From Static to Dynamic Analysis and Allocation of Resources for BPMN Processes

Francisco Durán[1], Yliès Falcone[2], Camilo Rocha[3], Gwen Salaün[2(✉)], and Ahang Zuo[2]

[1] ITIS Software, University of Málaga, Málaga, Spain
[2] Univ. Grenoble Alpes, CNRS, Grenoble INP, Inria, LIG, 38000 Grenoble, France
gwen.salaun@inria.fr
[3] Pontificia Universidad Javeriana, Cali, Colombia

Abstract. Business process optimisation is a strategic activity in organisations because of its potential to increase profit margins and reduce operational costs. One of the main challenges in this context is concerned with the problem of optimising the allocation and sharing of resources. In this work, processes are described using the BPMN notation extended with an explicit description of execution time and resources associated with tasks, and can be concurrently executed multiple times. First, a simulation-based approach for computing certain metrics of interest, such as average execution time or resource usage, is presented. This approach applies off-line and is static in the sense that the number of resources does not evolve over the time of the simulation. In a second step, an alternative approach is presented, which works online, thus requiring the instrumentation of an existing platform for retrieving information of interest during the processes' execution. This second approach is dynamic because the number of resource replicas is updated over the time of the execution. This paper aims at stressing pros and cons of both approaches, and at showing how they complement each other.

1 Introduction

Business process optimisation is a strategic activity in organisations because of its potential to increase profit margins and reduce operational costs. Optimisation is, however, a difficult task to be achieved manually since several parameters should be taken into account (e.g., execution times, resources, costs, etc.). One of the main challenges in this context is concerned with the problem of optimising the allocation and sharing of resources. Resource usage is crucial because it directly impacts the time it takes to execute a process. Moreover, by associating a certain cost to each resource, the total cost of executing a process a certain number of times can be computed. Optimising resource usage reduces the process execution time and the costs associated with its execution.

In this work, we assume that a description of a business process is given using the BPMN [24] workflow-based modelling language. BPMN has been standardised by the International Organization for Standardization (ISO). It was first

K. Bae (Ed.): WRLA 2022, LNCS 13252, pp. 3–21, 2022.
https://doi.org/10.1007/978-3-031-12441-9_1

published in 2013 and since then it has become the *de facto* notation for developing business processes. The BPMN language defines the set of tasks involved in a process and the order in which they should be executed. Beyond this description of the model, the time it takes to execute each task is also needed, as well as an explicit description of the resources required for executing each task. This extended model is precise enough for modelling both behavioural and quantitative aspects of processes.

This paper presents two ways to analyse BPMN processes with time and resources. Both techniques assume that the process is executed multiple times and multiple concurrent executions of a process compete for the shared resources. Such multiple executions correspond to realistic scenarios where a process is not executed once but several times (one execution per client or user for instance). Furthermore, both approaches also aim at computing some metrics of interest such as average execution times, resource usage, and costs. The first approach applies to design models, without the need for an implementation of the system running on real resources. To compute the previously mentioned metrics, offline simulation techniques are used, assuming that the allocation of resources is static (i.e., no update of the number of resources during the simulation). This first approach relies on a specification of a subset of BPMN in rewriting logic [21]. This specification is executable in Maude [6], and the computation of metrics is achieved by using Maude's rewriting tools.

The second approach applies at runtime or online, and works by instrumenting an existing platform for executable BPMN (Activiti [2] in this work). In this case, access to a database is used for storing information related to the execution of the process. This information is particularly useful for computing the process execution time, resource usage, and costs. This approach is also dynamic in the sense that the number of replicas for each resource is not defined once and for all, but can be updated by using the metrics computed during the process execution. In particular, a strategy that relies on the resource usage values for dynamically updating the number of replicas of each resource is presented.

The static approach applies to a model of the process, and additional information is required such as the probability to execute exclusive branches. This approach is useful for processes under development, or for potential changes that need to be evaluated before being applied. The approach can help for instance to simulate several scenarios and decide whether the number of required resources needs to be adjusted before the deployment of the process in production. On the other hand, the dynamic approach accepts as input an executable BPMN process and provides strategies to update resources at execution time thus allowing a certain stabilisation of the computed metrics over time (such as execution times and resource usage). However, this dynamic change does not apply in all contexts since it is not systematically possible to dynamically update the number of any kind of resources (such as human beings).

The organisation of the rest of this paper is as follows. Section 2 introduces the BPMN notation used in this work. Section 3 overviews the static approach for analysing resource usage. Section 4 surveys the main ideas of the dynamic approach for the allocation of resources. Section 5 presents existing works on this topic. Section 6 concludes by comparing both approaches.

2 BPMN with Time and Resources

BPMN 2.0 (BPMN, as a shorthand, in the rest of this paper) was published as an ISO/IEC standard [17] in 2013 and is nowadays extensively used for modelling and developing business processes. In this paper, for the sake of simplicity, we focus on activity diagrams including the BPMN constructs related to control-flow modelling and behavioural aspects. Beyond those constructs, execution time and resources are also associated with tasks, and probabilities are specified for exclusive and inclusive split gateways. Figure 1 summarises some of the BPMN constructs used in this work, with a focus on how time and resources are associated with flows and tasks.

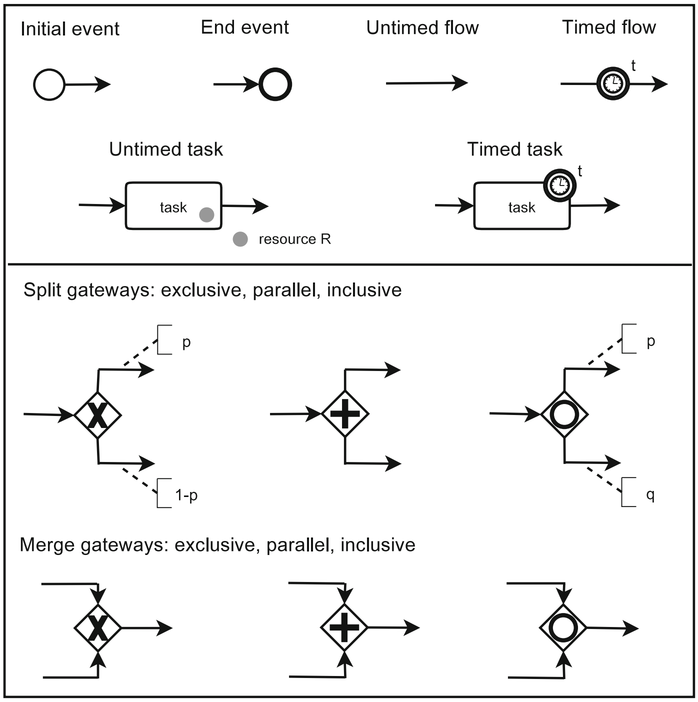
Initial event
End event
Untimed flow
Timed flow
t
Untimed task
task
resource R
Timed task
task
t
Split gateways: exclusive, parallel, inclusive
p
1-p
p
q
Merge gateways: exclusive, parallel, inclusive

Fig. 1. Extended BPMN Syntax

Specifically, the node types *event*, *task*, and *gateway*, and the edge type *sequence flow* are considered. Start and end events are used, respectively, to initialise and terminate processes. A task represents an atomic activity that has exactly one incoming and one outgoing flow. A sequence flow describes two nodes executed one after the other in a specific execution order. A task and a flow may have a duration or delay. The timing information associated with tasks and flows is described as a literal value (a non-negative real number, possibly 0). Resources are explicitly defined at the task level. A task that requires resources can include, as part of its specification, the name of the required resources. Thus, a task is specified with an amount of time (its duration), and information on its required resources. Then, once the resources required by a task are acquired, the task is going to execute for the defined duration.

Gateways are used to control the divergence and convergence of the execution flow. Three types of gateways are considered for the static analysis: *exclusive*, *inclusive*, and *parallel*. 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 synchronises concurrent flows for all its incoming branches, and creates concurrent flows for all its outgoing branches.

In the static approach, data-based conditions for split gateways are modelled using probabilities associated with 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. We will see that only the static approach presented in Sect. 3 does need such probabilities, whereas the dynamic approach presented in Sect. 4 requires an executable BPMN process as input (with real data-based conditions). Processes with looping behavior are supported, as well as unbalanced workflows.

Running Example. For illustration purposes, we present a simple example of a process describing how clients can deliver goods via an external service (a mail office for instance). This process is described in Fig. 2. First of all, an employee collects the goods brought by a client. Then, in parallel, the client pays for the delivery service and an employee prepares a parcel. The company can deliver the parcel using a car or using a drone (depending on the distance for example). Beyond the required resources appearing in the figure, we can also see times (expressed as durations) associated with tasks. As an example, the average duration for preparing a parcel is 5 units of time (e.g., 5 min). We also assume that the probability of delivering by car or by drone is the same (0.5).

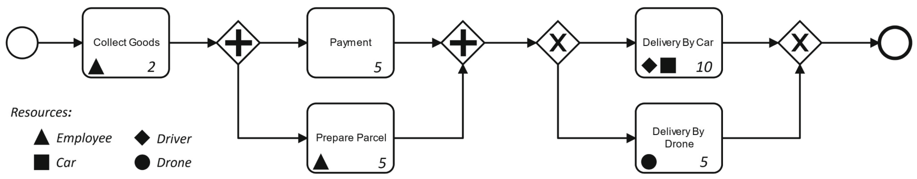

Fig. 2. Goods Delivery Process

3 Static Quantitative Analysis

In this section, we summarise the approach presented in [10] for analysing BPMN processes with resources. More precisely, we first introduce the specification of BPMN (syntax and semantics) in Maude's rewriting logic. Second, we present the quantitative properties of interest focusing on timing and resource-based properties. To compute these properties, we leverage Maude's rewriting capabilities to simulate and extract analysis results on a given BPMN process.

3.1 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 or a value modelling its duration (0 if there is no duration), and a set of resources required for its execution. A split node includes a node identifier, a gateway type (exclusive, inclusive, or parallel), 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 corresponding to the probability of executing that flow (1 by default).

3.2 Execution Semantics

The operational semantics of BPMN is defined using a rewrite theory, with rewrite rules modeling how tokens evolve through a process. This rewrite theory is executable, which allows us to simulate BPMN processes. In this specification, each 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 Sect. 3.1 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), and a set of resources. It also keeps track of the quantities being measured during the analysis of a process. Figure 3 presents the structure of the Simulation object.

Tokens. Tokens are used to represent the evolution of the workflow under execution. When a process instance is triggered, a token is added to the start node. The tokens move through nodes and flows of the process. When a token meets a split gateway (e.g., parallel gateway), several tokens are generated on outgoing flows, depending on the type of split gateway. On the contrary, when multiple tokens meet a merge gateway (e.g., inclusive gateway), they are merged into a

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

Fig. 3. Representation of the Simulation Object

single token depending on the type of merge gateway. A token is represented as a term token(TId, Id, T). Since several executions may happen simultaneously, each execution has a unique identifier, and 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, modelling 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 of this queue 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.

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 during which any replica of this resource was used. These two last parameters are stored during the simulation, and are particularly useful for analysis purposes. When a task requires several resources, it atomically uses all of them at once, or waits for them to become available.

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 time. The inter-arrival time is specified as a stochastic expression.

Rewrite Rules for BPMN Constructs. Rewriting rules represent how tokens evolve through the process and how nodes are executed, thus defining the execution semantics of BPMN. Each action supported by the system is modelled as a rewrite rule. These rules are overviewed in the rest of this section to gather an intuition on the formal semantics (see [8] for the complete specification).

Start/End Events. Figure 4 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

```
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) .
```

Fig. 4. Start Event Processing

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 initialisation 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 modelled with two rules. The first rule, the initTask rule shown in Fig. 5, represents the task initiation, which is applied when a token with zero time is available for the incoming flow (line 5). If all the resources 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 Sect. 3.3.

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, a new one is generated for the outgoing flow, and all resources are released.

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 with 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.

```
rl [initTask] :
   < PId : Process |
      nodes : (task(NId, TaskName, FId1, FId2, SE, RIds, SEI), Nodes), Atts >
   < SId : Simulation |
      tokens : (token(TId, FId1, 0) Tks),
      task-tstamps : TTSs, gtime : T, resources : Rs, Atts1 >
   < CId : Counter | counter : N >
=> if allResourcesAvailable(RIds, Rs)
   then < PId : Process |
            nodes : (task(NId, TaskName, FId1, FId2, SE, RIds, SEI), Nodes), Atts >
         < SId : Simulation |
            tokens : insert(Tks, token(TId, NId, time(eval(SE, N)))),
            task-tstamps : if TTSs[TId][NId] == undefined
                           then insert(TId, insert(NId, T, TTSs[TId]), TTSs)
                           else TTSs
                           fi,          ---- for loops, stamps get overwritten
            gtime : T,
            resources : grabResources&updateTime(RIds, Rs, time(eval(SE, N)), T), Atts1 >
         < CId : Counter | counter : int(eval(SE, N)) >
    else ...                            ---- if necessary, the scheduler is updated
    fi .
```

Fig. 5. Task Initiation Rule

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, synchronisation 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 [5], 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 backward 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 synchronisation times.

Loops and Unbalanced Workflows. The modelling 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 behaviour 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.

3.3 Properties

Several kinds of properties or metrics can be computed, particularly 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 synchronisation time (AST) for merge gateways, representing the time that elapse from the arrival of the first token through one of its incoming flows to its activation. Synchronisation times make sense only for parallel and BPMN 1.0 inclusive gateways, since there is no waiting nor synchronisation 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 of 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^1_R 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^1_A is 28 and GTU^1_B is 20.
– The average usage percentage UP_R for a resource R over the global execution time. E.g., continuing with the running example, on average, an instance of the resource A is used 24% of the global execution time when executing 200 instances of a process P.

To compute these metrics, Maude rewriting capabilities are used to simulate and extract analysis results on a given BPMN process. The simulation object presented in Sect. 3.2 is used to accumulate information of synchronisation times, task durations, and resource usages. At the end of all executions, these results are used for computing the expected average times and resource usage percentages. Since the analysed 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. Indeed, 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).

Last but not least, if one can associate a cost (in euros for example) to each kind of resource, we can compute the total cost of the simulation by using the collected data on execution times and resource usage. We can even go farther by computing the optimal allocation of resources. This is achieved by expressing this computation as a multi-objective optimisation problem since we may not want to reduce costs but also to reduce execution time for example. The solution to this optimisation problem is computed by using heuristic-based search algorithms such as gradient descent [26].

3.4 Example

Let us illustrate this approach with the running example presented in Sect. 2. Table 1 shows a few experiments consisting of 1,000 tokens with an inter-arrival time computed with an exponential probability distribution with 2 as the parameter. For each row, there is a variation in the input in terms of the number of replicas of the different resources. As a result, the table gives the total execution time for executing 1,000 times the process, the average execution time, and the total cost (assuming a cost per hour of 40, 30, 35, and 25 euros for each resource, respectively).

Table 1. Experimental Results for the Delivery Process

Resources								Total execution time	Average execution time	Total cost
Employee		Car		Driver		Drone				
Inst.	Usage %	Inst.	Usage %	Inst	Usage %	Inst.	Usage %			
1	99.11	1	72.63	1	72.63	1	34.48	7 063.00	2 794.26	918 190.00
2	99.18	2	69.85	2	69.85	2	35.92	3 528.92	904.02	917 519.74
3	81.42	2	90.37	2	90.37	1	84.09	2 865.94	463.75	788 133.61
4	85.24	3	84.42	3	84.42	2	58.45	2 053.10	131.00	831 508.77
4	86.75	4	57.88	4	57.88	4	33.04	2 017.26	100.40	1 048 976.52

First of all, we can observe a clear correlation between the number of resources and the execution time/costs. The more resources, the shorter it takes to execute once the process (or all processes), but the more resources, the higher cost. Secondly, we can see that the critical resource is the employee since whatever is the number of replicas, this resource is always very busy (active more than 80% of his time). In contrast, drones are less busy except if there is a single drone and several replicas for the other resources. Finally, if we assume that we both want to reduce the average execution time and the total cost with an equal weight (0.5 and 0.5), the optimal resource allocation is 4, 3, 3, and 2 (before last row in Table 1).

4 Dynamic Quantitative Analysis

In this section, we will show how an existing platform (Activiti [2] in this work) can be instrumented to extract the required information from its database and compute properties periodically during the process execution. We will also show how we can develop dynamic resource allocation strategies for varying the number of resource replicas at runtime and thus impact the results of these properties. Note that in this section, we do not have any restrictions on the BPMN syntax, we just need BPMN processes to be executable. Moreover, there is no need to have probabilities associated to split exclusive and inclusive gateways, since we have real data-based conditions.

4.1 Instrumentation

In this section, we use Activiti as BPMN platform. Activiti is an open-source workflow engine written in Java that can execute business processes described in BPMN 2.0. We first require monitoring techniques [4,13] for BPMN processes at runtime. These techniques are useful because a process is usually not executed only once. Instead, a process can be executed multiple times. Each execution of the process is called an instance. An instance of the process can be in one of the following states: *initial* means that the instance is ready to start (one token in the start event), *running* means that the instance is currently executing and is not yet completed, *completed* means that all tokens have reached end events. Tokens are used to define the behaviour of a process. Similarly to the static approach, an identifier is used to characterise a specific instance of process execution, and this identifier is thus associated to all nodes (e.g., tasks) executed by this instance.

Monitoring techniques for BPMN executed using Activiti mostly aim at analysing the information stored in a database, and extracting the information required for computing the properties of interest (such as AET and resource usage percentage). Figure 6 gives an overview of this data extraction. We first need to retrieve the information regarding task execution and completion. This is what we can see in Fig. 6 (top right, (a)). For each task, we also extract the corresponding process execution instance and the times of beginning and end. This information is useful for determining which resources were in use and for what amount of time. Second, we retrieve execution traces for each process instance as shown in Fig. 6 (bottom right, (b)). An execution trace corresponds to a list of tasks executed by this specific instance. The tasks are not stored with a specific order in the database. Therefore, we have to order these tasks by using time stamps, corresponding to the time at which each task is executed. These time stamps are computed by the process execution engine, which relies on a global clock. The execution trace corresponding to a specific instance can be computed only when the instance is in its *completed* state.

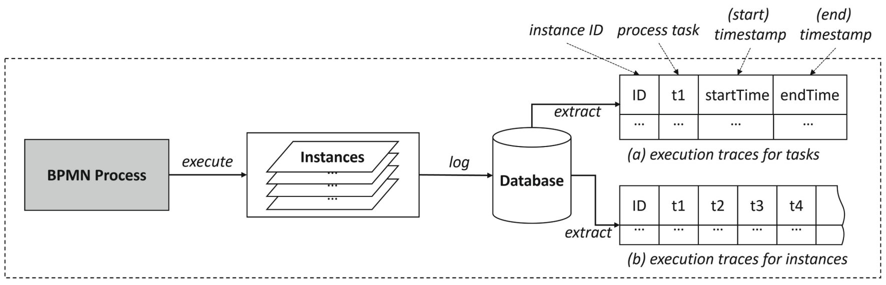

Fig. 6. Runtime Monitoring of Multiple Executions of a BPMN Process

4.2 Computation of Properties

Since new instances can execute at any time and possibly infinitely, the dynamic approach requires extracting data and computing properties on time and resources periodically. There are several possible strategies to choose the period. It can be based on a fixed amount of time (e.g., every 10 min) or it can apply when a certain number of process instances have been completed. These two strategies can also be combined, e.g., we get data whenever 100 instances have been completed or every hour if after one hour less than 100 instances have been completed. The choice of one of these strategies may have a different impact on the actual results. Note that the choice of this strategy is a parameter of the approach. In the rest of this section, we rely on a time-based strategy.

When the period completes, the data extraction is triggered. Then, we extract the required information from these data to compute the properties presented in Sect. 3.3 on execution times and resource usage. As an example, to compute the resource usage percentage per resource replica, we analyse the tasks executed during the last period of time. For these tasks, we look at the resources associated with each task and sum up the durations each resource was active during that period. Then, we divide this total time by the number of replica and compute a percentage out of these numbers by using the time of activity for each replica out of the time of the period.

As we will see below, the results are represented using curves that show the different property values (e.g., average execution time) along time.

4.3 Dynamic Resource Allocation

Several strategies can be defined for dynamically changing the number of replicas for each resource. These strategies rely on the metrics computed before and thus can vary in their choice and implementation. For instance, one strategy can aim at reducing the average execution time whereas another one may maintain the resource usage under a certain level, e.g., under 90%. We could also implement strategies that take several criteria into account at the same time, e.g., reduce process execution time while maintaining resource usage below a threshold. Another parameter of the strategy is when to apply this change. A simple solution is to apply it when we compute new values of the aforementioned properties. The strategy can rely on this fresh information to decide to change the number of resource replicas. However, we could decide to apply changes more or less often to avoid the classic oscillation problem (add one, remove one, add one, remove one, etc.). As an example, one can decide to change the number of replicas every three periods of time, every day, or when a certain number of process instances complete (e.g., 100).

For illustration purposes, we will present an example of strategy in the rest of this section. This strategy focuses on one specific property, namely the percentage of resource usage per replica. The strategy aims at maintaining this percentage within a certain interval, for instance, $[70\%, 90\%]$. After completion of a period of time, all properties are computed and the strategy then checks if the usage percentage for each resource is still included in this interval. If, for a given resource, this percentage goes above the highest value (e.g., 90% in our example), one replica of that resource is added. If this percentage goes below the lowest value (e.g., 70% in our example), one replica of that resource is removed. Note that we choose in this strategy to follow the same period of time as the one used for the computation of properties.

4.4 Example

Let us focus again on the goods delivery example introduced in Sect. 2. The only difference in terms of the input BPMN process is that here we do not need to make explicit the probabilities of executing the split exclusive gateway. This decision is taken based on internal data belonging to the (executable) BPMN process. We use the same workload as in Sect. 3.4, that is, 1000 tokens with exp(2) as inter-arrival time. There are additional parameters that are required for the dynamic approach. We use as initial allocation of resource one replica for each resource type. The targeted interval for resource usage is $[70\%, 90\%]$. The period for updating the metrics is fixed to 10 units of time whereas the strategy for dynamic resource update applies every 60 units of time.

In the rest of this section, we will show three different figures to give different insights on the results of the multiple process execution. Figure 7 describes the evolution of the number of replicas for each kind of resource. The employee is particularly important because every execution of the process requires an employee to collect goods and prepare parcels whereas the other resources are not systematically used for every process execution. One can see that this execution requires 2 or 3 employees to work properly. Cars and drivers take more time than drones to deliver goods (10 units of times for cars and 5 units for drones), therefore more replicas are required for allowing the delivery by car with driver.

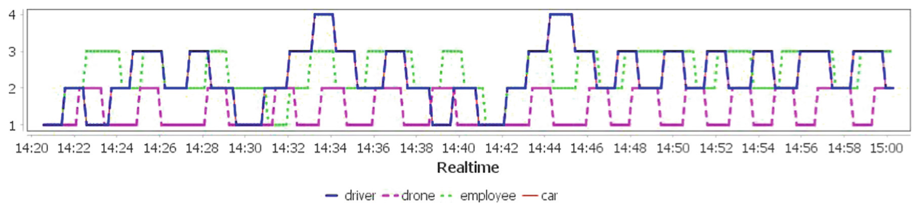

Fig. 7. Goods Delivery Process: Evolution of the Number of Replicas

Figure 8 focuses on the usage percentage per replica for each type of resource. It is worth reminding that the strategy used for these experiments aim at maintaining the percentage in the interval $[70\%, 90\%]$. We can see that from the beginning the usage percentage for employees is higher than 90% thus explaining why several replicas of employees were added at the beginning in Fig. 7. After the addition of these replicas for employee, the percentage remains lower.

The usage percentage for drones is the lower of all resources. We can observe important variations in all these percentages because we use a short period for computing these numbers (10 units of time) and because the use of an exclusive gateway for the delivery induces variations between the use of drones or cars.

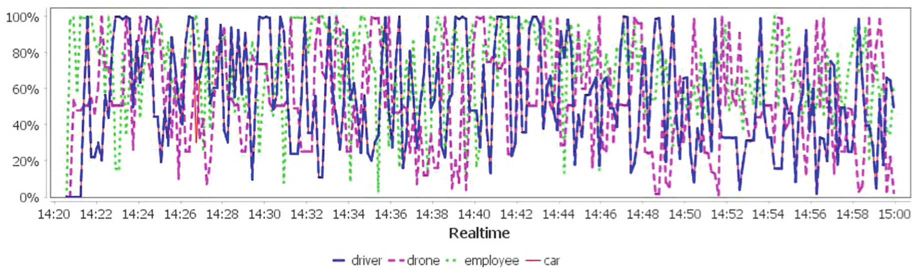

Fig. 8. Goods Delivery Process: Resource Usage

Figure 9 shows the evolution of the average execution time. The curve shows that this time tends to increase at the beginning, but at some point stabilises (since new executions occur on a periodic basis) and remains around 40 units of time. We can see peaks at some points of the execution corresponding to an increase in the number of delivery by car, which takes more time than drones. This increase in time can be correlated with the addition in Fig. 7 of additional replicas of cars and drivers.

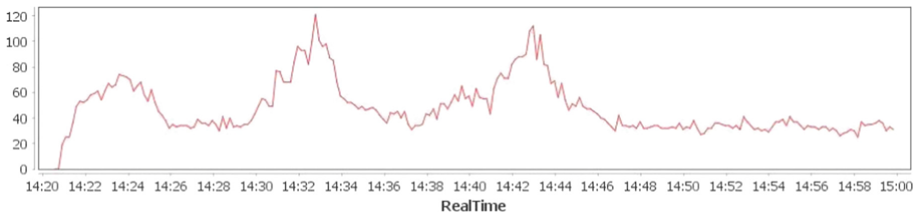

Fig. 9. Goods Delivery Process: Average Execution Time

5 Related Work

Several works on the analysis and provisioning of resources can be found in the literature. Schömig and Rau [27] use coloured stochastic Petri nets to specify and analyse business processes in the presence of dynamic routing, simultaneous resource allocation, forking/joining of process-control threads, and priority-based queuing. 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.* [20] introduce *multidimensional workflow nets* to model and analyse resource availability and workload. Oliveira *et al.* [23] use generalised stochastic Petri nets for correctness verification and performance evaluation of business processes. In their work, an activity can be associated with 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 [22] for understanding how bounded resources can impact the behaviour of a process. They introduce the notion of "flexible resource allocation" as a way to assign resources associated with a given role based on priorities. In their approach, alternative strategies are used to better allocate a fixed number of available resources. Havur *et al.* [15] 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.* [28] describe a stochastic method for quantifying resource utilisation relative to structural properties of processes and past executions.

In [29], a solution is presented to optimise resource allocation by focusing on the structure of the process, and more precisely on dependencies between resources and tasks. The approach then proposes a solution to adapt the structure of the business process to better fit the resources available in the enterprise. The authors in [7] focus on the specification and verification of concurrently running processes, operating in time-critical scenarios and having assigned a limited amount of resources. The authors propose to use a fragment of first-order logic to capture process fragments along the timeline and to combine them in a sound model, by observing constraints defined on both activity durations and resource availability. In [25], a contribution to the field of business process simulation is made by providing a new simulation engine, which supports advanced resource specificities such as queuing mechanisms, resource dependencies, or simulation parameters. A conceptual model supports these features and a prototype implementation of this conceptual model are proposed. Incorporating these features also allows for more accurate simulation of the processes and obtaining more relevant performance metrics. Finally, [16] presents a framework to integrate optimised resource allocation in business processes by adding a new component called *resource manager*. It is responsible for maintaining all relevant information concerning the availability of resources and for allocating resources to a process instance. The process designer can specify resource requirements within the business process model through dedicated resource-allocation activities.

There are many tools supporting the design and management of business processes (e.g., Activiti, Bonita, Camunda, or Signavio), of which a subset supports the analysis and optimisation of processes. For instance, this is the case of

Signavio [1], which packs tools such as the Signavio Process Intelligence for process optimisation. It 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. The proposal here takes a different approach since the idea is to compare the possibility to make the decision at design time or at runtime, with static or dynamic allocation of resources.

This work is part of a long term project with the goal of developing different tools for the analysis of BPMN processes. [18,19] present an approach transforming BPMN into the input language of the CADP model checker, thus allowing the automated verification of functional properties and the comparison of BPMN processes. In [12], basic BPMN processes were specified. This work provides operations for the estimation of execution times, and uses model-checking techniques to verify reachability problems and LTL properties. In [9], a model similar to the current one was proposed and was used for stochastic analysis using the statistical model checker PVeStA [3]. In [10], Maude is used to model and analyse the resource allocation of business processes. In that work, optimal allocation is presented as a multi-objective optimisation problem, where response time and resource usage are minimised. [11] 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. Four different strategies were implemented and compared from an experimental perspective. [14] presents an approach to perform probabilistic model checking of multiple executions of a BPMN process (including time and resources) at runtime.

6 Concluding Remarks

In this paper, the focus is on business processes developed using the BPMN notation extended with a description of time and resources. Processes are executed several times and those multiple instances compete for the shared resources. In this context, several metrics can be computed, such as average execution time or resource usage percentage. These metrics are helpful to optimise processes by, for instance, increasing the usage of resources or reducing the average execution time. Two different options to compute these metrics have been presented. The first approach relies on off-line simulation techniques and assumes that the allocation of resources is static (same number of resources). The second approach applies at runtime, which requires the instrumentation of an existing platform for executing BPMN processes. This latter approach is dynamic and the number of replicas can be updated for each resource during execution to adapt to a change in the resource usage. Both approaches are fully automated and have been applied to realistic processes.

The static approach is useful for a process that is under development and thus can be refined before being effectively deployed. This approach thus allows users to better understand the process and improve it in the early stage of its development. The static approach does not permit adjusting the resources to the workload, but still corresponds to realistic scenarios. This is the case, for instance, when the number of resources cannot be changed with simple or quick fixes. Complementarily, the dynamic approach adjusts at runtime the number of resources, resulting in a more stable resource usage in terms of occupancy percentages. However, this dynamic change is not always possible since there are some specific kinds of resources (such as human beings) that cannot be immediately or automatically updated. Another difference of the dynamic approach is that it applies to any executable BPMN (no restriction at the syntactic level), whereas the static approach works for a subset of BPMN and also requires probabilities for split exclusive and inclusive gateways.

The main perspective of this work is to investigate how AI techniques could help to develop new allocation strategies based on prediction analytics. More precisely, such techniques could be used to predict the resource usage in the short future and the strategy would rely on these values in order to anticipate the change in the number of resource replicas.

Acknowledgements. This work was supported by the Région Auvergne-Rhône-Alpes within the "*Pack Ambition Recherche*" programme. The first author was partially supported by projects UMA18-FEDERJA-180 (J. Andalucía/FEDER) and PGC2018-094905-B-I00 (Spanish MINECO/FEDER). The third author was in part supported by the ECOS-NORD project FACTS (C19M03).

References

1. Signavio (2019). https://www.signavio.com
2. Activiti: Open source business automation. Accessed Dec 2021
3. AlTurki, M., Meseguer, J.: PVeStA: a parallel statistical model checking and quantitative analysis tool. In: Corradini, A., Klin, B., Cîrstea, C. (eds.) CALCO 2011. LNCS, vol. 6859, pp. 386–392. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22944-2_28
4. Bartocci, E., Falcone, Y. (eds.): Lectures on Runtime Verification. LNCS, vol. 10457. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75632-5
5. Christiansen, D.R., Carbone, M., Hildebrandt, T.: Formal semantics and implementation of BPMN 2.0 inclusive gateways. In: Bravetti, M., Bultan, T. (eds.) WS-FM 2010. LNCS, vol. 6551, pp. 146–160. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19589-1_10
6. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude - A High-Performance Logical Framework. LNCS, vol. 4350. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71999-1
7. Combi, C., Sala, P., Zerbato, F.: A logical formalization of time-critical processes with resources. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNBIP, vol. 329, pp. 20–36. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98651-7_2
8. Durán, F., Rocha, C., Salaün, G.: A Note on Resource Allocation Analysis of BPMN Processes (2018). http://maude.lcc.uma.es/BPMN-R
9. Durán, F., Rocha, C., Salaün, G.: Stochastic analysis of BPMN with time in rewriting logic. Sci. Comput. Program. **168**, 1–17 (2018)

10. Durán, F., Rocha, C., Salaün, G.: A rewriting logic approach to resource allocation analysis in business process models. Sci. Comput. Program. **183** (2019)
11. Durán, F., Rocha, C., Salaün, G.: Resource provisioning strategies for BPMN processes: specification and analysis using Maude. J. Log. Algebraic Methods Program. **123**, 100711 (2021)
12. Durán, F., Salaün, G.: Verifying timed BPMN processes using Maude. In: Jacquet, J.-M., Massink, M. (eds.) COORDINATION 2017. LNCS, vol. 10319, pp. 219–236. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59746-1_12
13. Falcone, Y., Krstić, S., Reger, G., Traytel, D.: A taxonomy for classifying runtime verification tools. Int. J. Softw. Tools Technol. Transfer **23**(2), 255–284 (2021). https://doi.org/10.1007/s10009-021-00609-z
14. Falcone, Y., Salaün, G., Zuo, A.: Probabilistic model checking of BPMN processes at runtime. In: ter Beek, M.H., Monahan, R. (eds.) IFM 2022. LNCS, vol. 13274, pp. 191–208. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-07727-2_11
15. Havur, G., Cabanillas, C., Mendling, J., Polleres, A.: Resource allocation with dependencies in business process management systems. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNBIP, vol. 260, pp. 3–19. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45468-9_1
16. Ihde, S., Pufahl, L., Lin, M.-B., Goel, A., Weske, M.: Optimized resource allocations in business process models. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNBIP, vol. 360, pp. 55–71. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26643-1_4
17. ISO/IEC. International Standard 19510, Information technology - Business Process Model and Notation (2013)
18. Krishna, A., Poizat, P., Salaün, G.: VBPMN: automated verification of BPMN processes (tool paper). In: Polikarpova, N., Schneider, S. (eds.) IFM 2017. LNCS, vol. 10510, pp. 323–331. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66845-1_21
19. Krishna, A., Poizat, P., Salaün, G.: Checking business process evolution. Sci. Comput. Program. **170**, 1–26 (2019)
20. Li, J., Fan, Y., Zhou, M.: Performance modeling and analysis of workflow. IEEE Trans. Syst. Man Cybern. **34**(2), 229–242 (2004)
21. Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. Theor. Comput. Sci. **96**(1), 73–155 (1992)
22. Netjes, N., van der Aalst, W., Reijers, H.: Analysis of resource-constrained processes with colored Petri Nets. In: Proceedings of CPN. DAIMI, vol. 576, pp. 251–266 (2005)
23. Oliveira, C., Lima, R., Reijers, H., Ribeiro, J.: Quantitative analysis of resource-constrained business processes. Trans. Syst. Man Cybern. **42**(3), 669–684 (2012)
24. OMG. Business Process Model and Notation (BPMN) - Version 2.0, January 2011
25. Peters, S.P.F., Dijkman, R.M., Grefen, P.W.P.J.: Advanced simulation of resource constructs in business process models. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNBIP, vol. 329, pp. 159–175. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98651-7_10
26. Polyak, B.: Introduction to Optimization. Translations Series in Mathematics and Engineering. Optimization Software Inc. (1987)
27. Schömig, A.K., Rau, H.: A Petri Net Approach for the Performance Analysis of Business Processes. Technical Report 116, Universität Würzburg, Würzburg, Germany, May 1995

28. Sperl, S., Havur, G., Steyskal, S., Cabanillas, C., Polleres, A., Haselböck, A.: Resource utilization prediction in decision-intensive business processes. In: Proceedings of SIMPDA, CEUR Workshop Proceedings, pp. 128–141 (2017)
29. Xu, J., Liu, C., Zhao, X.: Resource allocation vs. business process improvement: how they impact on each other. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 228–243. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85758-7_18