A Rewriting Logic Approach to Resource Allocation Analysis in Business Process Models

Francisco Durán, Camilo Rocha, Gwen Salaün

A Rewriting Logic Approach to Resource Allocation Analysis in Business Process Models

Francisco Durán

University of Málaga, Málaga, Spain

Camilo Rocha

Pontificia Universidad Javeriana, Cali, Colombia

Gwen Salaün

Univ. Grenoble Alpes, CNRS, Grenoble INP, Inria, LIG, F-38000 Grenoble France

Abstract

This paper presents an approach for the modeling and analysis of resource allocation for business processes. It enables the automatic computation of measures for precisely identifying and optimizing the allocation of resources in business processes, including resource usage over time. The proposed analysis, especially suited to support decision-making strategies, is illustrated with a case study of a parcel ordering and delivery by drones that is developed throughout the paper. The paper comprises an encoding of a significant and expressive subset of the Business Process Model and Notation (BPMN) in rewriting logic, an executable logic of concurrent change that can naturally deal with state and with concurrent computations. The encoding is by itself a formal semantics and interpreter of the BPMN subset that captures *all* concurrent behavior and thus is used to simulate the concurrent evolution of any business process with a given number of resources and replicas.

Key words: Business processes, BPMN, resource allocation, rewriting logic, executable specification, automated verification, Maude.

1. Introduction

Business process optimization is a strategic activity in organizations because of its potential to increase profit margins and reduce operational costs. One of

the main challenges in this activity is concerned with the problem of optimizing allocation and sharing of resources. This is a key question because having a clear answer on how resources are used can help in identifying availability, detect bottlenecks, and improve sharing in order to make processes more efficient. However, analyzing resource usage in business processes is far from being an easy task. First, there is the need for a formal model expressive enough to capture the semantics of processes. Second, resources have to be included in the formal model supporting multiple concurrent executions of a process. Last, automatic verification is required to compute certain metrics and properties of interest (e.g., resource charge, resource occupancy, or usage percentage of each resource replica) on hundreds and even thousands of simulations of a process with its resources.

This paper presents a formal executable model for business processes with resource allocation and automatic verification techniques for analyzing resource usage. The formal model relies on an encoding of a subset of the *Business Process Model and Notation* (BPMN) in rewriting logic [22]. This specification is executable in Maude [9], and supports the concurrent execution of a process with different types of resources and with multiple replicas on any given workload. The verification techniques use Maude's rewriting tools for evaluating expected values of any numerical or path expression in the executable model. The Maude specification supports BPMN activity and collaboration diagrams, four types of gateways, quantitative aspects (such as times associated to flows and tasks, as well as probabilities associated to branching behaviour), loops, resource description, and unbalanced processes (which are workflows without a strict correspondence between split and merge gateways).

The overall idea, from a verification viewpoint, is that multiple concurrent executions of a process compete for the shared resources. With the help of Maude, this approach enables the automatic analysis of how resource usage evolves over time when varying the workload and the amount of resources. This analysis is performed on design models, without the need of an implementation of the system running on real resources. All that is needed is the BPMN specification of the process workflow completed with expert information on the operation of the system. The usefulness of this kind of analysis is illustrated with several experiments on real-world processes, confirming that the approach can help in detecting resource usage problems, thus ultimately leading to the improvement of the business process by optimizing its resource allocation. In particular, the experiments presented in this paper identify, e.g., low-level occupancy of resources and undesirable patterns of resource usage, such as sequential dependencies and bottlenecks provoked by some highly used resources that may induce performance

fall-downs. Furthermore, a recommender system is proposed to find an optimal assignment of resources following Monte Carlo simulations of the business process.

In summary, the main contributions of this work are:

(i) a formal semantics captured by an encoding in rewriting logic of a significant subset of BPMN, including activity and collaboration diagrams, four types of gateways, unbalanced workflows, loops, resource description, and concurrent process execution;

(ii) automated formal analysis of resource allocation properties (e.g., accumulated time of occupancy for resources) performed at design time and a recommender system computing the optimal assignment of resources;

(iii) the evaluation of the approach on a workbench of real-world BPMN processes.

An early version of this paper has been published in [12] and has been significantly extended here as follows:

(i) a version of BPMN with explicit description of resources is considered;

(ii) the Maude encoding has been extended to support resources and the multiple execution of a process (tokens, workload, scheduling) required for simulating the evolution of resource usage over time;

(iii) several properties of interest regarding average execution time and resource occupancy, that can be analysed using the proposed framework, have been identified;

(iv) the approach has been applied to several case studies for validation purposes; and

(v) the introduction, related work section, and conclusion have been fully revised, by taking the aforementioned extensions and improvements into account.

The rest of the paper is organized as follows. Section 2 introduces the main features of BPMN. Section 3 briefly presents rewriting logic and explains the encoding of the considered subset of BPMN. Section 4 presents how resource analysis is achieved using Maude's rewriting-based tools. This section also describes several experiments and illustrates how the automatic verification can help in improving process definitions. Section 5 surveys related work and Section 6 concludes the paper.

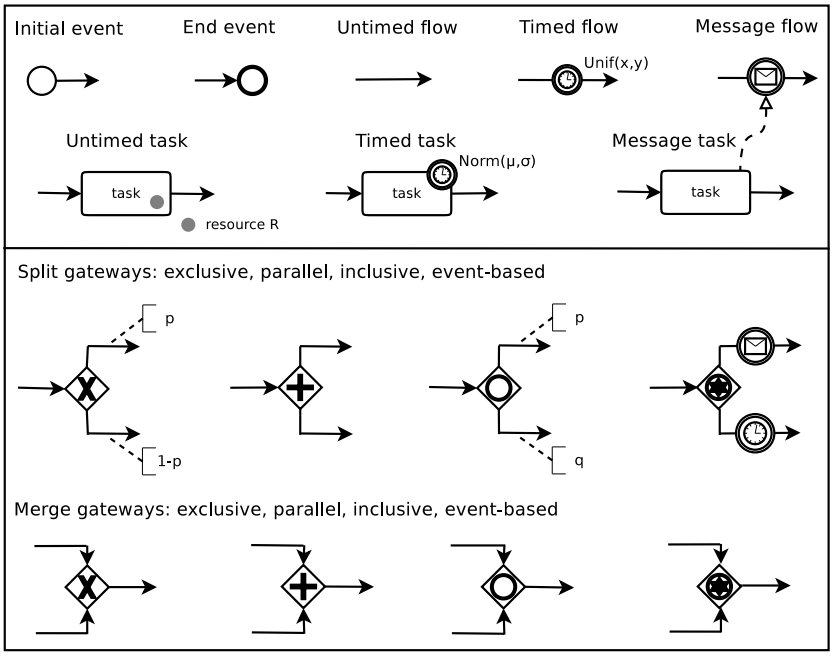

Figure 1: Supported BPMN syntax.

2. Business Process Model and Notation

BPMN 2.0 (BPMN, as a shorthand, in the rest of this manuscript) was published as an ISO/IEC standard [17] in 2013 and is nowadays extensively used for modeling business processes. In this paper, two main kinds of BPMN diagrams are considered, namely, activity and collaboration diagrams. In particular, the focus is on the BPMN elements related to control-flow modeling and behavioral aspects that can be represented in BPMN constructs. Beyond those constructs, resource description and allocation are also considered, for which an automated analysis method is proposed.

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.

Specifically, the node types *event*, *task*, and *gateway*, and the edge type *sequence flow* are considered. Start and end events are used, respectively, to initialize and terminate processes. A task represents an atomic activity that has exactly

one incoming and one outgoing flow. A task may have a duration (expressed as a stochastic expression) and may produce an event message. A sequence flow describes two nodes executed one after the other, i.e., by imposing an execution order with possible delays. The timing information associated to tasks and flows (durations or delays) is described either as a literal value (a non-negative real number, possibly 0) 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., [34]).

Gateways are used to control the divergence and convergence of the execution flow. Four types of gateways are considered: *exclusive*, *inclusive*, *parallel*, and *event-based*. Gateways with one incoming branch and multiple outgoing branches are called *splits*, e.g., split inclusive gateway. Gateways with one outgoing branch and multiple incoming branches are called *merges*, e.g., merge parallel gateway. An exclusive gateway chooses one out of a set of mutually exclusive alternative incoming or outgoing branches. For an inclusive gateway, any positive number of branches among all its incoming or outgoing branches may be taken (both BPMN 1.0 and 2.0 semantics for inclusive gateways are supported). A parallel gateway creates concurrent flows for all its outgoing branches or synchronizes concurrent flows for all its incoming branches; event-based gateway takes one of its outgoing branches or accepts one of its incoming branches based on events. Event-based split gateways may have a default branch fired by a timeout.

Data-based conditions for split gateways are modeled using probabilities associated to outgoing flows of exclusive and inclusive split gateways. 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. Workflows with looping behavior are supported, as well as unbalanced workflows.

Each lane in a collaboration diagram corresponds to a specific role or to a specific resource. Collaboration diagrams are also often used for modeling distributed systems and, in that sense, they heavily rely on message events and event-based gateways. Once completed, a task may generate a message with a flow in another lane as target. An event-based split gateway is triggered whenever a message is available from one of its outgoing flows or when its timer event has completed.

Instead of implicitly associating resources to lanes, resources are explicitly defined at the task level. A task that requires resources can include, as part of its specification, the number of required instances (or replicas) of a resource. Indeed, several tasks could compete for the same resources. Furthermore, since multiple instances of a same process may be executed concurrently, different instances may

also access and thus compete for the shared resources. Section 3 presents how the number of available resources can be specified and Section 4 how the optimal number of resources can be computed.

The execution semantics of BPMN is informally described in official documents such as [28, 17]. Although the focus here is on the provided analysis techniques, this work also gives a formal semantics to BPMN with time and resources. Last but not least, in this paper, BPMN processes are assumed to be syntactically correct. This can be enforced using existing works and tools, e.g., [15], the Activiti BPM platform, Bonita BPM, or the Eclipse BPMN Designer.

Running example. To introduce and illustrate the use of these elements and the approach presented in this paper, Figure 2 presents a process describing a parcel ordering and delivery by drones. This BPMN process is presented as a collaboration diagram consisting of three lanes, one for the client, one for the order management and one for the delivery process. In this example, the client first signs in and then repeatedly looks for products. Eventually, the client can decide to give up (termination) or to make an order by submitting it to the order management lane. The client then waits for a response (acceptance or refusal of this order). If the order can be completed, the client pays for it and then receives the parcel. Otherwise (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, a loop (Search products task), and unbalanced structures (for the event-based gateways used in the client lane and the exclusive gateway after the initial availability check).

To simplify the exposition of the running example (Figure 2), the delays in all flows are set to 0 and the specification of the task duration has been placed at the bottom-left corner of the process description. 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 $[3, 30]$. Exclusive split gateways are modeled using probabilities associated to outgoing flows. 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. All

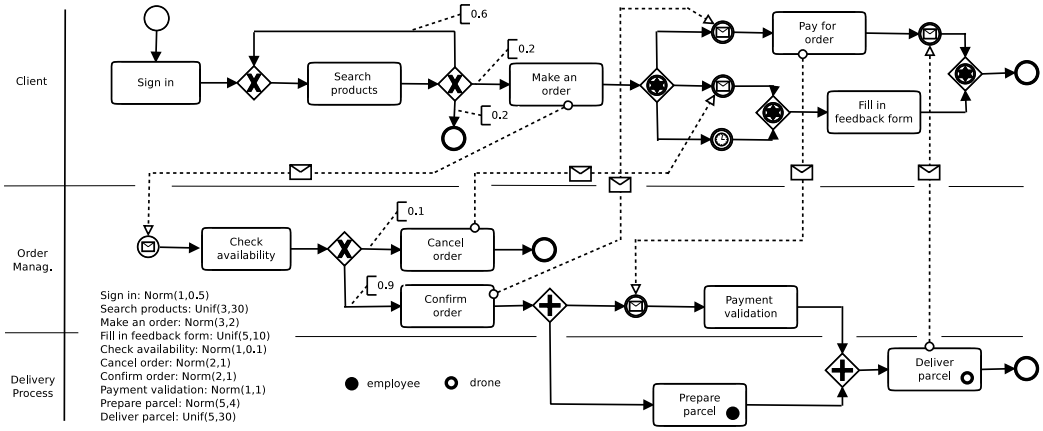

Figure 2: Running example: parcel delivery by drones.

these numbers (durations and probabilities) can be provided by experts or can be obtained by executing the process in practice on a reasonable period of time (a few days) and by extracting this information from the generated logs.

The process in Figure 2 relies on employees for parcel packing and drones for parcel delivery. Notice the small circles at the bottom-right corner of the Prepare parcel and Deliver parcel tasks, indicating that one instance of the employee resource and another one of the drone resource are required, respectively, for the tasks completion.

3. From BPMN to Maude

This section presents the encoding of BPMN processes including time and resources in Maude. The Maude system is chosen as the formal specification language and verification platform for several reasons. Its language is expressive enough for specifying all the aspects of BPMN introduced in the former section. It also offers several rewriting-based tools that are adequate for simulating the specification, and for computing numerical results for timing properties and resource analysis. This section first gives an overview of rewriting logic and the Maude framework. The Maude specification of BPMN is then described in two parts: the encoding of the process structure and the description of the semantics of the BPMN subset using rewrite rules. The complete Maude implementation is available online at [11].

3.1. Rewriting Logic and Maude Overview

Rewriting logic [22] is a logic of change that can naturally deal with state and with highly nondeterministic concurrent computations. A rewrite logic theory is a tuple $(\Sigma, E \cup A, R)$, where $(\Sigma, E \cup A)$ is a membership equational logic [4] theory with Σ its signature, E a set of conditional equations and sort membership axioms, A a set of equational axioms (e.g., associativity, commutativity and identity) so that rewriting is performed modulo A, and R is a set of labeled conditional rules.

Maude [9] is a high-level language and a high-performance interpreter that supports membership equational logic and rewriting logic specification and programming of systems. Thus, Maude integrates an equational style of functional programming with rewriting logic computation. Thanks to its efficient rewriting engine and its metalanguage capabilities, Maude turns out to be an excellent tool for creating executable environments of various logics, models of computation, theorem provers, or even programming languages.

A functional specification must be terminating, confluent, and sort-decreasing. Computation in a functional module is accomplished by using the equations as simplification rules from left to right until a canonical form is found. Some equations, such as those expressing the commutativity of binary operators, are —however— not terminating. Nonetheless, they are supported by means of *operator attributes*, so that Maude performs simplification modulo the equational theories provided by such attributes, which can be associativity (assoc), commutativity (comm), identity (id), and idempotency (idem). The above properties must therefore be understood in the more general context of simplification *modulo* such equational theories.

In Maude, a distributed system is axiomatized by a rewrite theory describing its states as an algebraic data type (an equational sub-specification) and a collection of conditional rewrite rules specifying its *behavior*. Rewrite rules are written crl [l] : t => t' if C, with l the rule label, t and t' terms, and C a guard or condition. Rewrite specifications are not required to be terminating nor confluent. Rules describe the local, concurrent transitions that are possible in the system, i.e., when a part of the system state fits the pattern t, then it can be replaced by the corresponding instantiation of t'. The guard C acts as a blocking precondition: a conditional rule can only be fired if its condition is satisfied. Sometimes rules are given without label or condition (which can be assumed to be true).

In the Maude language, object-oriented systems can be specified by object-oriented modules in which classes and subclasses are declared, with the usual support for inheritance, dynamic binding, etc. A class is declared with syntax

class $C \mid a_1 : S_1, \ldots, a_n : S_n$, where C is the name of the class, a_i are attribute identifiers, and S_i are the sorts of the corresponding attributes. The objects of a class C are then record-like structures of the form $< O : C \mid a_1 : v_1, \ldots, a_n : v_n >$, where O is the name of the object and v_i are the current values of its attributes.

In a concurrent object-oriented system, the concurrent state, which is called a *configuration*, consist of a multiset of objects and messages. Rewrite rules then define transitions between such configurations. These transitions represent the different actions that may occur in the system. For instance, there will be rules modeling the effects of events, or the synchronous and asynchronous communication events of objects and messages. The general form of a rewrite rule `r` is the following:

```
crl [r] :
  < O1 : C1 | atts1 > ... < On : Cn | attsn >
  M1 ... Mm
  => < Oi1 : C'i1 | atts'i1 > ... < Oik : C'ik | atts'ik >
     < Q1 : C''1 | atts''1 > ... < Qp : C''p | atts''p >
     M'1 ... M'q
  if Cond .
```

where

- r is the rule label,
- $M_1 ... M_m$ and $M'_1 ... M'_q$ are messages,
- $O_1 ... O_n$ and $Q_1 ... Q_p$ are object identifiers,
- $C_1 ... C_n$, $C'_{i_1} ... C'_{i_k}$ and $C''_1 ... C''_p$ are classes, $i_1 ... i_k$ is a subset of $1 ... n$, and
- *Cond* is a Boolean condition (the rule's *guard*).

The result of applying such a rule is that:

- messages $M_1 ... M_m$ disappear, i.e., they are consumed,
- the state, and possibly the classes of objects $O_{i_1} ... O_{i_k}$ may change,
- all the other objects O_j vanish,
- new objects $Q_1 ... Q_p$ are created, and
- new messages $M'_1 ... M'_q$ are created, i.e., they are sent.

The real-time aspects are modeled using Real-Time Maude [27], which supports the formal specification and analysis of *real-time systems*. Specifically, Real-Time Maude provides a sort Time to model the time domain, which in this work is assumed to be dense (i.e., represented as rational numbers). Then, given a system configuration and a time amount, time elapse is modeled with *tick rules* like

```
crl [l] : { t, T } => { t', T + τ } if C .
```

where t and t' are system states, T is the global time, and τ is a term of sort Time that denotes the *duration* of the rewrite, and that affects 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 [27]. Although there are other sampling strategies, in the most convenient one this single tick rule models time elapse by using 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. Then, time advances non-deterministically by any amount τ, which must be less than or equal to the maximum time elapse of the system.

Note that the above is a general form for the tick rules. As it will be seen in the following sections, the concrete representation of states and global time depend on the concrete system specification. This simple approach allows for having multiple clocks and timers, and model the different real-time events that could happen in the system quite naturally.

3.2. Process Description

In the Maude specification of BPMN, a process is represented as an object with sets of flows and nodes as attributes. Nodes can be of five different types: start, end, task, split, or merge. The representation of each of these types of elements includes the necessary information. 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 a 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. 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

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

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

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 automated by an extension of the VBPMN platform [18].

3.3. Execution semantics

The operational semantics of BPMN is defined using rewrite rules, modeling how tokens evolve through a process, thus defining the execution semantics of BPMN. Each observable action is modeled as a rewrite rule. For instance, when a token arrives at a parallel split gateway, the token corresponding to the incoming flow is removed, and one token is added for each outgoing flow. Technically, rewrite rules operate on systems composed of a process object and a *simulation* object.

Simulation object. While the process object introduced in Section 3.2 represents the BPMN process and does not change during an execution, the simulation object keeps information on the 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 quantities being measured during the analysis of a process. Figure 4 presents the structure of the Simulation object.

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 are simultaneously happening, each execution has a unique identifier, and tokens

```
< s : Simulation | tokens : ...,         ---- scheduler
                   gtime : ...,          ---- global time
                   resources : ...,      ---- resource set
                   events : ...,         ---- event set
                   process-execs : ....,  ---- execution times
                   sync-times : ...,     ---- synchronization times
                   task-times : ...,     ---- task execution times
                   ... >
```

Figure 4: Representation of the Simulation object.

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 may be consumed.

Scheduling. Tokens are stored in a *scheduler* implemented as a priority queue, so that they are kept according 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. A message event may be associated to a flow, and the flow is blocked until the message is received. A timer event may be associated to a flow and has a delay as parameter. When the flow of execution 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.

Resources. Each resource is represented 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 all of them at once, or waits for them to become available.

```
1  crl [startProc] :
2      < PId : Process | nodes : (start(NId, FId), Nodes),
3                        flows : (flow(FId, SE), Flows),
4                        Atts >
5      < SId : Simulation | tokens : (token(TId, NId, 0) Tks), ... Atts1 >
6      < CId : Counter | counter : N >
7   => < PId : Process | nodes : (start(NId, FId), Nodes),
8                        flows : (flow(FId, SE), Flows),
9                        Atts >
10      < SId : Simulation | tokens : insert(Tks, token(TId, FId, T')), ... Atts1 >
11      < CId : Counter | counter : N' >
12    if {T', N'} := eval(SE, N) .
```

Figure 5: Start event processing.

Workloads. Simulation-based analysis techniques are typically parameterized by the workload that represents the way a system is used. They define the rate at which new instances of a given process are executed. Currently, closed workloads can be handled by specifying the number of executions and the rate at which executions are started, that is, their inter-arrival times. Both the number and the rate are specified as stochastic expressions.

Rewrite rules for BPMN constructs. Rewriting rules represent how tokens evolve through the process and events are fired, thus defining the execution semantics of BPMN. Each action supported by the system is modeled as a rewrite rule. These rules are overviewed in the rest of this section to gather an intuition on the formal semantics (see [11] for the complete specification).

Start/end events. Figure 5 depicts the rule for the start event. When there is a token in the execution TId in the start node NId with delay 0 (note the token at the front of the scheduler in the Simulation object in line 5), then this rule generates a new token on the outgoing flow of the selected node to initiate the execution of a process instance (line 10). The insert function puts this token in the scheduler and the eval function evaluates the stochastic expression SE specifying the delay of the outgoing flow FId to be assigned to the new token. Details on the initialization of time stamps and recorded times for the initiated execution have been replaced by ellipses. A termination rule, associated to stop events, consumes tokens when they arrive at those events.

Tasks. A task execution is modeled with two rules. The first rule, the initTask rule shown in Figure 6, represents the task initiation, which is applied when a token with zero time is available for the incoming flow (line 5). If all the re-

```
1  rl [initTask] :
2     < PId : Process |
3        nodes : (task(NId, TaskName, FId1, FId2, SE, RIds, SEI), Nodes), Atts >
4     < SId : Simulation |
5        tokens : (token(TId, FId1, 0) Tks),
6        task-tstamps : TTSs, gtime : T, resources : Rs, Atts1 >
7     < CId : Counter | counter : N >
8  => if allResourcesAvailable(RIds, Rs)
9     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
16                            fi,            ---- for loops, stamps get overwritten
17             gtime : T,
18             resources : grabResources&updateTime(RIds, Rs, time(eval(SE, N)), T), Atts1 >
19          < CId : Counter | counter : int(eval(SE, N)) >
20     else ...                              ---- if necessary, the scheduler is updated
21     fi .
```

Figure 6: Task initiation rule.

sources required by this task are available, which is checked with the allResourcesAvailable function (line 8), 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 20). If available, all required resources are removed from the set of resources, and the time those resources have been in use is updated (grabResources&updateTime function, line 18). Since all auxiliary functions in the right-hand side of the initTask rule are defined equationally, the checking and grabbing of resources are performed atomically, without introducing any blocking issues. Note also that rules update the information on execution times, task durations, etc. (see, e.g., the update of the task-tstamps attribute, lines 13-16). This information is important for analysis purposes, as it will be seen in Section 4.

A second rule, which models task completion, is triggered when there is a token for that task with zero time. In that case, the token is consumed and a new one is generated for the outgoing flow. All resources are released, and all the message events associated to that task, if any, are added to the events set.

Exclusive gateways. There are two rules for the exclusive gateways, namely, one for the split and one for the merge. The rule for the split applies when a token with zero time is available on its incoming flow. A uniformly sampled probability distribution is used to choose the branch to be executed. The newly created token

is assigned with its run-to-completion time generated by evaluating the stochastic expression associated to the chosen outgoing flow—this is actually the case every time a new token is added for a flow. The exclusive merge gateway is triggered when one of its incoming flows has a token with zero time. In that case, a new token is generated, assigned to the outgoing flow, and added to the scheduler.

Parallel gateways. The parallel split gateway rule is triggered when a token with zero time corresponding to the input flow is available. If so, the token is consumed and one token is added to each of its outgoing flows. The merge rule for the parallel gateway is executed when there is a token with zero time for each incoming branch. In that case, these tokens are removed and a new token is generated for the outgoing flow. In the merge rule, synchronization times are also updated.

Inclusive gateways. The split rule applies when a token with zero time is available at the incoming flow. Since all outgoing branches are equipped with probabilities, a function in charge of computing the subset of branches to be triggered is invoked. For each one of the selected branches, a new token is added to the scheduler. Regarding merge gateways, both BPMN 1.0 and 2.0 semantics are supported in this research. In BPMN 2.0, merge inclusive gateways behave like exclusive ones. The 1.0 version of the semantics is more involved [8], since the merge rule for the inclusive gateway is executed when all the expected tokens are available with zero time. This requires a global analysis. To check whether all expected tokens have arrived, a backwards traversal that explores the process upstream and checks whether there are tokens on their way to that merge is performed. In both cases, once the 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 updated with synchronization times.

Event-based gateways. 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. Two additional rules handle the possible cases. If there is an outgoing flow with a message in the set of events, then that branch can be activated and one token is added for that flow. This behavior is formalized with the splitGatewayEventBased rule shown in Figure 7, where an event msg(MId) is in the events set and MId is the message associated to the flow FId1, which is one of the outgoing flows of the split NId. A second rule handles the case in which none of the expected messages is in the set of events, but there is at least one timer for which the time has expired. In this situation, a token is added to the outgoing flow corresponding to that timer.

```
1  rl [splitGatewayEventBased-msg] :
2     < PId : Process |
3         nodes : (split(NId, eventbased, FId, (FId1, FIds)), Nodes),
4         flows : (flow(FId1, SE, message(MId, MD)), Flows),
5         Atts >
6     < SId : Simulation |
7         tokens : (token(TId, NId, 0) Tks),                        ---- token is at gateway
8         events : ((TId |-> (msg(MId), Evs)), ME),                 ---- message is available
9         Atts1 >
10     < CId : Counter | counter : N >
11  => < PId : Process |
12         nodes : (split(NId, eventbased, FId, (FId1, FIds)), Nodes),
13         flows : (flow(FId1, SE, message(MId, MD)), Flows),
14         Atts >
15     < SId : Simulation |   ---- put token in the corresponding outgoing flow
16         tokens : insert(Tks, (token(TId, FId1, time(eval(SE, N))))),
17         events : ((TId |-> removeTimer(FIds, Flows, Evs)), ME), ---- remove timer
18         Atts1 >
19     < CId : Counter | counter : (N + int(eval(SE, N))) > .
```

Figure 7: Event-based split gateway: one message available.

Loops and unbalanced workflows. The modeling of the BPMN execution semantics using tokens and their circulation through the process structure supports intricate constructs such as loops and unbalanced workflows. As far as looping behavior is concerned, a token may circulate back to an already visited flow without any additional treatment. Similarly, tokens can advance through flows that are part of balanced or unbalanced gateways, independently of their structure.

4. Resource Allocation Analysis

This section illustrates how resource allocation analysis can be performed with the proposed approach using the running example. It also includes a discussion on how the analysis can help in refining and improving resource allocation. Specifically, given the process description, a specification of resources and a workload, the experiments illustrate how information on execution times and resource usage is collected. This information is used to find the optimal allocation of resources that minimizes costs and execution times relative to an optimization goal. The interested reader is referred to [11] for further details on the experiments carried out on the running example.

4.1. Properties

The BPMN subset encoded in Maude is quite expressive and several kinds of properties can be computed, including timing and resource-based properties. These properties are meaningful when executing multiple instances of a process that compete for the shared resources. As for *timing properties*, the approach presented in this paper allows the computation of average execution times (AET) of a process, its variance (Var), and the average synchronization time (AST) for merge gateways, representing the time elapse from the arrival of the first token through one of its incoming flows to its activation. Synchronization times make sense only for parallel and BPMN 1.0 inclusive gateways, since there is no waiting nor synchronization time for the other gateways.

As far as *resource-based properties* are concerned, which is the main focus in this work, the following properties are computed:

- The global time usage of all instances of each resource R (GTU_R). E.g., when executing 10 instances of a process P, with an AET of 42, it is possible that the two instances of a resource A are used for 56 time units and the three instances of resource B for 60 time units.
- The expression GTU_R^1 denotes the average GTU of resource R (i.e., the GTU per instance of resource R). Thus, although in the previous example GTU_B is greater than GTU_A, GTU_A^1 is 28 and GTU_B^1 is 20.
- The average usage percentage UP_R for a resource R over the global execution time. E.g., continuing with the running example, in average, an instance of the resource A is used 24% of the global execution time when executing 200 instances of a process P.

To verify these properties, Maude rewriting capabilities are used in order to simulate and extract analysis results on a given BPMN process. The simulation object presented in Section 3 is used to accumulate information of synchronization times, task duration, and resource usage. At the end of all executions, these results are used for computing the expected average times and resource usage percentages. Since the analyzed processes are assumed syntactically correct and processes that may lead to non-terminating analysis are not considered (e.g., loops without end events), the verification process always terminates. This is the case because all splits are probabilistic, and time duration and probabilities assigned to the branches respect specific assumptions (e.g., all probabilities are between 0 and 1, they sum up to 1 in exclusive branches, and times are positive).

Num. inst.	AET	Var	AST_{g8}	AST_{ee}	Total time	Resources						Anal. time
						GTU_e	GTU_e^1	UP_e	GTU_d	GTU_d^1	UP_d	
100	107	190	70	57	327	316	158	48	852	284	87	6s
200	160	37	81	108	599	506	253	42	1665	555	93	30s
400	301	213	107	252	1202	1202	601	43	3493	1164	97	225s
800	550	4	156	501	2367	1978	989	42	6953	2318	98	1748s
1600	910	56	250	862	4263	4005	2003	47	12648	4216	99	11828s

Table 1: Experimental results for the running example (2 employees, 3 drones).

4.2. Evaluation

Table 1 summarizes experimental results on execution times and resource usage on the parcel order and delivery example (Section 2). They were carried out on an iMac with 3,2GHz Intel Core i5 and 8GB of RAM. All simulations were performed assuming a given workload with a number of instances (first column) and an exponentially distributed interarrival time ($\lambda = 4$). Columns 2 to 6 contain, respectively, the average execution time (AET), its variance (Var), the average synchronization time for the parallel merge at the end of the delivery process lane (AST_{g8}), the average synchronization time for the end events (AST_{ee}), and the total time to complete the execution of all instances. The next six columns show results on resource usage for employees and drones—note the use of subindices e and d for employees and drones, respectively. The final column gives the overall time needed to complete the analysis. All times are logical (time units), except for the ones in the last column that are given in seconds. Other information, such as the duration of each task and the synchronization time of each merge gateway is also collected.

These experiments consist of 100, 200, 400, 800, and 1600 instances for 2 employees and 3 drones. Note that the average execution and synchronization times clearly increase with the number of instances. This is because the more tokens compete for resources, the more time it takes to execute the process and for the tokens to reach the synchronization points. Note the relationship between AET and AST_{ee} times, showing an unbalance between the three lanes: the client lane terminates earlier than the two other lanes, which exhibit a bottleneck because of the demand on the resources.

The global time (GTU) for each or all the instances increases with the number of executed instances. These times are particularly interesting because they can be materialized as costs (e.g., cost of a resource, salary of an employee/all

employees). In relation with usage percentage (UP), the results indicate that the employees are "underused" since they work around 40% of the time, in contrast to the drones that are constantly busy and used about 90% of the time for delivering parcels. This may suggest an inappropriate allocation of resources. It is worth observing that, although the number of instances clearly affects all computed times, the results for resource usage (UP) are quite stable and a small number of instances is enough for obtaining a good approximation of these percentages.

Resource allocation impacts execution times (AET) and resource usage (UP) of a process. Figure 8 focuses on average execution time and depicts the results when the number of employees and drones vary for a fixed number of executions (400). The objective here is to reduce the average execution time for completing the process: the quicker the parcel is delivered, the more satisfied the client is. It can be observed that, independently of the number of employees, execution times are not satisfactory with 1 or 2 drones (between 400 and 800 time units). The time becomes reasonable for more than 3 drones (less than 300 time units) and tends to stabilize. It is also worth noting that, given its low usage rate, the number of employees does not impact significantly the execution time. For more than six drones, only going from one to two employees makes a significant impact in the AET values.

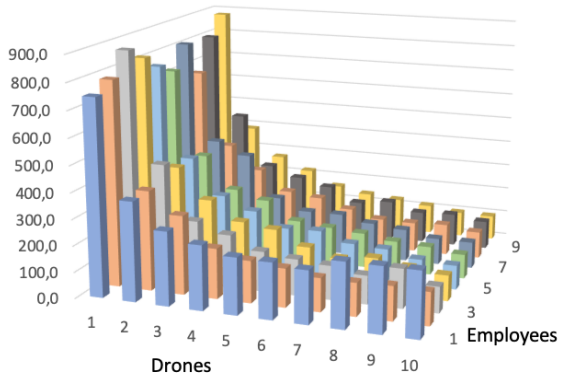

Figure 8: Running example (400 instances): average execution time.

Figure 9 gives a different point of view of resource usage by concentrating on each resource instance. Figure 9 (left) shows that employees are close to 100% usage only if there is 1 or 2 instances of that resource and at least 4 or 8 drones, respectively. If the number of employees increases, the usage percentage quickly drops, reaching a low level (e.g., 14% for 4 employees and 2 drones). This per-

centage slightly increases with the number of drone instances (e.g., 34% for 4 employees and 5 drones), but remains low (around 30%). Figure 9 (right) shows that the drone usage is always quite high whatever the number of employees is. With only 1 or 2 drones, the usage percentage is almost at 100% and slightly decreases with 4 drones. When there are 6 drones and 1 employee, the percentage is still about 60%. Another interesting fact is that the number of employees barely impacts the drone usage percentage. For example, with 4 drones, the usage percentage is around 90% for any number of employees (varying from 1 to 10).

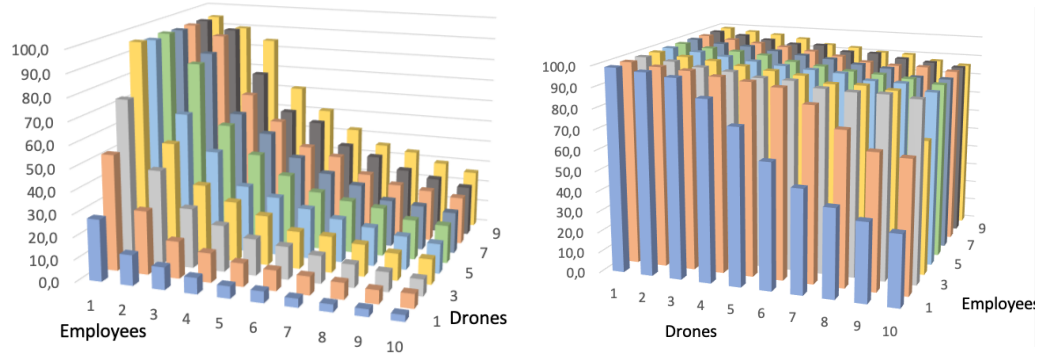

Figure 9: Running example (400 instances): average usage percentage per employee (left) and drone (right).

4.3. Optimal Resource Allocation

The collected data on execution times and resource usage can be used to compute the optimal allocation of resources for the running example. Given the data represented in Figures 8 and 9, an optimization problem can be expressed as follows. Given expressions f_{cost} and f_{aet} representing the normalized functions for drone and employee cost (per time unit) and for average execution time, and given w_i with $i \in \{cost, aet\}$ their respective weights, the following multi-objective optimization problem

$$\min_{x \in X} \sum_{i \in \{cost,\, aet\}} w_i f_i(x),$$

for which Pareto optimal solutions are required, can be solved. For instance, assuming that the cost per hour of a drone and an employee is 20€ and 50€, respectively, Figure 10 depicts normalized costs and average execution times. If the preference is to minimize delivery time, specified by, say, weights w_{cost} =

0.4 and $w_{aet} = 0.6$, the best solution would be the one marked in the figure, representing the combination of 3 employees and 8 drones.

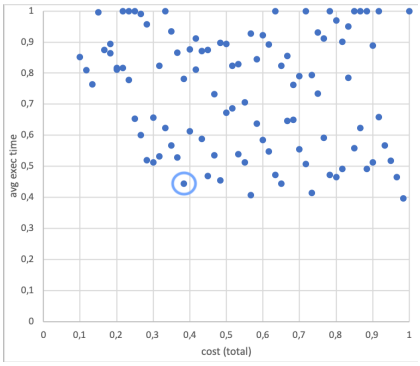

Figure 10: Cost vs. average execution time.

Table 2 shows the results of the simulations for the running example, with 3 employees and 8 drones. If the results in Tables 1 and 2 are compared, an improvement in all aspects can be observed, as expected. Specifically, average execution times keep variance values and synchronization times at lower rates. With this new assignment of resources, AETs have dropped significantly. E.g., from 910 to 302 for the 1600-instances case. The total time has similarly been reduced in all cases, e.g., from 4263 down to 1627 in the 1600-instances case. To conclude, this improvement is a consequence of a more balanced use of the assigned resources. Employees are now used in the range 31%-77% of their time (whereas they were in the range 42%-48% with the previous assignment). The load for drones has slightly dropped, as a consequence of the better balanced load of employees and drones.

Num. inst.	AET	Var	AST_{g8}	AST_{ee}	Total time	Resources						Anal. time
						GTU_e	GTU_e^1	UP_e	GTU_d	GTU_d^1	UP_d	
100	66	18	69	18	243	228	76	31	884	111	45	4s
200	73	0.3	76	27	232	434	145	62	1443	180	78	18s
400	109	67	102	63	460	865	288	63	3271	409	89	138s
800	181	7	156	135	873	1840	613	70	6540	817	94	1060s
1600	302	0.4	252	254	1627	3764	1255	77	12545	1568	96	7946s

Table 2: Experimental results for the running example (3 employees, 8 drones).

4.4. Search-Based Optimization

The analysis performed in the previous section requires the analysis of all possible combinations of numbers of resource instances. In the example in the previous section, the analysis was performed for the range [1..10] for both employees and drones. Considering more resources and wider ranges the analysis could be very costly.

It is a classical optimization problem and it can be solved by embedding the above-mentioned techniques for evaluating a specific resource assignment into some of the heuristic-based search optimization techniques. The possibility of using the techniques in the previous section into a simple gradient descent is illustrated next. Gradient descent is intended for finding local minimum (respectively, maximum) values starting from a specific point. After analyzing the function in the neighbouring points, the one with greatest value change is identified. Since the goal is to find a local minimum, the smallest value is chosen. If none of the neighbours is smaller, then the actual point is a local minimum.

By using the gradient descent algorithm on the running example, starting from the assignment ($employee \mapsto 4, drone \mapsto 4$), and using the optimization function and values in Section 4.3, the optimal solution can be found by following the sequence $(4,4) \longrightarrow (3,5) \longrightarrow (2,6) \longrightarrow (2,7) \longrightarrow (3,8)$. The values for which the optimization function is computed by our search function and the corresponding evaluations is depicted in Table 3. When compared to the analysis in Section 4.3, only 27 combinations were analyzed, instead of the 100 combinations analyzed above. More importantly, this dynamic search enables the exploration of greater search spaces. However, notice that this is a local minimum. A more sophisticated search algorithm may be required to find a global minimum, such as annealing or genetic algorithms.

4.5. Additional Examples

The BPMN process depicted in Figure 11 describes the recruitment process in an enterprise. This process shows how a candidate must fill in a hiring form, carry on a medical checkup, and in some cases apply for visa. Once the documentation is submitted, it is checked by the human resources office, which can decide to accept, reject or request additional documentation. If accepted, the candidate is informed, an assistant is in charge of preparing a welcome kit, and the technical staff is in charge of including the corresponding data in the enterprise's DB.

In this example, there are three resources modeled, namely human resources, the assistants, and the technical staff. The execution of the gradient descent algorithm on the recruitment example, starting from the assignment ($hr \mapsto 3, assistant \mapsto$

		Drones									
		1	2	3	4	5	6	7	8	9	10
Employees	1					0,591	0,591	0,567	0,687		
	2				0,560	0,492	0,467	0,425	0,428	0,446	
	3			0,514	0,621	0,508	0,480	0,464	0,420	0,528	
	4			0,622	0,685	0,653	0,527	0,461	0,508	0,466	
	5			0,626	0,731	0,603					
	6										
	7										
	8										
	9										
	10										

Table 3: Computation using the gradient descent search algorithm (starting with the assignment ($employee \mapsto 4, drone \mapsto 4$)).

$3, technicalstaff \mapsto 3$), using the optimization function and values in Section 4.3, with costs (per hour) ($hr \mapsto 50, assistant \mapsto 25, technicalstaff \mapsto 35$), a local optimal solution is found by following the sequence

$$(3,3,3) \longrightarrow (4,3,3) \longrightarrow (4,3,2) \longrightarrow (4,3,1) \longrightarrow (5,3,1) \longrightarrow (5,2,1).$$

If the range [1..10] is considered, then only 63 assignments are analyzed instead of $10 \times 10 \times 10$ cases.

The BPMN process depicted in Figure 12 describes a visa application process. The visa requester must provide a scanned copy of her passport and pay certain fees. The scanned passport is checked by an employee. If everything is all right, the visa is granted and delivered to the requester.

In this example, there are two resources modeled, namely the employee and the visa printer. Note that the task Deliver visa requires an instance of each of the resources. The execution of the gradient descent algorithm on the visa application example, starting from the assignment ($employee \mapsto 1, printer \mapsto 5$), using the optimization function and values in Section 4.3, with costs (per hour) ($employee \mapsto 50, printer \mapsto 5$), a local optimal solution is found by following the sequence

$$(1,5) \longrightarrow (2,5) \longrightarrow (2,4) \longrightarrow (2,3) \longrightarrow (2,2) \longrightarrow (2,1).$$

If the range [1..10] is considered, then 24 assignments are analyzed instead of

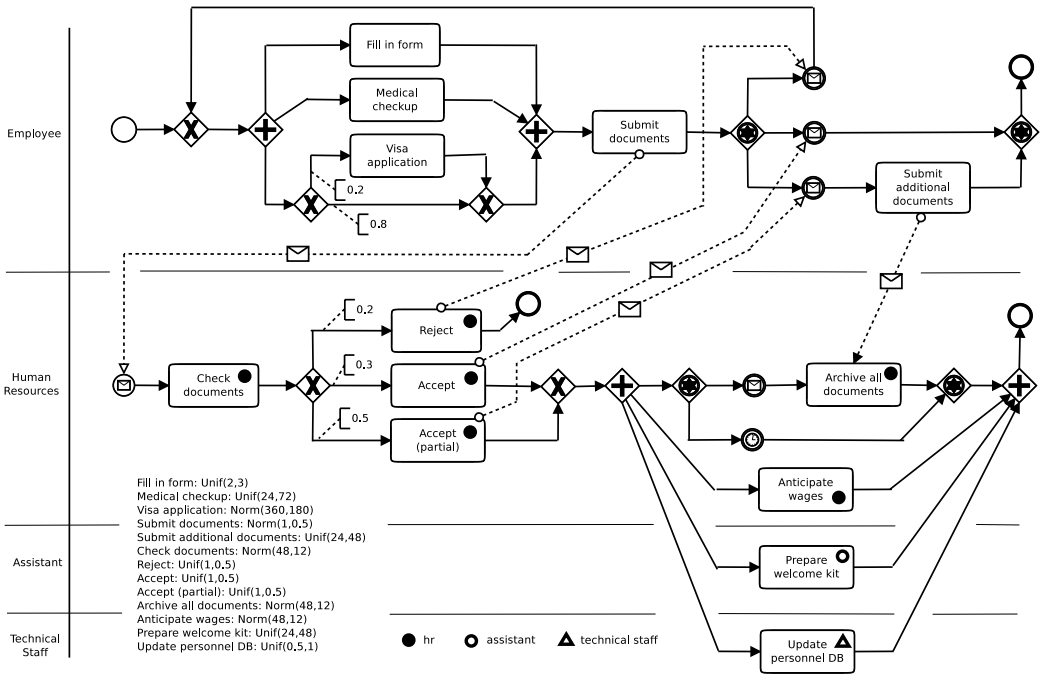

Figure 11: BPMN process of the recruitment example.

10×10 cases.

5. Related Work

Resource allocation—including optimization, allocation constraints, and optimal schedule allocation—has been extensively explored for the business process domain from several formal methods approaches. Schömig and Rau [32] use colored stochastic Petri nets to specify and analyze business processes in the presence of dynamic routing, simultaneous resource allocation, forking/joining of process-control threads, and priority-based queueing. In their work, each resource is equipped with properties grouped in a role defining if the resource is eligible to perform a certain activity. Li *et al.* [19] introduce *multidimensional workflow nets* to model and analyze resource availability and workload. Oliveira *et al.* [25] 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 to 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

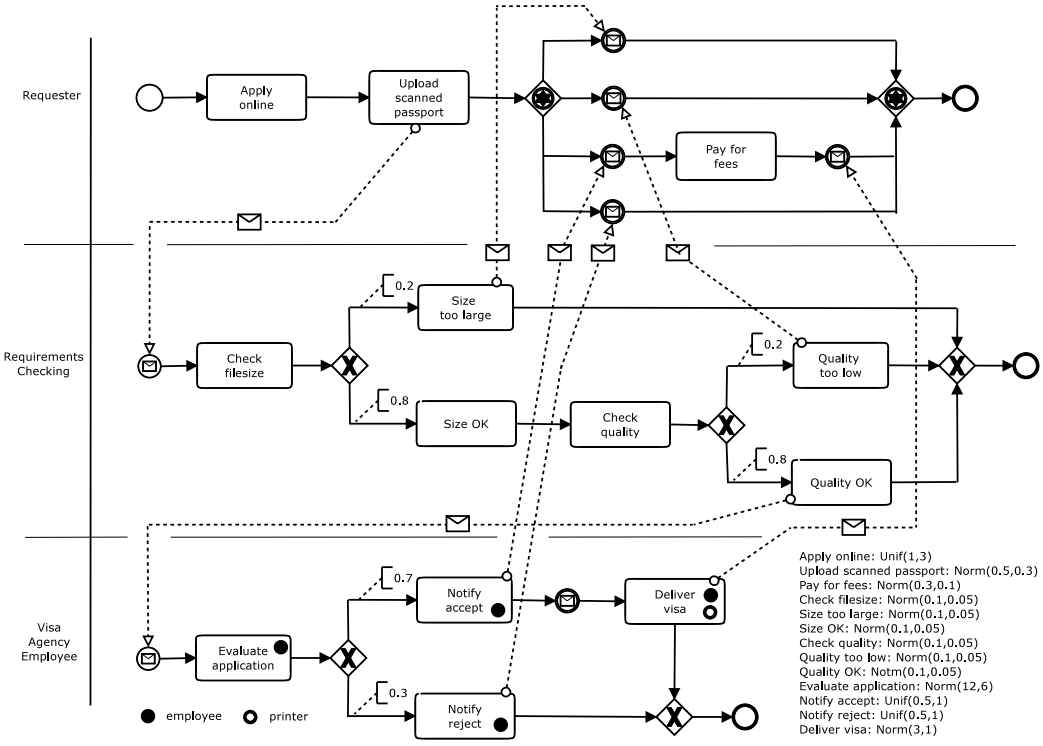

Figure 12: BPMN process of the visa example

a process, the expected number of activity instances when completing a process under the assumption of sufficient resources, and the expected activity response time.

The problem of understanding how bounded resources can impact the behavior of a process is approached by Netjes *et al.* by using colored Petri nets [24]. They introduce the notion of "flexible resource allocation" as a way to assign resources associated to a given role based on priorities. Havur *et al.* study the problem of resource allocation in business processes management systems where resources can be assigned constraints (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. More recently, Sperl *et al.* [33] describe a stochastic method for quantifying resource utilization relative to structural properties of the processes and historical executions.

The work presented in this paper can encompass most of the specification and analysis features in [32, 19, 25, 5, 33] for BPMN. The rewriting logic specification

presented here can be used to specify and verify resource availability and workload, several related measures, and the impact of bounded resources on a process behavior. Despite the fact that the main focus of this paper has been on resource allocation analysis and its optimization, the underlying rewriting logic semantics can simultaneously be a testbed for integrating and combining several types of mechanical formal analysis techniques such as algorithmic-based LTL model checking and statistical verification. Features such as multiple roles associated to an activity, flexible resource allocation, dependencies among the resources, and the use of historical data can be extensions of the present work.

There is also significant effort in the research community aimed at providing formal semantics and verification techniques for business processes using different formalisms, with Petri nets and process algebra as the preferred formalisms by most of them (see, e.g., [6, 10, 31]). [6] presents a business process monitoring platform that relies on the ProM mining tool and Petri nets tools for verifying synchronization, sojourn and waiting times. Raedts *et al.* [31] translate BPMN processes into Petri nets and use model checkers to analyze invariants such as deadlock freedom. The authors of [35] present a formal semantics for BPMN by an encoding into the process algebra CSP. They show in [36] how such a semantics can be used to verify compatibility between business participants in a collaboration. There is a further extension in [37] to propose a timed semantics of BPMN with delays. In [7, 23, 21], the focus is on the semantics formalized in [35, 37] by proposing an automated transformation from BPMN to timed CSP, as well as composition verification techniques for checking properties using the FDR2 model checker. Poizat *et al.* [30] present an encoding of an untimed subset of BPMN into the LNT process algebra for supporting the analysis of process evolution. [14] presents an early attempt to represent BPMN processes enriched with time features in Maude. This work was extended in [13] to provide automated verification of stochastic properties such as expected processing and synchronization times. Compared to these related works, the executable specification in rewriting logic presented in Section 3 can be seen as a semantic framework for BPMN, yet it is not the primary goal of this paper. The main difference with respect to those works is that the focus here is in the formal specification and verification of quantitative aspects of processes and their resources.

Several works propose extensions of BPMN with time constructs, see, e.g. [3, 16]. In [16], the authors present Time-BPMN, an extension of BPMN to represent various constraints and dependencies that may arise when modelling business processes. The temporal constraints are used to control the beginning/end of an activity, whereas temporal dependencies involve two activities and indicate the

relationship between their respective beginning/end. In [3], a metamodel-based approach to integrate temporal constraints and dependencies is introduced. The time aspects are specified using rules and OCL constraints capture the semantics of these rules. [16] introduces time as a first class citizen to the BPMN standard whereas [3] proposes an MDE-based approach that enrich the existing BPMN control-flow graph model. In comparison, the goal in the present paper was not to extend the BPMN notation to take expressive modeling of time constraints into account, but to rely on usual time representation (duration of task and flow) and concentrate on the analysis of key timing properties in these models.

The expected processing time, the expected synchronization time for merge gateways, and other similar measures have been considered by some authors to evaluate deployed processes by, e.g., collecting timestamps from logs (see, e.g., [1, 2, 6]). Task duration has been specified using literal values in [5, 6, 26] and probabilities have been used to specify the likelihood of branching in [5, 26]. Inclusive gateways are only supported in [5, 20, 30] and loops only in [15, 26, 29]. The goal in most of these works is, however, quite different. Overall, Oliveira *et al.* [26] possibly have the closest approach to the one presented in this paper: task durations and probabilities for exclusive branching can be specified, and loops and continuous time are considered. However, they cannot specify durations with stochastic expressions and do not perform analysis on shared resources.

6. Conclusion

This paper presented a solution to the problem of formal modeling and mechanical analysis of resource allocation for BPMN processes. The approach relies on the rewriting logic semantic framework and uses the Maude system (and language) to stochastically simulate multiple concurrent executions of a process instance that compete for the shared resources. The encoding of the BPMN syntax and execution semantics in rewriting logic supports an expressive subset of BPMN consisting of: activity and collaboration diagrams, several types of gateways, timed flows and tasks, probabilities for exclusive and inclusive split gateways, unbalanced workflows, and looping behavior. A main contribution of this work is to demonstrate how several properties of interest on resource allocation can be automatically analyzed, including resource charge over time and usage percentage for each resource replica. This paper also showed how the optimal allocation of resources can be computed by formulating a multi-objective optimization problem. The approach was validated on several examples, including real-world processes with large workloads.

As far as future work is concerned, a first perspective is to move from design time to runtime. The idea would be to dynamically adjust the amount of resources depending on the evolution of the workload. It is worth noting that this runtime solution would apply in some specific cases only when resources are available and could be used on-demand. A second perspective aims at using the results of the analysis proposed in this paper to address refactoring issues of a process in order to make it more efficient. As an example, the analysis may emphasize the unnecessary use of certain gateways resulting in bottlenecks in the execution. Ideally, the ultimate goal would be to have such a refactoring process as automated as possible for simplifying this optimization task.

Acknowledgments. The first author was supported in part by projects PGC2018-094905-B-I00 (Spanish MINECO/FEDER) and UMA18-FEDERJA-180 (FEDER Andalucía). The second author has been supported in part by CAPES, Colciencias, and INRIA via the STIC AmSud project "EPIC: EPistemic Interactive Concurrency" (Proc. No 88881.117603/2016-01), and Capital Semilla 2017 project "SCORES: Stochastic Concurrency in Rewrite-based Probabilistic Models" (Proj. No. 020100610).

References

[1] van der Aalst, W.M.P., 2016. Process Mining - Data Science in Action, Second Edition. Springer.

[2] van der Aalst, W.M.P., van Dongen, B.F., 2002. Discovering workflow performance models from timed logs, in: Han, Y., Tai, S., Wikarski, D. (Eds.), Engineering and Deployment of Cooperative Information Systems, First International Conference, EDCIS 2002, Beijing, China, September 17-20, 2002, Proceedings, Springer. pp. 45–63.

[3] Arévalo, C., Cuaresma, M.J.E., Ramos, I.M., Domínguez-Muñoz, M., 2016. A metamodel to integrate business processes time perspective in BPMN 2.0. Information & Software Technology 77, 17–33.

[4] Bouhoula, A., Jouannaud, J., Meseguer, J., 2000. Specification and proof in membership equational logic. Theoretical Computer Science 236, 35–132.

[5] Braghetto, K.R., Ferreira, J.E., Vincent, J., 2011. Performance evaluation of business processes through a formal transformation to SAN, in: Thomas, N.

(Ed.), Computer Performance Engineering - 8th European Performance Engineering Workshop, EPEW 2011, Borrowdale, UK, October 12-13, 2011. Proceedings, Springer. pp. 42–56.

[6] Bruni, R., Corradini, A., Ferrari, G.L., Flagella, T., Guanciale, R., Spagnolo, G., 2012. Applying process analysis to the italian egovernment enterprise architecture, in: Carbone, M., Petit, J. (Eds.), Web Services and Formal Methods - 8th International Workshop, WS-FM 2011, Clermont-Ferrand, France, September 1-2, 2011, Revised Selected Papers, Springer. pp. 111–127.

[7] Capel, M.I., Mendoza, L.E., 2012. Automating the transformation from BPMN models to CSP+T specifications, in: Bowen, J.P., Zhu, H., Hinchey, M. (Eds.), 35th Annual IEEE Software Engineering Workshop, SEW 2012, Heraclion, Crete, Greece, October 12-13, 2012, IEEE Computer Society. pp. 100–109.

[8] Christiansen, D.R., Carbone, M., Hildebrandt, T.T., 2011. Formal semantics and implementation of BPMN 2.0 inclusive gateways, in: Bravetti, M., Bultan, T. (Eds.), Web Services and Formal Methods - 7th International Workshop, WS-FM 2010, Hoboken, NJ, USA, September 16-17, 2010. Revised Selected Papers, Springer. pp. 146–160.

[9] Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.L. (Eds.), 2007. All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic. volume 4350 of *Lecture Notes in Computer Science*. Springer.

[10] Dijkman, R.M., Dumas, M., Ouyang, C., 2008. Semantics and analysis of business process models in BPMN. Information & Software Technology 50, 1281–1294.

[11] Durán, F., Rocha, C., Salaün, G., 2018a. A Note on Resource Allocation Analysis of BPMN Processes. URL: `http://maude.lcc.uma.es/BPMN-R`.

[12] Durán, F., Rocha, C., Salaün, G., 2018b. Computing the parallelism degree of timed BPMN processes, in: Mazzara, M., Ober, I., Salaün, G. (Eds.), Software Technologies: Applications and Foundations - STAF 2018 Collocated

Workshops, Toulouse, France, June 25-29, 2018, Revised Selected Papers, Springer. pp. 320–335.

[13] Durán, F., Rocha, C., Salaün, G., 2018c. Stochastic analysis of BPMN with time in rewriting logic. Sci. Comput. Program. 168, 1–17.

[14] Durán, F., Salaün, G., 2017. Verifying timed BPMN processes using maude, in: Jacquet, J., Massink, M. (Eds.), Coordination Models and Languages - 19th IFIP WG 6.1 International Conference, COORDINATION 2017, Held as Part of the 12th International Federated Conference on Distributed Computing Techniques, DisCoTec 2017, Neuchâtel, Switzerland, June 19-22, 2017, Proceedings, Springer. pp. 219–236.

[15] El-Saber, N.A.S., Boronat, A., 2014. BPMN formalization and verification using maude, in: Proceedings of the 2014 Workshop on Behaviour Modelling - Foundations and Applications, BM-FA 2014, York, United Kingdom, July 22-22, 2014, ACM. pp. 1–8.

[16] Gagné, D., Trudel, A., 2009. Time-bpmn, in: Hofreiter, B., Werthner, H. (Eds.), 2009 IEEE Conference on Commerce and Enterprise Computing, CEC 2009, Vienna, Austria, July 20-23, 2009, IEEE Computer Society. pp. 361–367.

[17] ISO/IEC, 2013. International Standard 19510, Information technology – Business Process Model and Notation.

[18] Krishna, A., Poizat, P., Salaün, G., 2017. VBPMN: automated verification of BPMN processes (tool paper), in: Polikarpova, N., Schneider, S. (Eds.), Integrated Formal Methods - 13th International Conference, IFM 2017, Turin, Italy, September 20-22, 2017, Proceedings, Springer. pp. 323–331.

[19] Li, J., Fan, Y., Zhou, M., 2004. Performance Modeling and Analysis of Workflow. IEEE Transactions on Systems, Man, and Cybernetics 34, 229–242.

[20] Mateescu, R., Salaün, G., Ye, L., 2014. Quantifying the parallelism in BPMN processes using model checking, in: Seinturier, L., de Almeida, E.S., Carlson, J. (Eds.), CBSE'14, Proceedings of the 17th International ACM SIGSOFT Symposium on Component-Based Software Engineering (part of CompArch 2014), Marcq-en-Baroeul, Lille, France, June 30 - July 4, 2014, ACM. pp. 159–168.

[21] Mendoza, L.E., Capel, M.I., Pérez, M.A., 2012. Conceptual framework for business processes compositional verification. Information & Software Technology 54, 149–161.

[22] Meseguer, J., 1992. Conditional Rewriting Logic as a Unified Model of Concurrency. Theoretical Computer Science 96, 73–155.

[23] Morales, L.E.M., Tuñón, M.I.C., Pérez, M.A., 2011. A formalization proposal of timed BPMN for compositional verification of business processes, in: Filipe, J., Cordeiro, J. (Eds.), Enterprise Information Systems - 12th International Conference, ICEIS 2010, Funchal, Madeira, Portugal, June 8-12, 2010, Revised Selected Papers, Springer. pp. 388–403.

[24] Netjes, M., van der Aalst, W.M., Reijers, H.A., 2005. Analysis of Resource-Constrained Processes with Colored Petri Nets, in: Jensen, K. (Ed.), Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark, October 24-26, 2005, pp. 251–266.

[25] Oliveira, C., Lima, R., Reijers, H., Ribeiro, J., 2012. Quantitative Analysis of Resource-Constrained Business Processes. Trans. on Syst., Man, and Cybern. 42, 669–684.

[26] Oliveira, C.A.L., Lima, R.M.F., Andre, T., Reijers, H.A., 2009. Modeling and analyzing resource-constrained business processes, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11-14 October 2009, IEEE. pp. 2824–2830.

[27] Ölveczky, P.C., Meseguer, J., 2007. Semantics and pragmatics of real-time maude. Higher-Order and Symbolic Computation 20, 161–196.

[28] OMG, 2011. Business process model and notation (BPMN) - v 2.0.

[29] Poizat, P., Salaün, G., 2012. Checking the realizability of BPMN 2.0 choreographies, in: Ossowski, S., Lecca, P. (Eds.), Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012, ACM. pp. 1927–1934.

[30] Poizat, P., Salaün, G., Krishna, A., 2017. Checking business process evolution, in: Kouchnarenko, O., Khosravi, R. (Eds.), Formal Aspects of Component Software - 13th International Conference, FACS 2016, Besançon, France, October 19-21, 2016, Revised Selected Papers, pp. 36–53.

[31] Raedts, I., Petkovic, M., Usenko, Y.S., van der Werf, J.M.E.M., Groote, J.F., Somers, L.J., 2007. Transformation of BPMN models for behaviour analysis, in: Augusto, J.C., Barjis, J., Ultes-Nitsche, U. (Eds.), Modelling, Simulation, Verification and Validation of Enterprise Information Systems, Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2007, In conjunction with ICEIS 2007, Funchal, Madeira, Portugal, June 2007, INSTICC PRESS. pp. 126–137.

[32] Schömig, A.K., Rau, H., 1995. A Petri Net Approach for the Performance Analysis of Business Processes. Technical Report 116. Universität Würzburg. Würzburg, Germany.

[33] Sperl, S., Havur, G., Steyskal, S., Cabanillas, C., Polleres, A., Haselböck, A., 2017. Resource utilization prediction in decision-intensive business processes, in: Ceravolo, P., van Keulen, M., Stoffel, K. (Eds.), Proceedings of the 7th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2017), Neuchâtel, Switzerland, December 6-8, 2017., CEUR-WS.org. pp. 128–141. URL: http://ceur-ws.org/Vol-2016/paper10.pdf.

[34] Walck, C., 2007. Hand-Book on Statistical Distributions for Experimentalists. Technical Report SUF–PFY/96–01. Universitet Stockholms. Stockholm.

[35] Wong, P.Y.H., Gibbons, J., 2008a. A process semantics for BPMN, in: Liu, S., Maibaum, T.S.E., Araki, K. (Eds.), Formal Methods and Software Engineering, 10th International Conference on Formal Engineering Methods, ICFEM 2008, Kitakyushu-City, Japan, October 27-31, 2008. Proceedings, Springer. pp. 355–374.

[36] Wong, P.Y.H., Gibbons, J., 2008b. Verifying business process compatibility (short paper), in: Zhu, H. (Ed.), Proceedings of the Eighth International Conference on Quality Software, QSIC 2008, 12-13 August 2008, Oxford, UK, IEEE Computer Society. pp. 126–131.

[37] Wong, P.Y.H., Gibbons, J., 2009. A relative timed semantics for BPMN. Electronic Notes in Theoretical Computer Science 229, 59–75.