Analysis of the Runtime Resource Provisioning of BPMN Processes using Maude

Francisco Durán[1], Camilo Rocha[2], and Gwen Salaün[3]

[1] ITIS Software, University of Málaga, Málaga, Spain
[2] Pontificia Universidad Javeriana, Cali, Colombia
[3] Univ. Grenoble Alpes, CNRS, Grenoble INP, Inria, LIG, F-38000 Grenoble France

Abstract. Companies are continuously adjusting their resources to their needs following different strategies. However, the dynamic provisioning strategies are hard to compare. This paper proposes an automatic analysis technique to evaluate and compare the execution time and resource occupancy of a business process relative to a workload and a provisioning strategy. Such analysis is performed on models conforming to an extension of BPMN with quantitative information, including resource availability and constraints. Within this framework, the approach is fully mechanized using a formal and executable specification in the rewriting logic framework, which relies on existing techniques and tools for simulating probabilistic and real-time specifications.

1 Introduction

A crucial concern in most organizations is to have explicit and precise models of their business processes. These models may allow organizations to better understand, control, and manage critical activities and, possibly, make improvements to their processes. Indeed, process optimization is at the heart of business process management because of its potential to increase profit margins and reduce operational costs.

A business process is a collection of structured activities or tasks that produce a specific product and fulfil a specific organizational goal for a customer or market. A process aims at modeling activities, and their causal and temporal relationships by defining specific business rules. Process instances then have to comply with such a description once they are deployed. The Business Process Model and Notation (BPMN) [12] is a graphical modeling language for specifying business processes, which has become the common notation for designing business processes. Several industrial platforms have been developed during the last 10 years to support the modeling and management of BPMN processes. Nowadays, organizations are making efforts to use such platforms to define their organizational processes, aiming at achieving better control over the processes when they are deployed.

Once a process description has been obtained, a key question to ask—from the business perspective—is the following: can this process be improved to, e.g., save money? Process optimization is becoming a strategic activity in organizations because of its potential to increase profit margins and reduce operational costs. One of the main problems in process optimization is concerned with the task of streamlining resource provisioning, allocation, and sharing. A resource can be a machine, a robot, a tool, or an employee profile, and it may be associated with a cost. Given the strategic importance in saving costs where possible, a collection of resource patterns have been defined in the context of the workflow patterns initiative [1].

Providing automated techniques for analyzing and optimizing BPMN processes is a challenging problem. It requires a model of the process including execution time of tasks and flows, as well as an explicit description of resource usage requirements. A solution to this problem would take such a process as input and compute a set of metrics (e.g., process execution time, waiting times, resource occupancy) as output. These measures would then be used as part of a further analysis stage with the goal of optimizing the process relative to a cost model. All this effort is to have a proposal for, e.g., better allocation of resources and, thus, reducing the overall time and/or costs of the process when it is finally deployed. However, the assignment of resources is seldom static, rendering the optimization problem more interesting. Modern enterprises and systems have access to resource repositories and to the possibility of acquiring/releasing or hiring/firing them with great flexibility. Thus, they can provision and release resource instances as needed. Since the analysis procedure involves complex computations and lengthy simulations, it is highly convenient to be able to perform resource analysis in a fully automated way, especially at design time before the processes are deployed.

This paper presents a solution for the analysis of alternative strategies for the dynamic adaptation of resource assignments in process models. Instead of focusing on the allocation of a fixed set of available resources, alternative strategies are analyzed for the dynamic provisioning of such resources. Once the best strategy is chosen using one of the proposed methods, such adaptation strategy will allow to automatically adjust the number of required instances of resources at runtime, depending on the workload and the behavior of the process. The annotations on the BPMN processes will provide the necessary information on task durations, probabilistic choice, and information regarding resources (e.g., initial number of available resources, resources required per task, maximum number of resources). The approach relies on a formal specification in rewriting logic of BPMN processes. The specification is given in the rewriting-logic based language Maude [6, 5, 7] and serves as an executable semantics of the BPMN language under consideration. Since it is *executable*, it has the advantage of enabling the use of Maude's verification tools for computing a number of metrics of processes with a precise mathematical meaning.

The approach presented here is concerned with the analysis of quantitative properties associated to BPMN processes. Although it encompasses a broad selection of quantitative measures, the main focus in this paper is given to execution time (i.e., the time it takes to execute a process) and resource occupancy (i.e., the percentage of usage of any or all replicas of a resource). The final goal is thus to use such analyses to streamline a process by reducing its operational costs in relation to execution time and resources, which can be directly inferred from the estimated execution times and resource usage. Since these measures are computed by significant simulations, and also along the actual executions, they can be used to dynamically adjust the number of resources at runtime. The given formalization and the accompanying tools will enable the comparison of different strategies for resource provisioning in a dynamic environment. More precisely, given a process description, and taking as parameters the workload and the provisioning strategy, the techniques and tools presented here provide detailed information on the evolution of execution times, resources in use, and therefore costs, which altogether will help in deciding on the best fit for the specific needs.

The application of the approach is presented and discussed on a case study with dynamic allocation of resources. It is used to show how the proposed approach can be helpful to effectively reduce the cost and execution time of a process. The current Maude specification

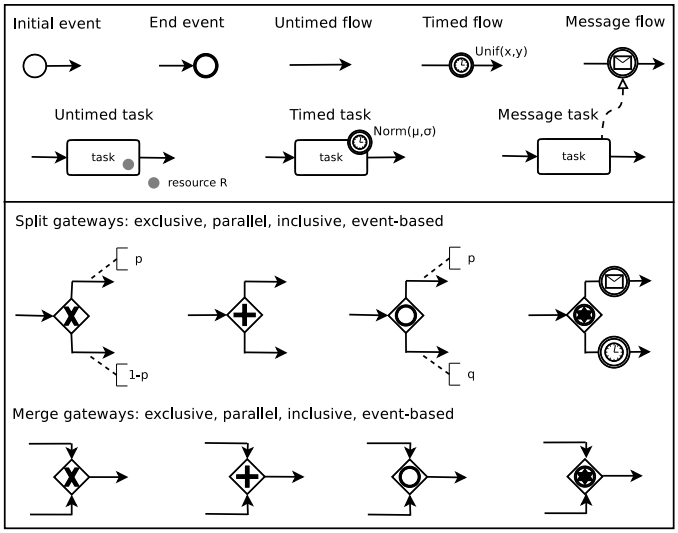

Fig. 1. Supported BPMN syntax.

builds on the one developed by the same authors in previous related work [8, 9] for different forms of analysis of business processes. The reader is referred to `http://maude.lcc.uma.es/BPMN-RA` for details on the formal specification, experiments, and additional examples.

The organization of the rest of the paper is as follows. Section 2 presents the BPMN notation extended with the annotations supporting the proposed approach. Section 3 introduces Real-Time Maude. Section 4 overviews the specification of the annotated BPMN extension in Maude's rewriting logic, which serves as a semantics for the language and makes automated analysis possible using Maude's tools. Section 5 presents the novel analysis techniques and case studies illustrating how the number of resources evolve, and how costs and time can be reduced in practice without the need for human intervention. Section 6 presents a discussion on related work. Finally, Section 7 concludes the paper.

2 Annotated BPMN

Familiarity with the BPMN notation is assumed. In this section the focus is on its extension to support quantitative information. In essence, times are expressed as stochastic expressions and branching alternatives as probabilities associated to branches. These parameters are supposed to be provided by the experts that specify the business process, or are learnt from available execution logs using, for instance, recent contributions on process mining and discovery such as [21, 14]. Figure 1 summarizes the BPMN constructs supported in this work. These elements are used to develop activity and collaboration diagrams of process models. In addition to the description of specific tasks and their sequencing, collaboration diagrams also involve *pools* and *lanes*, which are structuring elements that split processes into pieces.

To introduce and illustrate the use of the BPMN constructs and annotations supported, and the analysis techniques presented in this paper, a process describing a parcel ordering

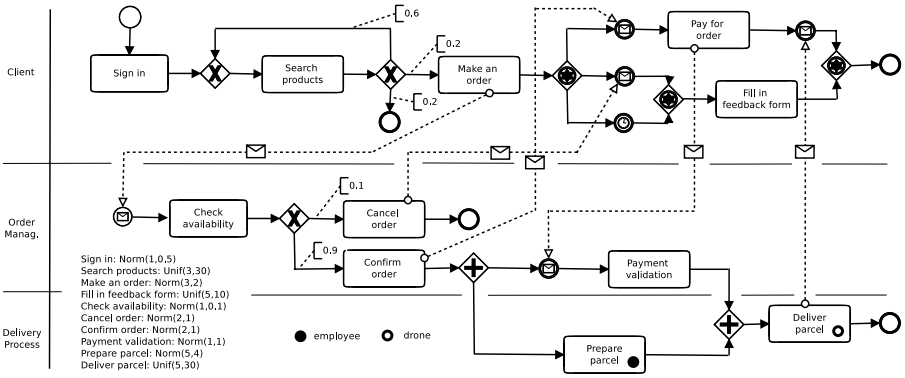

Fig. 2. Running example: parcel delivery by drones.

and delivery by drones will be used. Figure 2 presents a collaboration diagram modeling such a process. It consists of three lanes, namely, one for the client, one for the order management, and one for the delivery process. In this process, the client first signs in and then repeatedly looks for products. Eventually, the client can decide to give up (i.e., termination) or to make an order by submitting it to the order management lane. The client then waits for a response (i.e., acceptance or refusal of this order). If the order can be completed, then the parcel is received and the client pays for it. Otherwise (i.e., timeout or order refused), the client fills in a feedback form. As far as the management lane is concerned, the first task aims at verifying whether the goods ordered by the client are available. If they are not available, then the order is canceled; otherwise, the order is confirmed. The order management takes care of the payment of the order whereas the delivery lane is triggered to prepare the parcel to be delivered by a drone. This process exhibits different kinds of gateways, probabilities for choice gateways, stochastic functions for time associated to tasks, and a loop (Search products task).

The timing information associated to tasks and flows (durations or delays) is described either as a literal value (a non-negative real number) or sampled from a probability distribution function according to some meaningful parameters. The probability distribution functions currently available include exponential, normal/Gauss, and uniform (see, e.g., [22]). To simplify the reading of the process in Figure 2, the delays in all flows are set to 0 and the specification of the task duration has been placed apart from the process description, at the bottom-left corner. In the modelling tool, these parameters would be specified as properties of the corresponding elements. For instance, the duration of the Sign in task follows a normal distribution with mean 1 and variance 0.5, and the Search products task follows a uniform distribution in the interval [3,30].

Four types of gateways are considered: *exclusive*, *inclusive*, *parallel*, and *event-based*. Both BPMN 1.0 and 2.0 semantics for inclusive gateways are supported in this work. Data-based conditions for split gateways are modeled using probabilities associated to outgoing flows of exclusive and inclusive split gateways. For instance, notice the exclusive split after the Search products task in the Client lane of the running example, which has outgoing branches with probabilities 0.6, 0.2, and 0.2, specifying the likelihood of following each corresponding path. The probabilities of the outgoing flows in an exclusive split must

sum up to 1, while each outgoing flow in an inclusive split can be equipped with a probability between 0 and 1 without a restriction on their total sum.

Each lane in a collaboration diagram corresponds to a specific role or resource. However, instead of implicitly associating resources to lanes, resources are explicitly defined at the task level. Thus, a task that requires resources can include, as part of its specification, the number of required instances (or replicas) of a resource. The process in Figure 2 relies on employees for parcel packing and drones for parcel delivery. The small circles at the bottom-right corner of the Prepare parcel and Deliver parcel tasks indicate that one instance of the employee resource and another one of the drone resource are required, resp., for the tasks completion. Several tasks could compete for the same resources. Furthermore, since *multiple* instances of a same process may be executed concurrently, instances also access and may compete for the shared resources. Notice that resources can refer to humans (e.g., employee, cashier, executive) as well as non-human ones (e.g., drone, virtual machine, paper, money), and we can specify the number of instances or replicas as a natural number. In case of unlimited resources, the number of units of time, length, or volume can also be considered.

As presented in Section 4, provisioning strategies are specified in Maude. Although in future work alternative mechanisms to specify them can be developed, in the present work the interest is in their comparative analysis, and therefore a catalog of strategies is presented from which the desired one can be chosen. Independently of the criteria used for the provisioning/releasing of resources, it is assumed that the amount of resources is accounted for periodically (time between checks or TBC), and that in that check the recent history of the process is considered, being the length of the considered history another parameter (history length or HL) of the optimization process. The provisioning and releasing of resource instances is supposed to happen in accordance to some given thresholds, which will also be provided as parameters. Finally, it is assumed that there is a maximum and minimum number of instances available for each resource, which will be given by a range (*min*,*max*). Unlimited availability of a resource can then be modeled by just assigning a negative *max* value.

3 Real-Time and Probabilistic Rewrite Theories

This section provides an overview of real-time and probabilistic features of rewriting logic [15] and Maude [6]. The executable specification of BPMN presented in the following sections is a probabilistic rewrite theory [3] $\mathcal{R}=(\Sigma,E\uplus B,R)$, where $(\Sigma,E\uplus B)$ is a membership equational logic [4] theory with Σ its signature, E a set of conditional equations, B a set of equational axioms so that equational rewriting is performed modulo B, and R is a set of labeled conditional rules. The equational subtheory offers the infrastructure for defining a process in the sublanguage of BPMN described in Section 2, including the timing behavior for tasks and flows, resource dynamics, and probabilities for outgoing flows of split gateways. The real-time aspects are modeled using Real-Time Maude [18], which supports the formal specification and analysis of *real-time systems*. Specifically, the probabilistic rewrite rules R axiomatize how time advances and probabilistic choices are made in this infrastructure, in order for a given process to transition from an initial to a final state.

Real-Time Maude provides a sort Time to model the time domain, which can be either discrete or dense. Time advancement is modeled with *tick rules*, e.g.,

$$\texttt{crl [}l\texttt{] : \{ }t\texttt{, }T\texttt{ \} => \{ }t'\texttt{, }T+\tau\texttt{ \} if }C\texttt{ .}$$

where t and t' are system states (an evolving model in our case), T is the global time and τ is a term of sort Time that denotes the *duration* of the rewrite, affecting the *global time elapse*. Since tick rules affect the global time, in Real-Time Maude time elapse is usually modeled by one single tick rule and the system dynamic behavior by instantaneous transitions. Although there can be many sampling strategies, in this work time elapse is modeled with a single tick rule with the help of two functions: the delta function, that defines the effect of time elapse over every model element, and the mte (maximal time elapse) function, that defines the maximum amount of time that can elapse before any action is performed (see [18] for additional details).

In a standard rewrite theory, the conditions of rewrite rules are assumed to be purely equational. A rewrite rule $l(\overrightarrow{x}) \rightarrow r(\overrightarrow{x})$ if $\phi(\overrightarrow{x})$ specifies a pattern $l(\overrightarrow{x})$ that can match some fragment of the system's state t if there is a substitution θ for the variables $\overrightarrow{x}$ that makes $\theta(l(\overrightarrow{x}))$ equal modulo B to that state fragment, changing it to the term $\theta(r(\overrightarrow{x}))$ in a local transition if the condition $\theta(\phi(\overrightarrow{x}))$ is true. In a probabilistic rewrite theory, rewrite rules can have the more general form $l(\overrightarrow{x}) \rightarrow r(\overrightarrow{x}, \overrightarrow{y})$ if $\phi(\overrightarrow{x})$ with probability $\overrightarrow{y} := \phi(\overrightarrow{x})$, where some new variables $\overrightarrow{y}$ are present in the pattern r on the right-hand side. Because the pattern $r(\overrightarrow{x}, \overrightarrow{y})$ may have new variables $\overrightarrow{y}$, the next state specified by such a rule is not uniquely determined: it depends on the choice of an additional substitution ρ for the variables $\overrightarrow{y}$. In this case, the choice of ρ is made according to the family of probability functions π_θ: one for each matching substitution θ of the variables $\overrightarrow{x}$. Therefore, a probabilistic rewrite theory can express both non-deterministic and probabilistic behavior of a concurrent system.

4 Executable Specification of BPMN

This section presents the Maude representation of the timed and probabilistic extensions of BPMN introduced in Section 2. The algebraic semantics of BPMN is provided by a MEL theory $Spec_{\text{BPMN}}$ so that a process model P is an element of the initial algebra $\mathcal{T}_{Spec_{\text{BPMN}}}$. The rewrite theory RT_{BPMN} extends $Spec_{\text{BPMN}}$ and defines the behavior of BPMN processes by providing some additional definitions and rules specifying such a behavior. The Maude specification of BPMN therefore consists of two parts: the process structure as an equational specification and its evolution semantics using rewrite rules.

Process description. In the Maude specification of BPMN, a process is represented as an object with sets of flows and nodes as attributes. The representation of each node type includes the necessary information to describe its structure and to contribute to the overall process analysis. For instance, a task node involves an identifier, a description, two flow identifiers (input and output), a stochastic function modeling its duration (0 if there is no duration), a set of resources required for its execution, and a set of messages to be delivered after its completion. A split node includes a node identifier, a gateway type (exclusive, parallel, inclusive, or event-based), an input flow identifier, and a set of output flow identifiers. A merge node includes a node identifier, a gateway type, a set of input flow identifiers, and an output flow identifier. The representation of any flow includes a probability distribution function specifying its delay, a message produced by a task that blocks the flow until the message is received, and a timer representing a delay after which the execution can be triggered.

Figure 3 gives an excerpt of the representation for the running example. It shows how a Process object has attributes with the definition of its nodes and flows connecting them.

```
01 < pid : Process |
02     nodes : (start(initial, cf1),
03              merge(g1, exclusive, (cf2, cf5), cf3),
04              split(g2, exclusive, cf4, ((cf5, 0.6) (cf6, 0.2) (cf7, 0.2))),
05              split(g3, eventbased, cf8, (cf9, cf10, cf11)),
06              task(t10, "Prepare parcel", mf7, df1, Norm(5.0, 4.0), employee, empty),
07              task(t11, "Deliver parcel", df1, df2, Unif(5.0, 30.0), drone, parceldelivered),
08              ...),
09     flows : (flow(cf1,  0),
10              flow(cf9,  0, message(orderconfirmed, "Order confirmed")),
11              flow(cf10, 0, message(ordercanceled, "Order canceled")),
12              flow(cf11, 0, timer(timeout, 60)),
13              ...) >
```

Fig. 3. Running example: representation in Maude of the parcel delivery process.

For example, the exclusive split g2 has as incoming flow cf4 and outgoing flows cf5, cf6, and cf7, with associated probabilities 0.6, 0.2, and 0.2, respectively. As another example, the event-based split gate g3 has as incoming flow cf8 and outgoing flows cf9, cf10, and cf11. These flows are defined in the set of flows.

The transformation from the BPMN diagrammatic representation of processes into the corresponding Maude representation is carried out using the VBPMN platform [13].

Execution semantics. The operational semantics of BPMN is defined using rewrite rules, modeling how *tokens* (see below) evolve through a process, thus defining the execution semantics of BPMN. Each observable action is modeled as a rewrite rule. E.g., when a token arrives at an event-based split gateway, the token is made active with its optional timer. In that rule, if there is an outgoing flow with a timer, an event is added with the corresponding time to the set of available events. Another rule specifies the case where there is an outgoing flow with a message in the set of events. For instance, in that case, that branch is activated and one token is added for that flow. Additional objects of classes Workload and Supervisor are in charge of, respectively, modelling the workload of the process, and provisioning resources depending on the whereabouts of the process execution. In general, rewrite rules operate on systems composed of a Process object, a Simulation object, a Workload object, and a Supervisor object.

Simulation. While the process object represents the BPMN process and does not change during executions, a simulation object keeps information on an execution of the process. It stores a collection of tokens (in a scheduler, see below), a global time (gtime), a set of events (messages and timers), and a set of resources. It also keeps track of the metrics being computed. Figure 4 presents the attributes of the Simulation class. We can get an intuition of how these values get updated in the rule in Figure 5.

Tokens. Tokens are used to represent the evolution of the workflow under execution. A token is represented as a term token(TId, Id, T). Since several executions of the process are simultaneously happening, each execution has a unique identifier. Tokens are identified by the execution instance TId they belong to, and the flow or node Id they are attached to. The expression T represents a timer, of sort Time, modeling a delay on the token. Once this timer becomes 0, the token can be consumed.

Scheduling. Tokens are stored in a *scheduler*—see the attribute tokens of the Simulation object in Figure 4—implemented as a priority queue, so that tokens are stored according

```
01 class Simulation |
02   tokens : List{Token},              ---- scheduler
03   gtime : Time,                      ---- global time
04   resources : Set{Resource},         ---- resources in the system
05   events : Map{Id,Set{Event}},       ---- events in each execution
06   process-execs : Map{Id,Time},      ---- execution time of each execution
07   sync-times : Map{Id,Map{Id,Time}}, ---- synchronization time of each gate in each execution
08   task-times : Map{Id,Map{Id,Time}}, ---- task execution times
09   ...
```

Fig. 4. Declaration of the Simulation class (partial, please, note the ellipsis).

to their due time. However, even with its timer set to 0, the token at the front may be not enough to fire some action. Consider for example a task that requires some resource that is not available or a parallel merge for which some incoming flow is not yet active. To avoid blocking situations, the scheduler is provided with a *shifting* mechanism, which moves the first active token to the front of the scheduler in case the current head cannot fire the corresponding action. This scheduler is similar to those used in typical discrete event simulations.

Events.[4] A message event may be associated to a flow, which is blocked until the message is received. A timer event may be associated to a flow. When a token arrives at a timer event, its countdown is started: once the countdown is completed, the token moves to the outgoing flow. Both message and timer events are usually associated to event-based gateways, but it is not necessarily the case (see, e.g., the initial flow for the order management lane in the process in Figure 2). Asynchronous events are modeled using an event set in the Simulation object. When a message is dispatched, a corresponding event is added to the set. Flows and gateways that are waiting for specific messages use this set to check whether the messages have arrived.

Dynamic resources. Each resource is described with an identifier, the number of available replicas (initially the total number), the total amount of time this resource has been in use, and the intervals of time on which it was used. These two last parameters are required for analysis purposes only. When a task requires several resources, it atomically uses them or waits for them to be available.

Tasks. A task execution is modeled with two rules. The first rule, the initTask rule shown in Figure 5, represents the task initiation, which is applied when a token with zero time is available at the incoming flow (Line 05). If all the resources required by this task are available, which is checked with the allResourcesAvailable function (Line 08), then a new token is generated with the task identifier and the task duration (Line 12). Otherwise, the scheduler's token shifting mechanism is invoked (Line 19—note the ellipsis). If available, all required resources are removed from the resource set and the time those resources have been in use is updated (grabResources&updateTime function, Line 17). Note also that rules update the information on execution times, task durations, etc. (see, e.g., the update of the task-tstamps attribute, Lines 13–15).

Merge gateways. When a merge gateway is triggered, the incoming tokens are removed, a new token is added to the scheduler for the outgoing flow, and simulation information is

[4] Only inter-lane events are considered; to consider environment events, the environment may be added to the simulation model.

```
01 rl [initTask] :
02    < PId : Process |
03       nodes : (task(NId, TaskName, FId1, FId2, SE, RIds, SEI), Nodes), Atts >
04    < SId : Simulation |
05       tokens : (token(TId, FId1, 0) Tks),
06       task-tstamps : TTSs, gtime : T, resources : Rs, Atts1 >
07    < CId : Counter | counter : N >
08 => if allResourcesAvailable(RIds, Rs)
09    then < PId : Process |
10            nodes : (task(NId, TaskName, FId1, FId2, SE, RIds, SEI), Nodes), Atts >
11         < SId : Simulation |
12            tokens : insert(Tks, token(TId, NId, time(eval(SE, N)))),
13            task-tstamps : if TTSs[TId][NId] == undefined
14                           then insert(TId, insert(NId, T, TTSs[TId]), TTSs)
15                           else TTSs fi,  ---- for loops, stamps get overwritten
16            gtime : T,
17            resources : grabResources&updateTime(RIds, Rs, time(eval(SE, N)), T), Atts1 >
18         < CId : Counter | counter : int(eval(SE, N)) >
19    else ... fi .                          ---- if necessary, the scheduler is updated
```

Fig. 5. Task initiation rule.

```
01 rl [supervisor] :
02    < SId : Simulation | resources : Rs, gtime : T, Atts1 >
03    < Sup : SupervisorUsage | TBC : TBC, time-to-next-check : 0,
04                              CI : CI, thresholds : Thds, Atts2 >
05 => < SId : Simulation | gtime : T,
06       resources : update(Rs, Thds, CI, T), Atts1 >
07    < Sup : SupervisorUsage | TBC : TBC, time-to-next-check : TBC,
08                              CI : CI, thresholds : Thds, Atts2 >
```

Fig. 6. Usage-based strategy supervisor rule.

updated with synchronization times. For inclusive gateways, the semantics of BPMN 1.0 and 2.0 are both supported in this research.

Supervisor. A Simulation object collects all data relevant for the analysis, which is then used by a supervisor object to decide on the number of resource instances. Intuitively, the supervisor object is in charge of collecting the data on the chosen metric for the specified window of time (history length) and then decides in accordance. It takes into account ranges and thresholds for each resource to change, every TBC time units, the total amount of resources available to the process. Figure 6 represents the resource check action. Every TBC time units, the supervisor object updates the number of resource instances (Line 06) according to the state of the resources (Rs), the thresholds (Thds), the interval to consider (CI), and the current global time (T).

Workloads. Simulation-based analysis techniques are typically parameterized by the workload. They define the rate at which new instances of a given process are executed. The rule in Figure 7 specifies the behavior of closed workloads. Given a number of works, or times the process is to be executed (attribute works), and a stochastic expression SE describing the inter-arrival time (kept in the rate attribute), the rule generates a new work after the specified amount of time until all works have been created. Notice that the timer attribute of the Workload object is initialized with the result of evaluating the stochastic expression (Line 05). The rule is applicable when the timer becomes 0 and then a new token in the initial node is inserted in the scheduler (Line 07). The evaluation of stochastic expressions is carried out by the eval operation. Random numbers are generated using a pseudo-random number algorithm, which

```
01 rl [Workload] :
02    < WId : Workload | timer : 0, rate : SE, works : s W >
03    < SId : Simulation | tokens : Tks, events : ME, Atts1 >
04    < CId : Counter | counter : N >
05 => < WId : Workload | timer : time(eval(SE, N)), rate : SE, works : W >
06    < SId : Simulation |
07        tokens : insert(Tks, token(token(s W), initial, 0)),
08        events : (token(s W) |-> empty, ME),
09        Atts1 >
10    < CId : Counter | counter : int(eval(SE, N)) >
```

Fig. 7. Workload rule.

takes a number that indicate the position in the sequence (the Counter object is in charge of appropriately increasing these numbers).

5 Dynamic Resource Allocation

This section presents automated techniques for analyzing dynamic adjustment of resource allocation. The provisioning of resources may be carried out using different criteria. These adaptation strategies present trade-offs between the difficulty of use—-mainly due to the amount of parameters that need to be specified or the difficulty to estimate them—-and the benefits of an adaptive provisioning in terms of resource costs and response time. In this section, two alternative strategies are presented: one based on the observed resource usage (*usage-based* strategy) and another one based on the demand on the resources (*queue-based* strategy). Although only these two strategies are used here, other metrics could have been used instead. That is, the resource adaptation strategy presented is a parameter of the generic analysis techniques proposed here. It is fair to say that the selected strategies cover two alternative and complementary approaches: while the usage-based one is based on the observation of the behavior of the system, the queue-based one relies on the prediction of the resource demand. As behavioral observations, other typical metrics could have been considered, including the observed response time or its average or variance. As predictive indicators, synchronization times, bottlenecks, or other observations on the internals of processes could also be used.

Whatever the metric used to adapt the processes is, it is assumed that each of them is driven by a recommended range of values: If the observed value goes over some maximum threshold, then the number of instances of a resource is increased; if it goes below some minimum value, then the number is decreased. In trying to avoid under- or over-provisioning, it is assumed that the minimum and maximum number of instances are also bounded. E.g., due to office space limitations, a process cannot have more than ten employees, independently of its cost or productivity. It is also assumed that the strategy proceeds by checking on some given metrics periodically, and considering the latest values of such metrics in order to make a decision on the provisioning or releasing of resources.

Table 1 summarizes the parameters for the analysis of the process and a possible instantiation of them. TBC and HL stand for the Time Between Checks and the History Length or window of values considered in the check, respectively. Every TBC time units, the state of the system is evaluated and the amount of resource instances correspondingly updated. The evaluation takes into account the given metric for HL time units. Although these strategies consider

	TBC	HL	range		initial number of instances	thresholds	
			min	max		min	max
employee	5	10	1	3	1	50	70
drone			1	6	1	50	75

Table 1. Sample set of parameters for the delivery process.

the average value for the samples in the window, other strategies could also check that all values in the window were over/under a given threshold or any other check considered useful.

Notice that each resource has its own range and threshold. Table 1 specifies a possible selection of values with which the simulations of the delivery running example may be executed, as well as the threshold values for the usage-driven strategy. Specifically, the average usage of each resource replica is expected to be in the range [50%, 70%] for employees and [50%,75%] for drones.

Given a process description, a specification of resources (i.e., specific values for the above parameters), and a workload, the experiments discussed in what follows illustrate how information on execution times and resource usage is collected. This information can then be used to find the best strategy or best fit of its parameters. All simulations were performed assuming a closed workload with 1000 instances and an exponentially distributed inter-arrival time ($\lambda = 0.5$).

Figure 8 shows the evolution of the total amount of resources provisioned along the execution of the delivery process using the usage-guided strategy. The evolution of the number of instances is shown on the left for employees and on the right for drones. In this case, the parameters are: unlimited availability, thresholds (50,70) for employee and (50,75) for drone, TBC 1, and HL 0. Notice, first, that the variability is very high; since the TBC is set to 1, the amount of resources is almost continuously re-evaluated. A HL value of 0 increases this continuous adaptation, since decisions are taken by considering the values at the given time, even if that value is not maintained for some time. Also, note that employees, an expensive resource, move between 1 and 14, although most of the time it ranges between 2 and 6-7. For drones, although the most frequent values are in the range 10-20, it moves between 1 and 38. However, there are other values to take into consideration before changing the process parameters. The average execution time for the process is 55.01, with variance 0.61. The usage percentage was rather low, 42.86% for employees and 34.34% for drones. This results in a total cost, assuming the cost per hour for employee is 50€ and 20€ per drone, of 991,613.3€. A comparative study of these values will be presented later to better understand what these numbers mean for the example (see Table 2). But even with these raw data, a poor use of our resources can be observed, which means a higher cost than possibly required.

Figure 9 shows the evolution of the amounts of resources for a new set of parameters. Specifically, two parameters have been changed: the number of instances is now restricted, so that employee instances are now in the range [1,3] and drone instances in the range [1,6]; the number of instances is now re-evaluated every 10 time units. With these parameters, the execution time has slightly improved (average 57.22 and variance 0.72), and also the usage percentage (52.12% for employees and 67.05% for drones). A bigger TBC is allowing the system to stabilize before attempting a new adaptation. This leads to a significant reduction in the total cost to 454,713.8€ (assuming the same costs per hour for employee and drone as above).

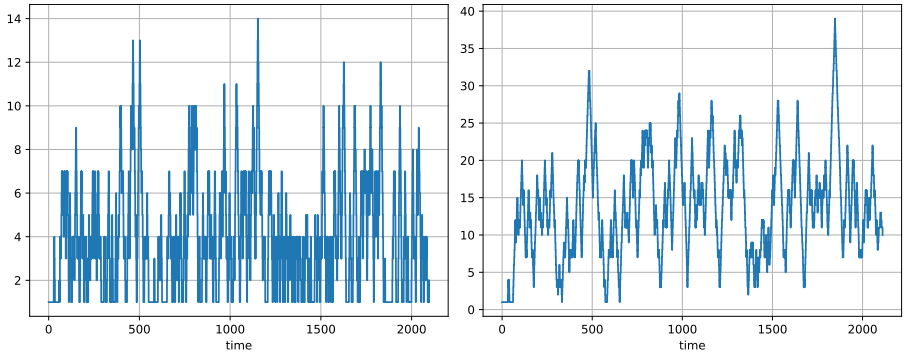

Fig. 8. Number of employee (left) and drone (right) instances along execution with a resource-usage-based strategy (unlimited availability, thresholds: employee (50,70), drone (50,75)). TBC: 1, HL: 0.

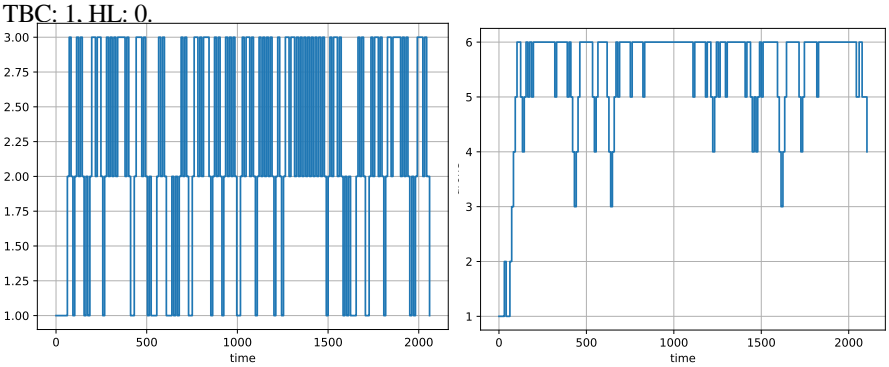

Fig. 9. Number of employee (left) and drone (right) instances along execution with a resource-usage-based strategy (ranges: employee [1,3], drone [1,6], thresholds: employee (50,70), drone (50,75)). TBC: 10, HL: 0.

As it is shown in the comparative study below, these results are quite good, although they are obtained at the expense of a great variability, as Figures 8 and 9 show. This variability could have been reduced by deciding on the provisioning or release of the resource instances with a larger TBC or a larger window of values, instead of just the latest one. Furthermore, there is also the more realistic alternative of deciding on the current demand of resources and not on the history of results, whatever the size of the window one may want to consider.

Figure 10 depicts the evolution of the total number of instances for employees and drones using the queue-based strategy with the following parameters: unlimited availability, thresholds [3,6] for employee and [2,8] for drone, TBC 10, and HL 5. It can be observed in the figure that the total number of instances remains much more stable: the number of employees stays between 1 and 3, and the number of drones varies between 1 and 8, going up and down repeatedly depending on the demand. The information observed in these graphs is complemented with the execution times (average 69.09, variance 1.49), and usage percentages (71.37% for employees, 86.38% for drones). Assuming the same costs as above, this results in a total cost of 370,742.7€.

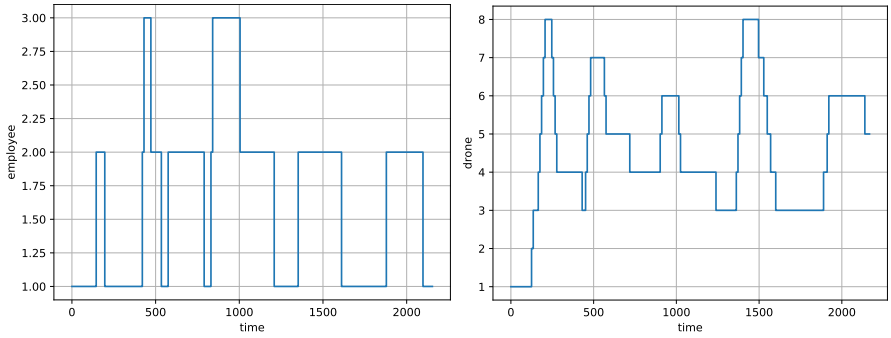

Fig. 10. Number of employee (left) and drone (right) instances along execution with a resource-queue-based strategy (unlimited availability, thresholds: employee (3,6), drone (2,8)). TBC: 10, HL: 5.

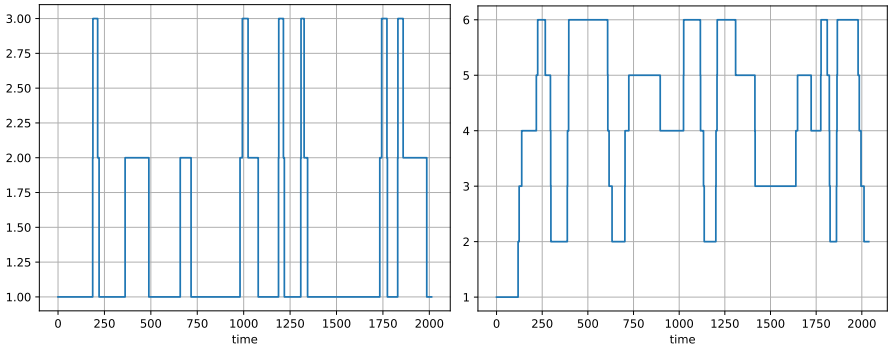

Fig. 11. Number of employee (left) and drone (right) instances along execution with a resource-queue-based strategy (ranges: employee [1,3], drone [1,6], thresholds: employee (3,6), drone (2,8)). TBC: 1, HL: 5.

Although many other values could have been considered for these parameters, the set of graphs we present here is completed by showing what happens if, again with the queue-based strategy, the number of instances of employees is restricted to [1,3] for employees and to [1,6] for drones, and re-evaluation takes place more often (TBC: 1). The evolution of the resources is depicted in Figure 11. The execution times improved (average 70.04, variance 1.57) as well as the resource usage (82.81% for employees, 93.83% for drones). The total cost with these parameters is 305,961.9€.

Table 2 shows the execution times, usage percentages and total costs for several simulations using different parameters. Specifically, a combination of TBCs of 1 and 10, HLs of 0 and 5, restricted and unrestricted resource amounts, and different threshold values are considered. As the previous discussion shows, the selection of the right parameters is indeed a multi-objective problem, where the goal is to minimize execution times and total costs. However, it is not only that, restrictions such as the tolerable variability and the maximum amount of resource instances available need to also be taken into account. It can be observed that the minimum cost in the table is obtained for Row 8, which is the result of restricted availability and stability. Unrestricted resource availability results in higher

	strat.	TBC	HL	range	threshold	exec. time		usage (%)		total cost
				empl.-drone	empl.-drone	avg	var	empl.	drone	(in €)
1	usage	1	0	unrestricted	(50,70)-(50,75)	55.01	0.61	42.86	34.34	991,613.3
2		10	0	unrestricted	(50,70)-(50,75)	57.06	0.71	36.10	29.87	972,868.4
3		1	0	[1,3]-[1,6]	(50,70)-(50,75)	57.30	0.73	62.79	70.69	453,761.4
4		10	0	[1,3]-[1,6]	(50,70)-(50,75)	57.22	0.72	52.12	67.05	454,713.8
5	queue	1	5	unrestricted	[3,6]-[2,8]	70.10	1.58	85.93	92.94	313,555.1
6		10	5	unrestricted	[3,6]-[2,8]	69.09	1.49	71.37	86.38	370,742.7
7		1	5	[1,3]-[1,6]	[3,6]-[2,8]	70.04	1.57	82.81	93.83	305,961.9
8		10	5	[1,3]-[1,6]	[3,6]-[2,8]	68.63	1.46	76.02	90.08	303,030.7

Table 2. Exec. times, resource usage, and total costs for different parameters

costs. Furthermore, having unrestricted amounts of resources may be unrealistic in practice. Notice, however, that the difference is not that significant with the queue-based strategy, where resource queues are already representing the accumulated demand. This is indeed what makes this strategy better in general terms than the usage-based strategy.

6 Related Work

Oliveira *et al.* [17] use generalized stochastic Petri nets for correctness verification and performance evaluation of business processes. In their work, an activity can be associated to multiple roles and the completion of an activity can use a portion of the resources available for a role. They also propose metrics for evaluating process performance such as: the minimum number of resources needed for a role in order to complete a process, the expected number of activity instances when completing a process under the assumption of sufficient resources, and the expected activity response time. Colored Petri Nets are used in [16] for understanding how bounded resources can impact the behavior of a process. They introduce the notion of "flexible resource allocation" as a way to assign resources associated to a given role based on priorities. In their approach, they use alternative strategies to better allocate a fixed number of available resources. Havur *et al.* [11] study the problem of resource allocation in business processes management systems where constraints can be assigned to resources (e.g., time of availability) and have dependencies. Their technique is based on the answer set programming formalism and is capable of deriving optimal schedules. Sperl *et al.* [20] describe a stochastic method for quantifying resource utilization relative to structural properties of processes and historical executions. In [9], Maude is used to model and analyze the resource allocation of business processes. In this work, optimal allocation is presented as a multi-objective optimization problem, where response time and resource usage are minimized. None of the aforementioned works attempts at providing analysis techniques or tools for the dynamic allocation of resources with respect to response time and resource usage, as the proposed approach does.

There are many tools supporting the design and management of business processes (e.g., Arena, ARIS10, iGraphx, Signavio, BPMOne, BIMP, Camunda), of which a subset supports the analysis and optimization of processes. This is the case of, for instance, Signavio [2], which packs tools such as the Signavio Process Intelligence for process optimization. This tool automatically mines process models from currently running systems and monitors those processes with the purpose of collecting data that enables end-users to make decisions for process

improvement. Our proposal takes a different approach, since the focus here is on predicting the behavior of designed models given resource provisioning strategies: thus, the approach presented in this work supports the decision making at design time, even before a process is deployed. Then, given a resource allocation strategy, resources are dynamically provisioned and released, respecting the constraints specified as parameters to the process model.

7 Concluding Remarks

This paper focuses on the problem of dynamic resource allocation using BPMN as modeling language for business processes. It presents a version of BPMN extended with annotations for describing the duration of task execution, probabilities in split gateways, and additional information about resources. Given such a process specification, automated techniques are proposed for analyzing its behavior and dynamically adjusting the number of necessary resources following some given adaptation strategy. In this paper, two strategies for resource provisioning (usage-based and queue-based) were presented and illustrated on a concrete example showing how parameters (namely, time between checks, history length, resource ranges, and adaptation thresholds) could be adjusted. The automatic approach was able to handle analyses about the response time and total cost associated to the process. These results were possible thanks to an encoding of the annotated BPMN language into rewriting logic and by using Maude's tools for automating all checks on the concurrent executions of the processes.

Providing mechanisms to automatically finding the best values for given strategies is the first future work direction. This is a multi-objective problem, which is restricted by the concrete nature of the process at hand. The plan is also to investigate on alternative provisioning strategies, with the goal of providing more precise decision criteria. Considering that the provisioning of resources depends on the predictive analysis of the future executions, it is something to also be considered (see, e.g., [10, 19]). Another aim would be at designing and implementing more precise modeling support for the provisioning/releasing procedure, by taking into account aspects such as the time to provision, the releasing cost, etc. Finally, the plan is also to consider a broader form of resources, and to cover resource patterns not currently covered such as the chain and pile-based execution patterns (see [1]).

Acknowledgements. The first author was partially supported by projects PGC2018-094905-B-I00 (Spanish MINECO/FEDER) and UMA18-FEDERJA-180 (J. Andalucía/FEDER). The second author was partially supported by CAPES, Colciencias, and INRIA via the STIC AmSud project EPIC: EPistemic Interactive Concurrency (Proc. No 88881.117603/2016-01) and by via the Colciencias ECOS-NORD project FACTS: Foundational Approach to Computation in Todays Society (Proc. code 63561).

References

1. Workflow resource patterns. BETA Working Paper Series, WP 127, 2004.
2. Signavio. Available: https://www.signavio.com, 2019.
3. G. Agha, J. Meseguer, and K. Sen. PMaude: Rewrite-based Specification Language for Probabilistic Object Systems. *ENTCS*, 153(2):213 – 239, 2006.
4. A. Bouhoula, J.-P. Jouannaud, and J. Meseguer. Specification and Proof in Membership Equational Logic. *Theoretical Comput. Sci.*, 236(1):35–132, 2000.

5. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. Maude: specification and programming in rewriting logic. *Theor. Comput. Sci.*, 285(2):187–243, 2002.
6. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *LNCS*. Springer, 2007.
7. F. Durán, S. Eker, S. Escobar, N. Martí-Oliet, J. Meseguer, R. Rubio, and C. L. Talcott. Programming and symbolic computation in maude. *J. Log. Algebraic Methods Program.*, 110, 2020.
8. F. Durán, C. Rocha, and G. Salaün. Stochastic Analysis of BPMN with Time in Rewriting Logic. *Sci. Comput. Program.*, 168:1–17, 2018.
9. F. Durán, C. Rocha, and G. Salaün. A Rewriting Logic Approach to Resource Allocation Analysis in Business Process Models. *Sci. Comput. Program.*, 183, 2019.
10. C. D. Francescomarino, C. Ghidini, F. M. Maggi, G. Petrucci, and A. Yeshchenko. An Eye into the Future: Leveraging A-priori Knowledge in Predictive Business Process Monitoring. In *Proc. of BPM'17*, volume 10445 of *LNCS*, pages 252–268. Springer, 2017.
11. G. Havur, C. Cabanillas, J. Mendling, and A. Polleres. Resource Allocation with Dependencies in Business Process Management Systems. In *Business Process Management Forum*, pages 3–19. Springer, 2016.
12. ISO/IEC. International Standard 19510, Information technology – Business Process Model and Notation. 2013.
13. A. Krishna, P. Poizat, and G. Salaün. VBPMN: Automated Verification of BPMN Processes. In *Proc. of IFM'17*, volume 10510 of *LNCS*, pages 323–331. Springer, 2017.
14. S. Leemans, D. Fahland, and W. van der Aalst. Scalable Process Discovery and Conformance Checking. *Software & Systems Modeling*, 17(2):599–631, 2018.
15. J. Meseguer. Conditional Rewriting Logic as a Unified Model of Concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.
16. N. Netjes, W. van der Aalst, and H. Reijers. Analysis of Resource-Constrained Processes with Colored Petri Nets. In *Proc. of CPN*, volume 576 of *DAIMI*, pages 251–266, 2005.
17. C. Oliveira, R. Lima, H. Reijers, and J. Ribeiro. Quantitative Analysis of Resource-Constrained Business Processes. *Trans. on Syst., Man, and Cybern.*, 42(3):669–684, 2012.
18. P. C. Ölveczky and J. Meseguer. Semantics and Pragmatics of Real-Time Maude. *Higher-Order and Symbolic Computation*, 20(1-2):161–196, 2007.
19. M. Polato, A. Sperduti, A. Burattin, and M. de Leoni. Time and activity sequence prediction of business process instances. *Computing*, 100(9):1005–1031, 2018.
20. S. Sperl, G. Havur, S. Steyskal, C. Cabanillas, A. Polleres, and A. Haselböck. Resource Utilization Prediction in Decision-Intensive Business Processes. In *Proc. of SIMPDA*, CEUR Workshop Proceedings, pages 128–141, 2017.
21. W. van der Aalst, R. De Masellis, C. Di Francescomarino, and C. Ghidini. Learning Hybrid Process Models from Events - Process Discovery Without Faking Confidence. In *Proc. of BPM'17*, volume 10445 of *LNCS*, pages 59–76. Springer, 2017.
22. C. Walck. Hand-Book on Statistical Distributions for Experimentalists. Technical Report SUF–PFY/96–01, Universitet Stockholms, Stockholm, Sept. 2007.