation type and have a set capacity for concurrent jobs, typically corresponding to the number of available

machines. More complex configurations such as nesting cannot be modelled in this setup, and are instead simplified as an infinite capacity group.

- *Operation Costs:* The user can configure a fixed cost for each operation and a cost per unit of time for each machine.
- *Human Resources:* The standard working hours, shifts and holidays of workers can be customised by the user. Workers can also be modelled individually, though we have not pursued this level of detail in our current models yet.

Data available in the ERP system, such as assembly tree structures, estimated operation times, worker shifts, and costs are fed automatically into the appropriate simulation parameters. Live operation and workflow analytics from the monitoring stage, including the processed IoT feed, can also help inform simulation parameters, such as the estimated duration of a particular movement, unexpected events and delays that have occurred (often leading to rescheduling), and the state of machines across the factory.

The production flows to be simulated can be introduced in two ways:

1. Manually through the web interface, by selecting a workflow skeleton corresponding to a SOW and its associated operations (or machine *configurations* in this context).
2. Automatically based on scheduled and unfinished jobs in the ERP system. The duration of each operation is set to be the estimated duration specified by the manager when the job is entered in the ERP system.

The simulation results in three categories of data:

1. *Flows:* includes start and end times of each job, total delay and cost, and number of operations performed.
2. *Operations:* includes start and end times of each operation, delay and cost, and resources used.
3. *Resources:* includes total busy and idle time of each machine or worker, number of tasks performed, and total operational cost.

The results are visualised in a timeline, as shown in Figure 6, which depicts a predicted sequence of operations for each available resource (machine). An alternative view shows the timeline of operations per job. These effectively provide an optimised schedule of operations. Hovering above an operation reveals further details, including its delay and cost. The data is also made available in raw format for any further processing.

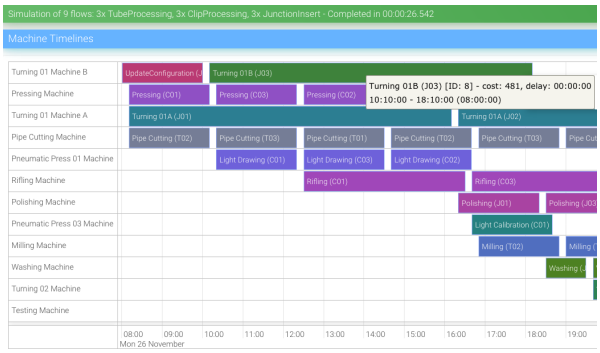


Figure 6. Excerpt of an example predicted timeline for 9 SOWs.

6. Results

A key goal of our workflow simulation is to provide predictive scheduling capabilities that improve upon the current practices at our industrial partners.

As mentioned in Section 2.3, managers in both our collaborating manufacturers take a manual approach to scheduling. They calculate an estimated duration of all the operations and subassemblies required for each job. This data is imported from the ERP system into a custom spreadsheet. They then schedule the appropriate start date based on the job due date. the estimated duration, and some buffer time. One of the main challenges for this type of scheduling problem is that of coordinating many different assets (machines, human resources, restrictions, etc.) across multiple jobs that are happening at the same time. Our workflow framework addresses this problem by allowing the concurrent simulation of thousands of jobs over a set of virtual assets.

More specifically, using data collected directly from the manufacturers, we built a simulation environment containing some of their assets (machines) and their respective capacities and restrictions. We used this to evaluate the predictive capabilities of our simulator compared to the manual scheduling using historical data from the ERP system of one of the manufacturers.

For each job in the data set we extract a) its starting date d_s as the date the job was entered in the system, which is also the minimum starting time of the operations, b) its manually estimated completion date d_m, and c) its actual completion and delivery date d_a.

We also extract the assembly tree of all the involved assemblies and operations, including the *manually estimated* duration of each operation. In this way, we only allow our simulator access to the same information a shop floor manager has when they manually schedule the jobs. In future work, this can be further improved by incorporating a machine learning model that better

predicts the duration of each individual operation.

The assembly trees of all the orders in the dataset are then automatically converted into assembly workflows and simulated with the given starting date d_s. The simulator predicts a completion date d_p for each job. We then compare the accuracy of that prediction against d_m compared to d_a.

First, we calculate the manual error $\epsilon_m = d_m - d_a$ and the prediction error $\epsilon_p = d_p - d_a$ in days. Our evaluation metric ϵ is the percentage of improvement of ϵ_p over ϵ_m, where $\epsilon = \frac{\epsilon_m - \epsilon_p}{\epsilon_m}$.

For example, assume an order took $d_a = 5$ days to actually fulfil (starting at $d_s = 0$). Also assume the manual estimate was $d_m = 1$ day, whereas the simulator predicted $d_p = 3$ days. We have $\epsilon_m = 4$ and $\epsilon_p = 2$, therefore an improvement of $\epsilon = \frac{4-2}{4} = 50\%$.

A *positive* result, i.e. $\epsilon > 0$, indicates the simulator was better than manual scheduling at predicting the completion date. Specifically a value of $\epsilon = 100\%$ indicates the simulator was able to predict the exact time of actual completion. A *zero* result $\epsilon = 0$ indicates that the simulator did not improve over manual scheduling. A *negative* result $\epsilon < 0$ indicates the simulator did worse than manual scheduling.

For our evaluation, we extracted and simulated a dataset of **6885** jobs that were due in 2018 for **Partner B**. The mean actual duration of these jobs was **56** days, with 90% being under 100 days and 75 jobs (1%) lasting longer than a calendar year. On average across the board, our simulator improved the scheduling prediction by **16.1%** compared to current practice.

Figure 7 shows a comparison of accuracy between the manual scheduling (blue) and our simulator's predictions (orange) measured in 15-hour working days. The closer the values are to zero, the smaller the error, corresponding to a more accurate prediction of the real finish times. Positive values correspond to the predicted schedule overestimating the actual finish time of the involved jobs. Extreme negative values corresponding to very large delays (up to almost 2 working years) are also observed. So far, we have been unable to recover sufficient data to explain the reason for such long delays, but we speculate that these correspond to make-to-stock jobs that were put on hold manually as needed.

The results on the error improvement ϵ can be broken down as follows:

1. *Positive results*: an average improvement of **32.8%** over **3807 jobs (55.3%)** with a maximum of 99% (near perfect prediction). A histogram of the positive results is shown in Figure 8.

2. *Zero results* for **2788 jobs (40.5%)**. This is due to the fact that our model does not yet have a complete map of all the assets in the shop floor, due to their high complexity. Lacking sufficient information, the simulator effectively falls back to the human prediction.

3. *Negative results*: a worse prediction for **289 jobs (4.2%)**. These were mainly repair and maintenance jobs that are flagged as regular jobs in the ERP system, and thus our system queues them instead of prioritising them immediately (as would happen in reality). A histogram of the negative results is shown in Figure 8.

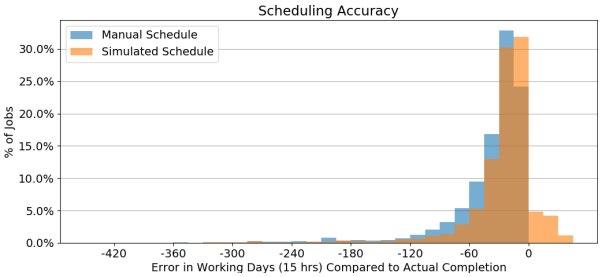


Figure 7. Manual and Simulated Schedule Accuracy

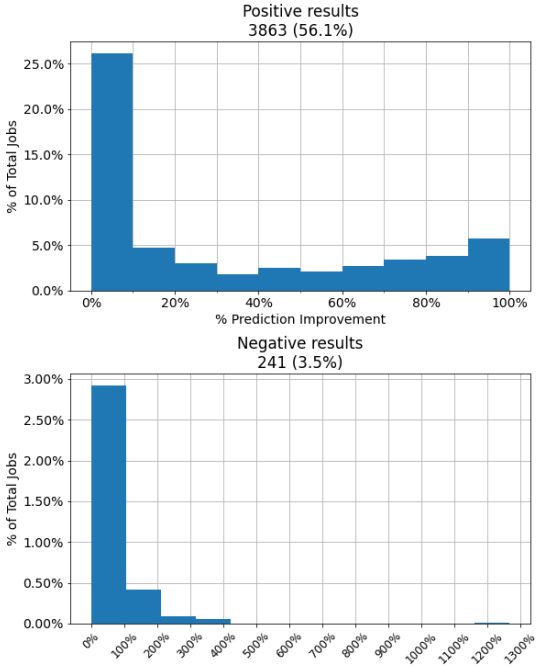


Figure 8. Positive and Negative Results

These results can be further improved as we iterate and expand our model and gather more data. This includes capturing more asset restrictions in the factory, such as human resource availability, and more accurately flagging special types of jobs such as maintenance operations and nested operations.

7. Conclusion

Our system tightly integrates business process modelling, anchored in a formal methods approach, and simulation as a key analytics and decision support tool. We can create scenarios that are faithful to the shop floor process workflows, yet are risk-free, and thus come up with real rather than merely hypothetical improvements to day-to-day operational decisions. Our novel simulation approach combines a formally verified interleaved execution of workflows with live data extracted directly from our IoT and workflow monitoring platform and the ERP system. Test results for almost 7000 real manufacturing jobs show that even with a partial resource model of the factory and noisy ERP data, our system improved over manual scheduling for more than half of the cases, with perfect accuracy for some. There is clear room for further improvement by iterating our workflow and resource models, and extending our analysis to measure waiting times and detect bottlenecks. This will require the same continuous and committed observation and interaction with our manufacturing partners.

Acknowledgements

This work was part of the *"DigiFlow: Digitizing Industrial Workflow, Monitoring and Optimization"* innovation activity funded by EIT Digital. We would like to thank our project partners FBK CREATE-NET, ThinkInside, and Reply as well as the participating manufacturers. We would also like to thank the anonymous reviewers for their constructive feedback.

References

[1] Y. Yusuf, M. Sarhadi, and A. Gunasekaran, "Agile manufacturing:: The drivers, concepts and attributes," *International Journal of Production Economics*, vol. 62, no. 1, pp. 33 – 43, 1999.

[2] P. Papapanagiotou and J. Fleuriot, "WorkflowFM: A logic-based framework for formal process specification and composition," in *Automated Deduction – CADE 26*, pp. 357–370, Springer, 2017.

[3] P. Papapanagiotou and J. D. Fleuriot, "Formal verification of collaboration patterns in healthcare," *Behaviour & Information Technology*, pp. 1–16, 2013.

[4] W. M. P. van der Aalst, *Business Process Simulation Survival Guide*, pp. 337–370. Springer, 2015.

[5] Z. R. Wang and D. Kennedy, "The internet of things, teamwork, and service projects," in *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[6] L. Kassner, P. Hirmer, M. Wieland, F. Steimle, J. Königsberger, and B. Mitschang, "The social factory: connecting people, machines and data in manufacturing for context-aware exception escalation," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[7] S. Narayanan and P. Kidambi, "Interactive simulations: History, features, and trends," in *Human-in-the-Loop Simulations: Methods and Practice* (L. Rothrock and S. Narayanan, eds.), pp. 1–13, Springer, 2011.

[8] R. Pinto, T. Mettler, and M. Taisch, "Managing supplier delivery reliability risk under limited information: Foundations for a human-in-the-loop dss," *Decision Support Systems*, vol. 54, no. 2, pp. 1076 – 1084, 2013.

[9] K. D. Barber, F. W. Dewhurst, R. Burns, and J. Rogers, "Business-process modelling and simulation for manufacturing management," *Business Process Management Journal*, vol. 9, no. 4, pp. 527–542, 2003.

[10] M. Dong and F. F. Chen, "Petri net-based workflow modelling and analysis of the integrated manufacturing business processes," *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 9-10, pp. 1163–1172, 2005.

[11] A. Senderovich, A. Rogge-Solti, A. Gal, J. Mendling, A. Mandelbaum, S. Kadish, and C. A. Bunnell, "Data-driven performance analysis of scheduled processes," in *Business Process Management*, pp. 35–52, Springer, 2015.

[12] Object Management Group, "Business Process Model and Notation (BPMN), version 2.0," 2011.

[13] W. Reisig, *Petri Nets: An Introduction*, vol. 4. Springer, 1 ed., 1985.

[14] W. Aalst, van der, "Challenges in business process management : verification of business processes using petri nets," *Bulletin of the EATCS*, vol. 80, pp. 174–199, 2003.

[15] P. Papapanagiotou and J. Fleuriot, "A pragmatic, scalable approach to correct-by-construction process composition using classical linear logic inference," in *Logic-Based Program Synthesis and Transformation*, pp. 77–93, Springer, 2019.

[16] J.-Y. Girard, "Linear logic: its syntax and semantics," in *Advances in Linear Logic*, no. 222 in LMS Lecture Notes Series, Cambridge University Press, 1995.

[17] S. Abramsky, "Proofs as processes," in *J. Theoretical Computer Science*, vol. 135, pp. 5–9, 1994.

[18] H. DeYoung, L. Caires, F. Pfenning, and B. Toninho, "Cut Reduction in Linear Logic as Asynchronous Session-Typed Communication," in *EACSL 2012*, vol. 16 of *LIPIcs*, pp. 228–242, 2012.

[19] B. L. Dietrich, "A taxonomy of discrete manufacturing systems," *Operations Research*, vol. 39, no. 6, pp. 886–902, 1991.

[20] M. Khabbazi, M. Hasan, R. Sulaiman, A. Shapi'i, and A. Zadeh, "Business process modelling in production logistics: Complementary use of BPMN and UML," *Middle-East Journal of Scientific Research*, vol. 15, pp. 516–529, 2013.

[21] J. F. C. Oliveira and J. A. S. Ferreira, "Algorithms for nesting problems," in *Applied simulated annealing*, pp. 255–273, Springer, 1993.

[22] K. Tumay, "Business process simulation," in *Winter Simulation Conference Proceedings, 1995.*, pp. 55–60, IEEE, 1995.

[23] P. J. Kiviat, "Digital computer simulation: computer programming languages," tech. rep., RAND Corporation, 1969.