Assessment of Risks in Manufacturing Using Discrete-Event Simulation

Renaud De Landtsheer[1], Gustavo Ospina[1], Philippe Massonet[1], Christophe Ponsard[1(✉)], Stephan Printz[2], Sabina Jeschke[2], Lasse Härtel[3], Johann Philipp von Cube[3], and Robert Schmitt[3]

[1] CETIC Research Centre, Charleroi, Belgium
{rdl,go,phm,cp}@cetic.be
[2] Institute for Management Cybernetics (IfU), RWTH Aachen University, Aachen, Germany
{stephan.printz,sabina.jeschke}@ifu.rwth-aachen.de
[3] Fraunhofer Institute for Production Technology (IPT), Aachen, Germany
{lasse.haertel,philipp.von.cube,robert.schmitt}@ipt.fraunhofer.de

Abstract. Due to globalisation, supply chains face an increasing number of risks that impact the procurement process. Even though there are tools that help companies address these risks, most companies, even larger ones, still have problems for adequately quantifying the risks on their current process as well as on alternative process. The aim of our work is to provide companies with a software supported method for quantifying procurement risks and establishing adequate strategies for risk mitigation at an optimal cost. Based on the results of a survey on risk management practices and industrial needs, we developed a tool that enables them quantifying these risks. The tool makes it easier to express key risks via a process model that offers an adequate granularity for expressing them. A simulator incorporated in our tool can efficiently evaluate these risks through Monte-Carlo simulation technique. Our main technical contribution lies in the development of an efficient Discrete Event Simulation (DES) engine, together with a Query Language that can be used to measure business risks from the simulation results. We show the expressiveness and performance of our approach by benchmarking it on a set of cases that are taken from industry and cover a large set of risk categories.

Keywords: Discrete Event Simulation · Manufacturing · Supply chain · Procurement risks · Risk management

1 Introduction

Companies are faced with increasing procurement risks within the context of a global economy. These risks can be related to many different factors, such as, for example, the geographic location and the political and economic situation of the involved parties (suppliers, warehouses and factories). Assessing these risks

B. Vitoriano and G.H. Parlier (Eds.): ICORES 2016, CCIS 695, pp. 201–222, 2017.
DOI: 10.1007/978-3-319-53982-9_12

alone is a difficult task as the risks may only reveal themselves at the end of the production chain and it also requires consideration be given to the impact of internal risks, as well as thought be given to the complexity of the manufacturing process (which could decrease the capacity for adaptation in the case of supply failure) or the level of optimisation in place (which could increase the impact in case of disruption).

Helping company managers make the right decision in the face of risks is not an easy task. Whilst analytical reasoning is impractical, model-based simulation has proved to be a competent approach [1]. Procurement risks present extra challenges as they occur at one end of the process (output), but can sometimes only be measured at the other end (input), therefore they require thought throughout the whole manufacturing process. The scope of our work is to address this challenge, focusing on small and medium enterprises in the field of mechanical engineering.

Our ultimate goal is to produce a user-friendly, tooled methodology that will guide the user through the whole process of risk assessment. In order to reach this goal, our work is structured as follows:

- A taxonomy of supplier and internal risks is identified, starting with the simplest risk - a shortage of raw materials, which can eventually lead the whole process chain to more elaborate risks, depending on the kind of order policy used.
- A survey was conducted on the practice of risk evaluation in an industrial context [2]. The results of this survey showed that nearly 66 % of the companies perform risk evaluations, although only 10 % rely on dedicated software tooling. This means that in practice risks are evaluated by an individual estimation of the cost factor and the probability of occurrence. In general, and by including historical data in the estimation, the quality of the estimation improves. However, relying on historical data and estimating the impact of factors, like delivery times, means that material quality is impossible to estimate, even in the case of changing suppliers or adding parallel processes to the chain. Based on the requirements identified in the survey, a software based risk management framework was defined.
- We developed a modelling and simulation toolset for identifying risks, quantifying them and deciding on design alternatives that can help mitigate risks. The main technical scope of this paper is to detail our toolset framework and show how it helps focus be retained on the risks during modelling, so as to stay efficient during the modelling time, simulation time and result analysis time.
- Finally, we are validating our work through a group of companies that are already trying out our tool via an easy to use web interface. Although this validation is not yet complete, we can already benchmark our approach on a number of industrial cases and in doing so assess the expressiveness and performance of our approach.

Our modelling framework includes concepts such as *storages*, where items can be stored or retrieved with a maximum capacity, and several types of production

processes, each with different timing and failure behaviours. In addition, we defined a query framework on models that are fully declarative and include arithmetic, temporal and logic operators, as well as basic probes for the elements of our factory model (contents of a storage, whether a process is running or not, etc.). Based on this query language, the software tool is able to calculate the probabilities of different scenarios (e.g. delay in deliveries, defective parts or poor quality) and their impact, which is all based on a timed model of the relevant factory process.

This approach to monetary risk quantification is based on an approach developed in the Q-Risk project [3]. The simulation toolkit relies on the discrete event simulation module of the OscaR framework, particularly its simulation layer, and adds dedicated abstractions that are dedicated to the timed modelling of factories, and the modelling of risk-related queries [4].

Our main contribution lies not only in the risk-driven dimension of our framework, but also in regards to the usability factors. Its design is based on a number of trade-offs between the expressiveness and simplicity of the modelling language, as well as efficiency of the simulation engine.

The paper is structured as follows: Sect. 2 presents the context of our work; Sect. 3 presents our modelling language for representing factories; Sect. 4 presents our query language that can serve to evaluate risks; Sect. 5 illustrates how complex risks can be included in our query language; Sect. 6 describes our DES prototype and more specifically the Query Engine. Section 7 shows the benchmarking of our simulation tool both on the expressiveness and performance dimensions; Sect. 8 discusses some related work; Sect. 9 concludes the paper.

2 Background

In order to assess and quantify different kind of risks in manufacturing processes, we model the manufacturing process as a flow graph. This model captures the key procurement steps and the production process itself. Resource storage (in warehouses or stockrooms) and the flow of raw materials in basic processes will be explained in Sect. 3.1. The main graphical notations implemented by the graphical part of framework are shown in Fig. 1, which is a model used later in the benchmarking process. Notations are quite self explanatory: a supplier is a little truck, storage types are represented by different variations of cylinders (the one with vertical bars can overflow) and processes are depicted with the industry icon (also with some variants: multiple horizontal lines means parallel batches, the cross means possible failure, the rounded, the rounded box depict a conveyor belt).

The operation of the whole manufacturing process can be described as a sequence of timed events. For instance, the first event involved with simulating a factory is fetching some materials for storage. This action can trigger a new order being sent to a supplier if the storage level reaches a certain threshold - in accordance with supply chain policy.

In the rest of this section we recall the nature of risks and the goal of risk management, then we give some details about Discrete Event Simulation and

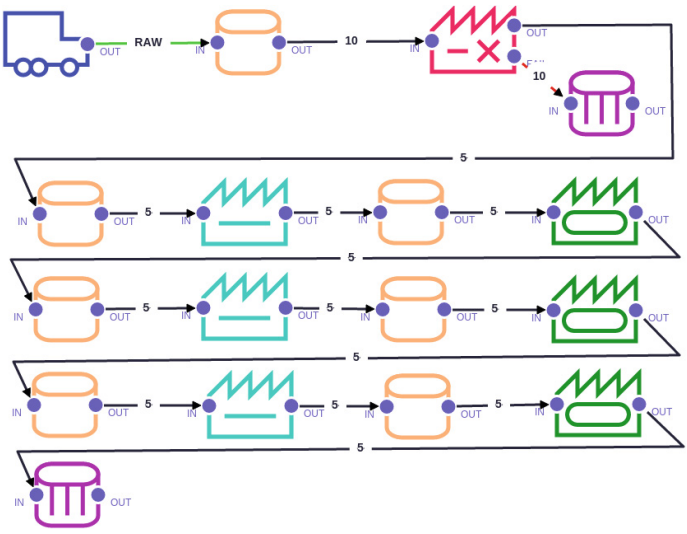

Fig. 1. Beer game model.

why it provides an adequate framework for modelling the operations of manufacturing processes and quantifying risks [5].

2.1 Risks and Risk Management

In order to develop a Discrete Event Simulation approach for quantifying the impact of risks in manufacturing enterprises, the nature of risk and the underlying process of risk management needs to be understood in detail.

Risk. Risks strongly affect an enterprise's business success and are directly related to costs, effort and yield [6]. Thereby, risk is understood as an event likely to occur with an undesired consequence. The most common and, for the approach, most convenient categories of risks are the cause and impact-oriented type. The root-cause-oriented approach considers information uncertainty and validity as risk [7]. The chance of not meeting a planned target is understood as an impact-oriented risk. However, only combining both categories of risk leads to the necessary scope of information needed to properly manage risks. Hence, risks need to be understood as having a certain likelihood of missing a defined target. Hence, the concept of risk is defined through three components: the *hazards*, or potential dangers, the *consequences* of those hazards, and their predicted frequency, or *likelihood* [8]. A "natural" quantification of the hazards associated with a risk is the quantification of product consequences in order of likelihood. A cybernetic model of procurement-based hazards and their management is presented in [9].

Risk likelihood can be modelled with probability distributions [10], as the occurrence of a risk hazard in a process or system is *uncertain*. In [11], a theory of probabilistic risk analysis is developed, which is associated with the concept of system reliability. As a risk is defined as the deviation from a planned value, statistical measures can thus be applied to operationalise and compare the possible magnitude of such deviations [12]. Evaluation of the risk analysis and the reliability of a system can be done with the Monte-Carlo method [1].

Risk Management. The main objective of risk management lies in the assessment of major corporate goals in regards to risk policy strategies. Hence, risks affecting long lasting business success need to be controlled. However, enterprises will never be able to totally eliminate risks and will always have to consider a certain degree of residual risk [13]. One key task of risk management is to identify and analyse risks as early as possible in order to take cost optimal risk treating actions [6].

The basic process of risk management (Fig. 2) is described in standards ISO 31000 and ONR 49000 ff. IEC 31010 provides an overview of corresponding risk management methods and techniques for a specific process.

2.2 Discrete Event Simulation

There are two main approaches for representing time if we want to simulate the behaviour of a system: the first approach is to use a *continuous time*, in which the events affecting the system occur as time "ticks", all of which is proportional to the actual expected time of system operation. The other approach is to have a *discrete time*, and concentrate the simulation on only the operational events, rather than the time. This is the basis of Discrete Event Systems (DES).

In the literature [14,15], the main components of a DES model are described as: *entities* (which are the items that are flowing and transformed throughout the

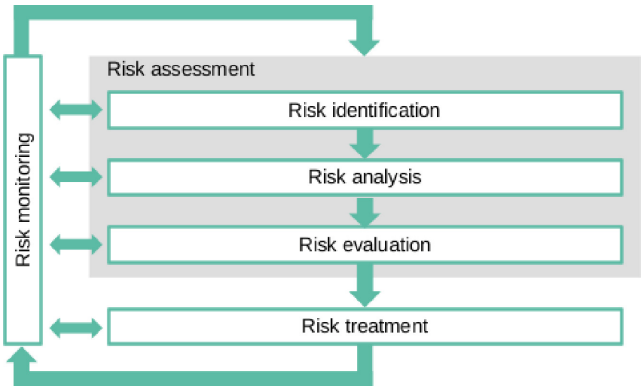

Fig. 2. Risk management process.

simulation), *queues* (representing storage devices or other areas in which entities wait to be used), *activities* (that actually perform some work on the entities), and *resources* (a special kind of entity that is required to operate activities).

DES models define *events* as discrete points in time in which the system state changes. The simulation of the model is simply a queue of different events triggered by the previous event, in other words the "next-in-time". Checking an event can trigger other events in the queue. For instance, checking the event starting an activity will trigger the events of fetching the corresponding entities needed to perform the activity, and thus ending the activity. The event of activity failure can also be triggered by a given probability.

Several software solutions exist to support DES based modelling for a variety of applications. Among the commercial software, we can cite AnyLogic [16], Arena [17] and Plant Simulation [18].

3 A Simulation Meta-Model for Factories

All the main elements of manufacturing processes are represented in our simulation meta-model, which allows us to define concrete models that are simulated in a Discrete Event Simulation engine. In addition to this, we designed a Query Language over concrete simulations in order to collect and analyse data.

3.1 Modeling Factory Processes

This section introduces the basic blocks for representing factories. In our approach, factories are modelled as a flow of items, processes and stocks.

Storages represent any kind of stock device or place for raw materials, like a warehouse, a barrel, a silo or a dumpster. They have a maximum capacity. When this capacity is reached, they either overflow, or create a blockage in the process, all of which is dependent on the settings of the storage. If a full stock storage overflows, any unloading material of that stock is lost.

Batch processes are factory processes that work in a batch fashion; supplies are collected from various stocks, then the process runs for some time, and finally the produced outputs are dispatched to their respective stocks before this whole cycle starts again.

Continuous processes are factory processes that typically run on a conveyor belt. Items are continuously picked from input stocks and undergo the process immediately at the physical end of the machine, they then pass through the machine in a queue, and when they reach the other end of the machine, the resulting items are dispatched to their respective stocks. A simple example is a conveyor belt that passes through a bakery oven; raw pastries are set on one end of the conveyor belt; they go through the oven and are cooked when they reach the other end of conveyor belt, from there they are dispatched to their output storage.

Splitting processes are similar to batch processes, except that they have several sets of outputs and when completed, one set of outputs is selected and the

produced items are dispatched to the stocks associated with the selected output. This represents a quality assurance process, whose item flow is split into two (or more) separate flows based on the result of the quality assurance analysis.

Parallel processes are variants of the processes above, where several lines of the same process are running in parallel. Basically, all processes introduced here have a parameter specifying the number of process lines running in parallel.

Items flowing in processes and stocks are indistinguishable at any given point of the factory process as they all share the same part number. Yet, they have some intrinsic features: some items might come from a given process, others might be made out of poor quality supplies, etc. These intrinsic features can influence the behavior of some processes, such as the splitting process representing a quality assurance process. This notion of intrinsic features leads us to distinguish between two different types of storage, namely: First In-First Out (FIFO) storage and Last In-First Out (LIFO) storage.

3.2 Process Activation and Supply Chain Policies

Supply chain policies are also integrated in our model of the factory, together with activation policies that are able to turn a process on or off, depending on the demand for the output stock. To model these two concepts, we introduce the notion of *activable* and *activation*. An activable is something that can be enabled through an activation. We also associate a magnitude with the activation, or in other words, an integer. An activable can be a process or a supply order. In the case of a process, the activation represents the number of batches that the process is allowed to execute. In the case of an order, the magnitude represents the number of ordered items.

In our model, an order is a stationary, activable object that represents a class of order that can be passed. The order is passed when the modelled order object is activated.

Activables can be activated based on various rules that are also part of our modelling framework. There are three types of activation rules, namely: regular activations (performs the activation on a regular basis over a period of time);

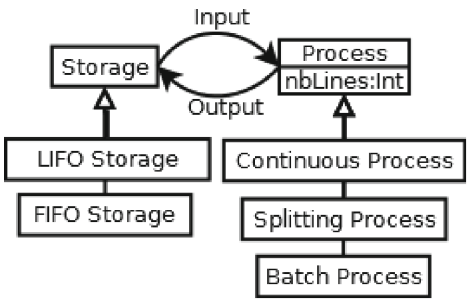

Fig. 3. Concepts of our process modelling languages.

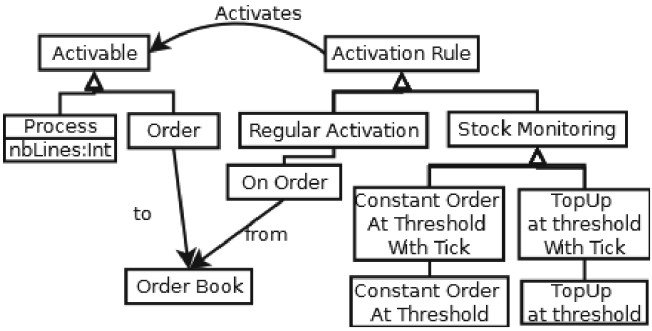

Fig. 4. Concept for modelling activation rules.

order-based activations (performs the activation when an order is received); and stock monitoring activation (performs the activation when the stock level gets below the threshold).

3.3 Modelling Intrinsic Item Features

To represent these intrinsic features, we introduce the notion of *item class*, representing the set of intrinsically identical items. Item classes are characterised by a set of boolean *attributes*. A global set of attributes is defined on the whole simulation model; each item has its own attributes, which is a subset of the global set of attributes. This set of attribute attached to the item defines the item class to which it belongs.

When an item flows through a process, the process can update the attribute of the item to reflect the process that was applied to this item. Similarly, when an item flows through a splitting process, the selected output can be specified according to the attributes of the item. At this point, we had to set a trade-off between the expressiveness of the modelling language, its simplicity, and the efficiency of the simulation. Processes can update the class of items through three basic operations: setting an attribute, clearing an attribute, or loading a constant set of attributes.

Another restriction that we have implemented regards how the class of items produced by a process is linked to the classes of several potential inputs of this process. We consider the union of all attributes with all inputs performed in order to start a batch of attributes in the process, and set this union as the start class for the whole batch. The class transformation function of the process is then applied to this class, and every item output in the process from this batch share the same output class computed by this class transform function.

Since items are distinguishable through their item classes, we introduced two models of storages, namely FIFO and LIFO storages, representing queues and stacks of items, respectively.

4 Performing Queries over Simulations

The goal of our approach is to perform risk-related queries on factory simulations. These queries are meant to be performed on single runs of simulations occurring inside the Monte-Carlo engine, which aggregates the query results over the runs. It can then be queried afterwards e.g. for mean, median, extremes, variance of these queries over the runs.

Our query language can roughly be split into three sets of operations, namely: probes on processes, probes on storages, and operators. Operators are split into five sets, namely logic operators, temporal logic operators, arithmetic operators, temporal arithmetic operators, and history recording operators. Arithmetic and logic operators differ by their their return types; they return numeric and boolean values respectively.

Since this query language runs over simulated time, we assume that the value of the queries are computed at the end of the trace on which they are evaluated. We define the operator of our language together with their semantics by using the $\models$ notation: $t \models P$ is the value of expression P when evaluated at position t of the current trace.

Some temporal operators refer to the previous position in time, denoted as $\text{prev}(t)$, notably to compute deltas or assess changes. These should be used with care since we are in an event-based model of time, so adding such operators in the query will add extra time events in the simulation.

4.1 Probes for Processes

The probes on processes are atomic operators that extract basic metrics from processes of the simulation model. Suppose that p is such a process, the following probes are supported:

- $t \models \text{running}(p)$ true if the process is running at time t, false otherwise.
- $t \models \text{completedBatchCount}(p)$ the total number of batches performed by the process between the beginning of the trace, and time t.
- $t \models \text{startedBatchCount}(p)$ the number of batches started by the process between the beginning of the trace, and time t. For a process with multiple lines, it sums up the started batches of each line.
- $t \models \text{totalWaitDuration}(p)$ the total duration where the process was not running between the start of the trace, and time t. for a process with multiple lines, it sums up the waiting time of each line.
- $t \models \text{anyBatchStarted}(p)$ true if a batch as started by the process at time t.

4.2 Probes for Storages

The probes on storages are atomic operators that extract basic metrics from storages of the simulation model. Suppose that s is such a storage:

- $t \models$ empty(s) true if the storage s is empty at time t, false otherwise.
- $t \models$ content(s) the number of items in the storage s at time t.
- $t \models$ capacity(s) the maximal capacity of s. This is invariant in time.
- $t \models$ relativeCapacity(s) the relative content of storage s at time t, that is: the content of the stock divided by the capacity of the storage.
- $t \models$ totalPut(s) the number of items that have been put into s between the beginning of the simulation and time t, not counting the initial ones.
- $t \models$ totalFetch(s) the number of items that have been fetched from s between the beginning of the simulation and time t.
- $t \models$ totalLostByOverflow(s) the number of items that have been lost by overflow from s between the beginning of the trace, and time t. If s is a blocking storage, this number will always be zero.

4.3 Operators

Logical Operators

- $t \models$ true the constant true.
- $t \models$ false the constant false.
- $t \models !l$ the negation operator.
- $t \models l_1$ op l_2 where *op* is one of $\{\&, \|\}$ represent conjunction, and disjunction operators, respectively, returning their conventional results.
- $t \models a_1$ comp a_2 where *comp* is one of $\{<, >, \leq, \geq, =, \neq\}$ represent comparison operators over numerical values, returning their standard results.

Temporal Logic Operators

- $t \models$ hasAlwaysBeen l true if for each t' in $[0; t]$, $t' \models l$
- $t \models$ hasBeen l true if there is a t' in $[0; t]$ such that $t' \models l$
- $t \models l_1$ since l_2 true if there is a position t' in $[0; t]$ such that $t' \models l_2$ and for each position t in $[t, t]$, $t \models l_1$
- $t \models @l$ true if both $t \models l$ and prev(t) $\models !l$.
- $t \models$ changed(e) e might be a logic or arithmetic expression; this evaluate to true when $t \models e$ and prev(t) $\models e$ have different values.

Arithmetic Operators

- $t \models n$ where n is a numerical literal represents a literal constant value
- $t \models a_1$ op a_2 where *op* is one of $\{+, -, *, /\}$ represent the classical arithmetic operators over numerical values, returning their conventional results.
- $t \models -a$ represents the unary negation.

Temporal Arithmetic Operators

- $t \models$ delta($a1$) is a shorthand for $t \models a$ prev(t) $\models a$
- $t \models$ cumulatedDuration(b) let be $T = (t_1, t_2) \| t_1 = prev(t_2) \& t_1 \models b \& t_2 \models b$ the accumulated duration of b is the sum over the couples (t_1, t_2) in T of t_2, t_1

- $t \models$ time evaluates to t.
- $t \models \min(a)$ the minimum over all the values of $t' \models a$ with t' in in $[0; t]$
- $t \models \max(a)$ the maximum over all the values of $t' \models a$ with t' in in $[0; t]$
- $t \models \text{avg}(a)$ the average of all the values of $t' \models a$ with t' in in $[0; t]$
- $t \models \text{integral}(a)$ the integral of $t' \models a\ dt'$ with t' in $[0; t]$. The integral is computed through the trapezoidal rule taking the events as discretisation base.

History Recording. Two operators are available for recording the evolution of a query throughout the simulation run. The type of these operators is a history of value. The value itself is Boolean, or arithmetic, respectively. These operators cannot be composed with any other operator. Semantically, they are only evaluated at the end of the run, so that there is no position in time in their definition.

- record(a) is the history of the value of the arithmetic query a over the simulation run.
- record(b) is the history of the value of the boolean query b over the simulation run.

5 Modelling Risks as Queries

Our query language allows for computation of any metric measurement given in the simulated factory model, and metrics associated to risks in particular. Thus, it is necessary to identify the risks we want to assess. The risk identification process is partly generic because there are well-known categories of risks. We considered here the classical delay/quality/quantity triangle, which can be defined as follows:

- **Delay risks** are related to the possibility of taking more time than expected to produce goods, for instance due to process malfunction, or problems in stocking materials due to supply and storage failures.
- **Quantity risks** are related to the possibility of producing less goods than expected in a process or losing materials in the storage stage.
- **Quality risks** are related to the possibility of a process producing "bad" goods, or the degradation of materials whilst in storage.

The characteristics of these risks, such as their likelihood and their impact, may vary a lot. We quantify these risks by expressing them in our query language, generally as a cost value. Risks can occur at different levels of granularity. *Elementary risks* occur at component level (such as a provider, a storage type, a process) but not necessarily result in a system level risk if the supply chain is designed to cope with the risk. *System-level risks* occur at system level, for example, the global production latency, or the critical path of a supply chain.

In the rest of this section, we present how typical risks can be expressed in our query language. Queries can reference other queries; this makes it possible to refer fine grained risks in the definition of higher level risks.

Risks Specific to Suppliers. For a single supplier, the risk model is directly encoded in terms of quality, quantity and delay, so it makes little sense to measure it. We can consider more complex configurations, such as a combination of suppliers, reactive suppliers with less quality or quantity, and a slower supplier with high quality or quantity. We can assess the adequacy of the design of a supplier combination by measuring the probability of underflow in the common input storage they are supposed to replenish, e.g. using the probe cumulatedDuration(content(sto) > MIN), where MIN is some minimal "safety" stock.

Risks Specific to Storages. As a delay risk for storage elements, we can compute the amount of time it takes before a storage overflow occurs ovst with the probe cumulatedDuration(hasAlwaysBeen(ovst) < 1). This allows us to know whether the storage has been losing pieces for a long time and be able to estimate the extent to which the storage should be enlarged.

The most basic quantity risk for an overflowing storage is the number of lost pieces. This is measured by the probe totalLostByOverflow(ovst). Together with the previous probe, useful adaptations to the size of the storage container can be decided.

With respect to the quality risks, it is possible to find that a storage is oversized, that is that the maximum contents of the stock throughout the simulation are a lot lower than the storage's capacity. This can be measured with the probe max(relativeCapacity(stock)) and it can be used to verify whether that value is higher than an acceptable percentage number.

Risks Specific to Processes. For quantity risks, we are interested in processes that do not work enough in the simulation. The percentage of idle time for a process p is measured by the probe (totalWaitDuration(p)/currentTime) $* 100$. To detect whether the process p operated at all in the simulation, we can use the probe: hasAlwaysBeen(!anyBatchStarted(p)).

About quality risks, a basic query is used to look at the ratio of failed part from a failing batch process: totalPut(p)/completedBatchCount(p). This is however of limited interest as it will return a value close to the encoded probability of failure. More general attributes are however being defined on processes with more complex transformation functions - this includes more randomness, which would make such a query informative at that level.

System Level Risks. Quantifying risks at system level is highly dependent of each considered case. Let us illustrate some typical queries that can be used. A general consideration is that at system level risk are systematically turned into costs so they can be compared. In order to do this specific attributes are available: for example the cost of all parts going through stock, the cost to repair a damage machine, penalties for supplier being late, etc.

For quality risks, considering losses occur from failing processes FP1 and overflowing storage ST1. Considering there is no rework, the query LOSS is:

totalFailed(FP1) * partCost(FP1)+totalLostByOverflow(ST1) * partCost(ST). Assuming there is also a similar query to define the VALUE_ADDED (based on the cost of incoming parts versus produced parts), the profitability is then defined as VALUE_ADDED/(LOSS + VALUE_ADDED). These queries could be made more general using a component selector to select all relevant parts to use during specified computation.

For quantity risks, an OUTPUT storage can be compared with an ORDER book: (content(OUTPUT) − content(ORDER)) * partCost(ORDER). We use the agreed order cost which may differ from the cost of the produced parts.

Global delay risks are more difficult to express because the DES engine does simulate the flow of parts but only reports transformation events. It is however possible to monitor the time required to fulfill some order and compare it with the agreed delivery time and take into account possible penalties of being late.

6 Implementation

In this section, we first give an overview of the global architecture of the software tool before giving more details about the execution of queries in the simulation, which is the main body of our work.

6.1 Global Simulator Architecture

The simulator is implemented using the OscaR DES module [4] and is written in Scala [19]. A modelling web front-end was developed with JavaScript technologies, mainly Bootstrap [20], JQuery [21] and JointJS [22]. The lightweight Scalatra [23] web framework was used to wrap up the simulator as a set of web services. The web server contains a Monte-Carlo simulation engine that is able to aggregate the results (especially the queries) of several simulations over the same model. Figure 5 illustrates the architecture.

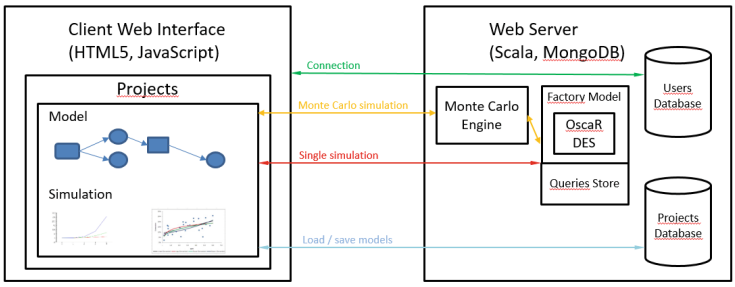

Fig. 5. Global architecture of the software tool.

6.2 Supporting Risk Identification

The software tool includes a wizard helping the user to express risk using our query language. A succession of sreenshot of this wizard is shown in Fig. 6. It starts from the main risk categories (quantity/quality/delay) then guides the user into specifying how the risk can be measured both at system level and then at component level. A number of predefined queries are available in each context. Queries can also be edited and designed from scratch using a plain text editor. The figure shows the encoding of a quantity risk measuring the number of lost pieces in a factory.

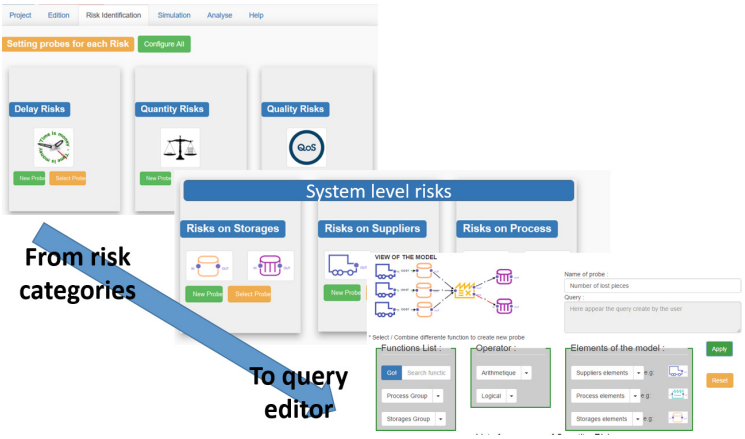

Fig. 6. Encoding a query for a quantity risk.

6.3 Efficient Evaluation of the Queries

All the elements of the factory feature optimal $O(1)$ complexity for their update operations. Attribute update operations performed by processes are collapsed into a constant number of bit-wise operation. Any combination of the operations on attributes can be aggregated into two efficient bit-wise operations performed using bit masks that represent attributes.

Queries must be evaluated efficiently during simulation runs. To this end, we have incorporated three mechanisms and the way they are evaluated on traces, namely:

- Incremental evaluation throughout the simulation run.
- Minimal updates that allows updating only the relevant fragment of queries.
- Bottom-up updates to allow the sharing of sub-queries.

Incremental Evaluation. Queries are evaluated incrementally throughout the simulation, so that no simulation trace is actually generated. At each step of the simulation, queries are notified about the new step, and they update their value accordingly.

A very simple expression illustrating this is `a+b`. At each step t, the value of this expression is the sum of the values of a and b: `sum.Value = a.value+b.value`.

A slightly more complex one is the $\max(a)$ expression. The `max` operator updates its output at each step t of the simulation run by applying the code: `if(a.value > max) max.value = a.value` where `max` is the output value of the operator, and `a.value` is the value of `a` at time t.

Some operators require more intricate update mechanisms with auxiliary internal variables. Consider the expression $\text{avg}(a)$; the `avg` operator maintains two variables: the number of steps so far, and the sum of all the values of `a` collected at these step so far. The output value of the expression is the ratio between these two internal variables: `avg.value = sum_of_a / number_of_time_Steps`.

All operators are able to perform such updates of their outputs.

Minimal Updates. To further increase the efficiency of the query evaluation, we distinguish two type of operators: non accumulating ones and accumulating ones. *Non accumulating operators*, such as the + operator do not actually need to perform updates at every step, they just need to be updated at the end of the simulation trace in order to deliver a correct final value. *Accumulating operators* include all the temporal operators, and need to perform some sort of update at each step of the simulation run in order to deliver a correct value at the end of the run.

Some non-accumulating operators might also need to be updated at each step, for instance if their output value is needed by an accumulating operator. We therefore introduce the notion of accumulating expressions and non accumulating expressions. An *accumulating expression* is rooted at an accumulating operator, or is the direct sub-expression of an accumulating expression. *Non-accumulating expressions* are all the remaining sub-expressions.

At each step of the simulation, all operators rooted at accumulating expressions are updated. The non-accumulating expressions are only updated at the end of the simulation run.

Bottom-up Update. A third mechanism that we have incorporated in our query evaluation engine is bottom-up update of the queries expressions. Queries are not structured into trees. They might actually be directed acyclic graphs because some fragment of queries can be shared by several queries. Bottom-up evaluation enables us to update shared fragment of the queries only once, thus gaining effficiency.

Overall Algorithm. When the simulation model and all the queries are instantiated into the engine, a one-shot analysis is performed on the structure

of the queries in order to label each sub-expression as accumulating or non-accumulating. All accumulating expressions are then put on an evaluation list, which also captures the order to follow when updating them at each step to comply with the bottom-up order. At each step of the simulation, accumulating expressions are updated following this evaluation list. At the end of the simulation run, non-accumulating expressions are updated once, following a similar bottom-up process.

7 Benchmarking

In order to assess the approach, we took a benchmarking approach based on a set of representative models that were run on the implementations described in the previous section. Our benchmarking addresses our two main contributions:

1. **Expressiveness:** shows that all the risks identified in the cases can easily be captured by the modelling primitives and measurement probes, either by being based on the set of generic probes identified, or by writing case specific probes.
2. **Performance:** shows that running probes does not degrade the performance of the simulation engine a significant amount.

7.1 Benchmark Models

We selected four representative models out of a set of about 20 examples inspired by classic academic cases (these have specific, complex aspects) and other anonymous cases collected in industry. The cases also vary in the level of use of random variables. We describe relevant modelling aspects of each supply chain, together with specific risk issues associated with each model.

First Case: A Simple Assembler Factory. This case, illustrated in Fig. 7, is a simple factory that builds industrial produce using two kinds of parts, Parts A and B. Each part has its supplier which feeds the stocks when they become lower than a given threshold. For part A, the supplier policy is to refill the stock to its maximum capacity. For part B, the policy is the delivery of a fixed amount of material. Part B must be preprocessed before assembly. The factory combines two units of part A with one unit of preprocessed part B, resulting in 80 % of products passing the quality tests. So the assembly process can be represented by a failing single batch process. The goal is to assess if the input stocks are kept within safe limits and are able to cope with production demand.

Second Case: A Beer Game Model. Our second case model is a classical problem called "beer game" [24]. It is a long linear supply chain going from the beer factory to the final retailer, passing by distributors and wholesalers. The beer factory is considered here as a supplier, and single batch processes are used to represent external sources of delay in transport. Intermediary stocks also add extra delays. The continuity of retailing, distribution and wholesaler processes is modelled by conveyor belts. The resulting model is shown in Fig. 1. The goal is to assess where potential bottleneck can occur.

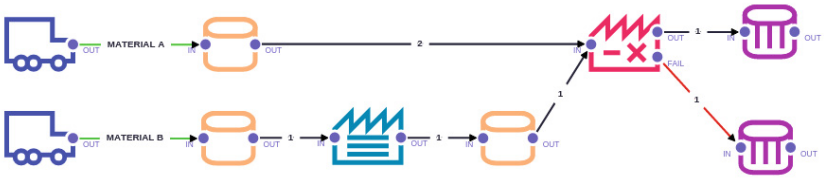

Fig. 7. Model 1: a simple assembler factory.

Third Case: Multiple Suppliers. This case is inspired by a real industrial case, where manufacturing involved three different materials having their own supplier and refill policy, with random delays belonging to a Gaussian probability distribution. 90 % of produce built by a batch in the factory fulfilled the quality requirements. We want to evaluate the effects of different supplying policies in order to ensure the supply chains operate at optimal capacity, whilst minimising the frequency of orders.

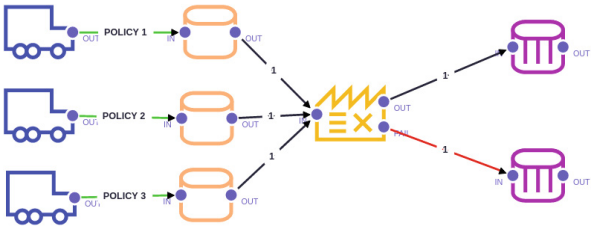

Fig. 8. Model 3: multiple suppliers.

Fourth Case: A Complex Assembly Process. Our last case is inspired by an industrial case in a factory where complex parts are assembled from 3 different materials following a complex process. Two of the parts are preprocessed on factory units that can fail (10 % of failures for the first one, 40 % for the second one). The process is shown in Fig. 9.

7.2 Expressiveness Analysis

We identified a number of basic probes relating to risks directly related to model elements. Such probes are automatically generated. So, for each stock, we generated three different probes for measuring the average and maximum contents of the stock and for verifying whether the stock is full or overflowing. For each process, we generated a probe for measuring the amount of time in which the process was idle or blocked in the simulation.

In addition, the user can specify extra probes for expressing business specific risks that are typically more complex queries in the model. Table 1 summarises some model characteristics like size (suppliers/processes/storages), risks

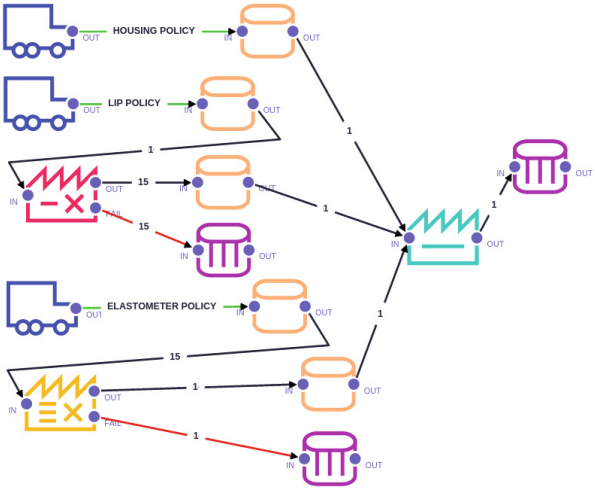

Fig. 9. Model 4: complex assembly process.

and number of probes. To assess expressiveness, we considered a single probe which actually proved enough to cover the targeted risks when dealing with basic probes. In the final two models we also explored risk mitigation strategies.

- In the first case, assessing whether the stocks of raw materials were full could be achieved with the probe $cumulatedDuration(relativeContent(stockA) = 1)$. For most of the time in the simulation, the stocks were full and the assembly process worked at full capacity.
- In the second case, we looked at the relative idle times in the process chain. We noticed that the distribution process, just after the fabrication, is the only one that blocks goods.
- In the third case, we both looked at process idle time (basic probe) and the average contents of stocks using the probe $avg(content(st))$. This helped us discover the best threshold to trigger an order, whilst minimising idle time.
- In the fourth case, a full stock was blocking the production. We mitigated the problem by experimenting with overflowing storage to estimate the right sized storage in order to avoid overflow by using the probe $totalLostByOverflow(lipStorage)$.

7.3 Performance Analysis

A Monte Carlo simulation was run for each case with a time limit of 10000 units and 2000 iterations to be more precise. Table 2 shows the computed average. We performed the benchmarks on an Intel Core i7-4600U CPU at 2.10 GHz with 8 GB of RAM. A single core is currently used. The simulation was triggered from the web interface on the same machine as the server.

Table 1. Benchmarking table for expressiveness.

Name	Size	Risk types	#probes	Comments
M1	9 (2/2/5)	Full stock Process failure	20	Simple manufacturing
M2	17 (1/7/9)	Blocked process Process failure	36	Beer game
M3	8 (3/1/5)	Full stock Process failure Supplier failure	20	Multiple suppliers
M4	14 (3/3/8)	Stock losses	31	Complex part assembly

Table 2. Benchmarking table for performance.

Name	No probes	Std probes	All probes	Overhead
M1	7,3 ms	11,9 ms	12,5 ms	71,2 %
M2	11,3 ms	17,6 ms	17,8 ms	57,5 %
M3	25,8 ms	29,2 ms	34,2 ms	32,6 %
M4	6,8 ms	13,3 ms	13,4 ms	97,1 %

The overhead in the simulation with probes varies from 32,6 % in Model 3 to 97,1 % in Model 4. Model 3 has the longest run time because of the randomness of the supply delays induced by probability distributions associated with suppliers. Model 4 has the shortest run time because the simulation stops at an earlier stage due to a full intermediary stock. In this case the relative overhead is bigger as a result of the DES engine - the load is more efficiently calculated due to the probe's evaluation in the modelling stage.

Globally, overheads are quite acceptable. Some improvements are still possible in the context of integration with a web application, especially in optimising the network requests between the web interface and the simulator, thus making that interface more responsive. The total simulation time allows thousands of simulations to run in a only a few minutes and explore risk mitigation alternatives within an hour.

8 Related Work

A typical risk assessment conducted on a given factory plan is reported in [25], which is based on the Arena simulation tool, featuring DES and Monte Carlo methods as is the case in our work. It stresses the importance of conducting stress-tests using such simulation platforms. Its focus is mainly on the disruption risk, unlike our work which can cope with other classes of risks, like quality for example. Our framework provides an added abstraction layer that can cut down the cost of performing these important stress tests, thus making them achievable by smaller industries.

A similar analysis has been performed on a beer supply chain in [24], whose model was presented in Sect. 7.1. This analysis leads to an evaluation of excessive accumulation in the inventory or back ordering sections. Again, no dedicated tooling was used for representing factories at a higher level, which would lead to higher costs for conducting such an evaluation in an industrial setting. Our tool could cope with using the available primitives.

Another simulation-based risk assessment is reported in [13]. It features an aerospace company with very low production volumes, and leads to the elaboration of a dedicated simulation engine. The engine was first developed with purely deterministic behavior, and then enriched with failure models and stochastic aspects. It was shown to be of great value to the company, despite mainly focusing on disruption risks, it helped the company develop a risk mitigation procedure. Our framework has a similar purpose and tries to propose a compromise between genericity and efficiency.

[26] presents a general framework that combines optimisation and the DES for supporting operational decisions in supply chain networks. Their idea is to iterate between a simulation phase, in which some parameters are estimated, and an optimisation phase that adapts the decision rules for the simulation. Our current work does not cover the minimisation of risk. The tool is rather designed to ease the identification of risk controls by the risk manager. We plan to address optimisation in a later phase, based on the optimisation engines also present in the OscaR framework [4].

9 Conclusions

We presented a Discrete Event Simulation Approach that is supported by a software tool for modelling the supply chain of manufacturing processes with the goal of assessing several kinds of risks on those processes, with a specific focus being placed on procurement risks. This assessment is performed through a simulation engine which uses Monte-Carlo techniques that can also be used to further explore risk mitigation strategies.

The strength of our approach is to support a declarative and easy to use graphical model for representing factory processes and stocks, together with a declarative query language for defining metrics to be measured whilst simulating the behaviour of the modelled system. We could successfully benchmark our approach both from the expressiveness and performance perspectives shown in several typical examples of factories, together with their supply policies.

Further work is still required in order to fully align our approach with the needs of industry. One of our current steps is to validate the tool by putting the tool in the hands of risk managers in a pilot case. We have already identified a number of requests regarding:

– extending the modelling language, e.g. to support the notion of shared resources among processes and have a statistic model of process failures and breakdowns. More specialised processes allowing controlled fork/joins are also required.

– identification of model parameters making easier the manual (and later optimised) exploration of risk mitigation strategies.
– availability of a companion library of specific risks and related probes.
– producing specific reports (e.g. business continuity plans). We have already explored some work in this direction [27].
– possibly creating model refinements and better granularity of the simulation. However our aim is not to capture the full reality but what will help in assessing identified risks.
– finally, parallelisation of the Monte-Carlo simulation engine to obtain better execution times, is needed.

Our framework combining usability, expressiveness and efficiency is an important milestone in our work regarding raising the company awareness, especially in smaller companies, of the need to evaluate their procurement risks and elaborate their supply policies in the most optimal manner. We believe it can be used to manage more general risks. Our design ideas can also be used to improve other risk management tools. Our framework is available online [28] and we plan to make it available open source.

Acknowledgement. This research was conducted under the SimQRi research project (ERA-NET CORNET, Grant No. 1318172). The CORNET promotion plan of the Research Community for Management Cybernetics e.V. (IfU) has been funded by the German Federation of Industrial Research Associations (AiF), based on an enactment of the German Bundestag.

References

1. Deleris, L., Erhun, F.: Risk management in supply networks using Monte-Carlo simulation. In: 2005 Winter Simulation Conference, Orlando, USA (2005)
2. Printz, S., von Cube, J.P., Ponsard, C.: Management of procurement risks on manufacturing processes - survey results (2015). http://simqri.com/uploads/media/Survey_Results.pdf
3. von Cube, J.P., Abbas, B., Schmitt, R., Jeschke, S.: A monetary approach of risk management in procurement. In: 7th International Conference on Production Research Americas' 2014, Lima, Peru, pp. 35–40 (2014)
4. OscaR: OscaR: Scala in OR (2012). https://bitbucket.org/oscarlib/oscar
5. Romeike, F.: Der prozess der risikosteuerung und kontrolle. In: Romeike, F. (ed.) Erfolgsfaktor Risiko-Management, pp. 236–243. Gabler, Wiesbaden (2004)
6. Zsidisin, G.A., Ritchie, B.: Supply Chain Risk: A Handbook of Assessment, Management, and Performance. Springer, New York (2009)
7. Siepermann, M.: Risikokostenrechnung: Erfolgreiche Informationsversorgung und Risikoprävention. Erich Schmidt, Berlin (2008)
8. Sutton, I.: Process Risk and Reliability Management, 2nd edn. Elsevier (2015)
9. Printz, S., von Cube, J.P., Vossen, R., Schmitt, R., Jeschke, S.: Ein kybernetisches modell beschaffungsinduzierter störgößen. In: Exploring Cybernetics - Kybernetik im interdisziplinren Diskurs. Springer Spektrum (2015)
10. Artikis, C., Artikis, P.: Probability Distributions in Risk Management Operations. Springer, London (2015)

11. Zio, E.: The Monte Carlo Simulation Method for System Reliability and Risk Analysis. Springer, London (2013)
12. Gleißner, W.: Quantitative methods for risk management in the real estate development industry. J. Prop. Investment Financ. **30**(6), 612–630 (2012)
13. Finke, G.R., Schmitt, A., Singh, M.: Modeling and simulating supply chain schedule risk. In: 2010 Winter Simulation Conference, Baltimore, USA (2010)
14. Brailsford, S., Churilov, L., Dangerfield, B.: Discrete-Event Simulation and Systems Dynamics for Management Decision Making. Wiley, Chichester (2014)
15. Byong-Kyu, C., Donghun, K.: Modeling and Simulation of Discrete-Event Systems. Wiley (2013)
16. AnyLogic: AnyLogic Multimethod Simulation Software (2015). http://www.anylogic.com
17. Automation, R.: Arena Simulation Software (2015). https://www.arenasimulation.com
18. Siemens: Plant Simulator (2015). http://goo.gl/gH63jw
19. Wampler, D., Payne, A.: Programming Scala. 2nd edn. O'Reilly media (2015)
20. Boostrap: Bootstrap website (2016). http://getbootstrap.com
21. The jQuery Foundation: jQuery website (2016). https://jquery.com
22. ClientIO: JointJS website (2016). http://jointjs.com
23. Scalatra: Scalatra website (2016). http://scalatra.org
24. Klimov, R.A., Merkuyev, Y.A.: Simulation-based risk measurement in supply chains. In: 20th European Conference on Modelling and Simulation (ECMS 2006), Bonn, Germany (2006)
25. Schmitt, A., Singh, M.: Quantifying supply chain disruption risk using Monte Carlo and discrete-event simulation. In: 2009 Winter Simulation Conference, Austin, USA (2009)
26. Almeder, C., Preusser, M., Hartl, R.F.: Simulation and optimization of supply chains: alternative or complementary approaches? In Günther, H.O., Meyr, H. (eds.) Supply Chain Planning, pp. 1–25. Springer, Heidelberg (2009)
27. Arenas, A.E., Massonet, P., Ponsard, C., Aziz, B.: Goal-oriented requirement engineering support for business continuity planning. In: Jeusfeld, M.A., Karlapalem, K. (eds.) ER 2015. LNCS, vol. 9382, pp. 259–269. Springer, Cham (2015). doi:10.1007/978-3-319-25747-1_26
28. SimQRi: Online SimQRi tool (2015). https://simqri.cetic.be