A Context-Based Approach for Real-Time Adaptation Need Detection

Jamila Oukharijane[1][a], Imen Ben Said[1], Mohamed Amine Chaâbane[1], Eric Andonoff[2] and Rafik Bouaziz[1]

[1]*MIRACL Laboratory, University of Sfax, Tunisia*

[2]*IRIT, University Toulouse 1-Capitole, 2 Rue du Doyen Gabriel Marty, 31042 Toulouse Cedex, France*

Keywords: Context, Real-Time Monitoring, Situation Reasoning, Situation Filtering, Adaptation Need Detection, Context Analysis.

Abstract: The increasingly dynamic and changing environment in which business processes are evolving requires companies to adapt them frequently. Thus, we propose in this paper a new context-based approach recommending to analyze well the environmental changes. This approach recommends to structure knowledge required for process adaptation need in accordance with our BPMN4V-Context meta-model, which supports the modeling of the operating environment, using the context along with the versions of the model of each BPMN process and their use conditions. It also advocates (i) a filtering activity to retain only significant context changes in the monitored data, as low-level context parameters, (ii) a reasoning activity to deduce high-level context parameters from filtered low-level ones, enhancing the current situation of running processes and (iii) examining the current situation before its analysis in order to resolve problems related to the used units and synonym values. Finally, the feasibility and applicability of this approach is demonstrated by a case study from the crisis domain and two performance tests.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

The dynamic environment in which companies operate forces them to frequently adapt their processes to face changes occurring in this environment. Thus, their capability to rapidly and efficiently adapt their running processes to changes is an essential requirement for them. So, Business Process Management Systems (BPMS) implementing processes often support manual adaptation; the process designer has to identify which changes in the operating environment require an adaptation and resolve them by defining the required adaptation operations and carrying them out (Rinderle et al., 2004). However, the manual process adaptation is a costly, time-consuming and error-prone task (Masoumi et al., 2013). In fact, it requires the presence of a business expert who must possess knowledge of the environment in which supervised processes operate to be able to identify each adaptation need and resolve it. Consequently, self-adaptation is seen as an effective solution to deal with the complexity of process adaptation with minimum human (De Lemos et al., 2013) (Oukharijane et al., 2019).

[a] https://orcid.org/0000-0001-6155-0215

As argued in (IBM, 2006), self-adaptation in the general level encompasses various self-* properties in the major level, including self-configuring, self-healing, self-optimizing and self-protecting. We consider that the approach presented in this paper falls under self-healing category. Self-healing is the capability of the adaptation engine of discovering, diagnosing and reacting to disruptions (Huebscher and McCann, 2008). Self-healing can be classified into self-diagnosing and self-repairing, where the former concerns itself with identifying adaptation needs, and the latter focuses on the resolution of the identified adaptation needs, namely the definition and the execution of the adaptation operations. As this paper deals with the adaptation need detection issue, it provide a comprehensive approach for self-diagnosis.

On the other hand, self-adaptive systems must have the capability of self-adjusting to the variations of their operating environment, which is often referred to as context. To address process self-adaptation, we thus have to deal with the detection of process adaptation needs. In the BPM area, the notion of context is defined in (Rosemann et al., 2008) as "*the minimum of variables containing all relevant information that impacts the design and the execution*

Copyright © 2023 by SCITEPRESS – Science and Technology Publications, Lda. Under CC license (CC BY-NC-ND 4.0)

of a process". For his part, (Rosemann et al., 2008) outlined that four relevant types of context must be considered: (i) an *immediate context*, which covers information on the behavioral, informational and organizational dimensions of processes, which are three important dimensions to be considered when modeling processes, (ii) an *internal context*, which covers information about the internal environment of an organization that impacts the processes, (iii) an *external context*, which covers information about the external stakeholders of an organization, and finally, (iv) an *environmental context*, which covers information about external factors. If several literature contributions addressed the adaptation need detection issue, their recommended solutions had drawbacks. Firstly, they did not address the modeling of the operating environment in a comprehensive way as they did not consider all context types defined in Rosemann's taxonomy (Rosemann et al., 2008). Second, their models that support the representation of the operating environment are rather simplistic or even poor, as only low-level context parameters are supported to model the operating environment as a context. They do not contain high-level context parameters inferred by a reasoning component in order to deduce new information that enhances the current situation knowledge.

To overcome these weaknesses, we recommend a context-based approach, which addresses the adaptation need detection issue and has the four following features:

- First, it aims to ensure for a powerful monitoring of the operating environment of running processes, able to capture relevant context changes and enhance context parameters. Therefore, it allows better analysis of changes to identify whether or not process adaptations are required.
- Second, it recommends the use of the *push* mode (*i.e.*, publish/subscribe communication), as a technique for current situation acquisition. The advantages of this mode are as follows: (1) real-time monitoring and processing of context changes and (2) loose coupling between the operating environment and the adaptation engine, which makes the integration of new sensors in the operating environment easier as the adaptation engine does not need to be modified.
- Third, it recommends the context notion to represent the operating environment of processes, *i.e.*, the current situation in which running processes operate, using context parameters belonging to any type of context parameter types defined in Rosemann's taxonomy.
- Finally, it recommends the reasoning on context parameters, which contributes to the enhancement of the current situation by high-level context parameters and thus the improvement of the decision-making for process adaptation.

Accordingly, the remainder of the paper is organized as follows. Section 2 provides the state-of-the-art on self-adaptation of processes, focusing mainly on adaptation need detection. Section 3 introduces the BPMN4V-Context we propose for the modeling of knowledge required by the adaptation need detection. Section 4 discusses in detail the context-based approach proposed for process adaptation need detection. Section 5 demonstrates the applicability of the proposed approach on the case study, whereas, Section 6 reports on the evaluation of the recommended solution to demonstrate its feasibility. Finally, section 7 summarizes the paper contributions and gives some directions for future research.

2 RELATED WORK

In the literature, there are several works that focus on the automatic and autonomous context monitoring of running business processes and eventually on adapting them to the occurring context changes (*e.g.*, (Ayora et al., 2012), (Ayoub and Elgammal, 2018), (Ferro and Rubira, 2015), (Monteiro et al., 2008), (Oliveira et al., 2013) and (Seiger et al., 2019)). Due to the space limitation, this section considers only the recent contributions that focus on autonomic detection of the adaptation need.

For their part, Oliveira et al. (Oliveira et al., 2013) introduced an approach that enables the modeling of autonomic processes at the design-time and managing them at the run-time. At the design-time, this approach makes it possible to model an autonomic process using models, which provide all the necessary concepts that guide the self-adaptation of processes at the run-time, *i.e.,* which tasks must be monitored, which context changes impact the execution of a process and how to resolve them. At run-time, this approach goes through the following steps. first, it periodically requests to retrieve the value of any context element in the operating environment. Then, it checks all the variation points and evaluates the received values according to the modeled context in each variant in order to detect the adaptation needs. Therefore, if adaptations are required, it selects and executes an alternative variant that satisfies the context of the operating environment. Moreover, this approach does not address the separation of concerns between the adaptation logic and the operating environment. Furthermore, while the acquired context situations are modeled according to a meta-model, the modeling of these situations is not addressed in a comprehensive way encompassing all the context types defined in Rosemann's taxonomy (Rosemann et al., 2008). In addition, this approach does not support the context situation reasoning in order to enhance the current situation with high-level context parameters; only the low-level context situations are supported. finally, it does not allow the detection and resolution of situation problems related to the used units and to the synonymous values.

Moreover, another interesting contribution by Ferro and Rubira (Ferro and Rubira, 2015) introduced an agent-based adaptation engine for the self-adaptation of processes at run-time. This adaptation engine acts as follows: The *monitor* agent continuously acquires and evaluates the current state of the process instance and its context until it determines a symptom that needs to be analyzed (*e.g.,* business rule violation). The *adapter* agent implements the decision making for adaptation, which is driven by goal and business rule analysis and the operations required for the resolution of the detected adaptation need. Finally, the *executor* agent sends adaptation operations to the process engine to be executed. This contribution supports the pull mode for the acquisition of situations of internal context elements in the operating environment, which implies that it embeds the current situation capturing code into the *adaptation engine* and therefore, leads to poor maintainability. Besides, it does not consider all context types defined in Rosemann's taxonomy (Rosemann et al., 2008). Finally, the context situation reasoning is not implemented, which implies that the high-level context situations are not supported.

On the other hand, Ayoub and Elgammel (Ayoub and Elgammal, 2018) introduced a framework for monitoring and improving social business process (i.e., process that integrates Web 2.0 technologies such as Facebook and Twitter). In addition to this framework, these authors recommend a concrete approach that enables the adaptation need detection based on social data as it particularly uses data mining and machine learning techniques to detect the need for process adaptations. Thus it supports the predictive analysis in detecting the adaptation needs. However, this approach does not separate the concerns between the adaptation logic and the operating environment. In addition, this approach is specific as it gathers and models only social data. Thus it enables the modeling of only external context type defined in Rosemann's taxonomy (Rosemann et al., 2008). Finally, it does not support the interpretation of current situation, *i.e.,* enhancement of the current situation by adding new knowledge and translation of situation values to resolve problems related to their units and synonyms.

Finally, Seiger et al. (Seiger et al., 2019) proposed a framework that enables the self-adaptation of processes in the cyber-physical systems. More precisely, this framework monitors and analyzes the consistency between the sensed physical world and the assumed cyber world of task executions. In case an inconsistency is detected, the resource involved in the concerned task execution is replaced by another alternative resource, and then the task is executed. As in (Ayoub and Elgammal, 2018), this framework uses the pull (*i.e.,* request/response) mode for the retrieval of internal context element values in the operating environment without addressing the modeling of all context types defined in Rosemann's taxonomy (Rosemann et al., 2008) and the separation of concerns between the adaptation logic and the operating environment. In addition, this framework does not support the interpretation of current situation. Therefore, only the low-level context situation is supported.

To overcome the weaknesses of the examined contributions, we recommend a self-diagnosis approach which takes into account the interesting features of the examined adaptation contributions, namely, the use of the context-based model to define the current situation context of each running process. But our approach differs from them in the following respects. First, it advocates taking into account the Rosemann's taxonomy for a comprehensive modeling of process contexts, where the context parameters observed from the immediate, internal, external and environmental contexts of processes may be represented. Second, it supports real-time context monitoring by implementing the push technique for the current situation acquisition from sensors. Finally, it recommends the reasoning on context parameters in order to infer new knowledges.

3 KNOWLEDGE FOR ADAPTATION NEED DETECTION

Our approach recommends to model knowledge required by the adaptation need detection using the two levels of BPMN4V-Context meta-model recommended in (Gamma et al., 1993). An excerpt of the UML class diagram of BPMN4V-Context is given in Figure 1. This figure adopts the following policies: blue background for concepts related to private process versions, grey background for concepts related to the definition of the use conditions of versions, yellow background for concepts corresponding to process executions, and finally green background for concepts related to current situation modeling of running process. In this figure, we particularly focus on two abstraction levels for the modeling of the context parameters: (i) the *model level*, in which we describe the use conditions of versions of processes at design-time, and (ii) the *instance level*, in which we describe the current situation of running processes at run-time.

These levels declare the structures of this knowledge as described below.

3.1 Model Level

The model level (left side of Figure 1) of BPMN4V-Context meta-model concerns the concepts involved in the definition of the process, sub-process and task versions and their use conditions at design-time. In fact, this meta-model introduces new classes to define these conditions of use and the involved context parameters. A use condition may be atomic or composite. Therefore, the class of the *Use condition* is specialized in two sub-classes: *Atomic use condition* and *Composite use condition*. An atomic use condition refers to the *context parameter* involved in this condition and defines the associated *operator*, *value* and *unit* of measurement. A composite use condition is an aggregation of several use conditions connected together by logical operators (and, or). For each defined operator, we define the *rank* that indicates the position of the use condition in the composite use condition expression. It should be noted that the use condition of a process (or sub-process) version is the combination of the use conditions of activities (which are versions of tasks or versions of sub-processes) that make up this process (or sub-process) version.

3.2 Instance Level

This level (right side of Figure 1) of BPMN4V-Context meta-model enables to model the operating environment of running process versions. It deals with the concepts involved in the definition of running process versions and their corresponding current situations. Each occurrence of *Running process* is related to exactly one occurrence of *Version of Process*, but an occurrence of *Version of Process* can be related to several occurrences of *Running process*. An occurrence of *Running process* is composed of a set of occurrences of *Running activity*. An occurrence of *Running activity* can be an occurrence of *Running sub-process* or of *Running task*. Moreover, current situations of running tasks are modeled as instances of the class *Current situation*, which may be either atomic or composite:

- An *atomic situation* is defined by a context parameter, a value, a timestamp and a state: *context parameter* can be immediate, internal, external or environmental context parameter according to the Rosemann' taxonomy; *value* specifies the acquired value by the Monitor component; *unit* specifies the used unit for measuring a context parameter value (*e.g.*, degrees Celsius, pressure, meter, etc.); *timestamp* indicates the time value when the context parameter value was acquired.
- A *composite situation* is a set of atomic situations connected together by the logical operator "and".

Note that the current situation of a running process or sub-process is the union of the current situations of its running activities, *i.e.*, running tasks and running sub-processes.

Knowledge structures expressed at these two levels are respectively implemented by the *Models repository* and the *Instances repository*

4 THE CONTEXT-BASED APPROACH FOR THE PROCESS ADAPTATION NEED DETECTION

Our approach addresses the adaptation need detection aiming to analyze changes of the operating environment in order to identify whether or not process adaptations are required. This approach is based on (i) sensors and the push mode to capture context parameter values in real-time, *i.e.*, during the execution of processes, and (ii) context notion to represent both the operating environment of process model versions (the current situation in which running process versions operate) and the use condition of each process model version. Therefore, the notion of context is crucial because it serves as a basis for identifying adaptation needs. Indeed, identifying adaptation needs is like comparing these two contexts (*i.e.,* current situation and use condition). The process driving this approach is shown in Figure 2 as a BPMN collaboration diagram.

As shown in this diagram, sensors and process engine listeners simultaneously observe the execution of the running process versions and their operating environment, and push to the *adaptation engine* the events indicating changes occurring on the context parameters. But as indicated in (Da et al., 2014), the received data from sensors, without any further interpretation, can be meaningless, trivial, vulnerable, or

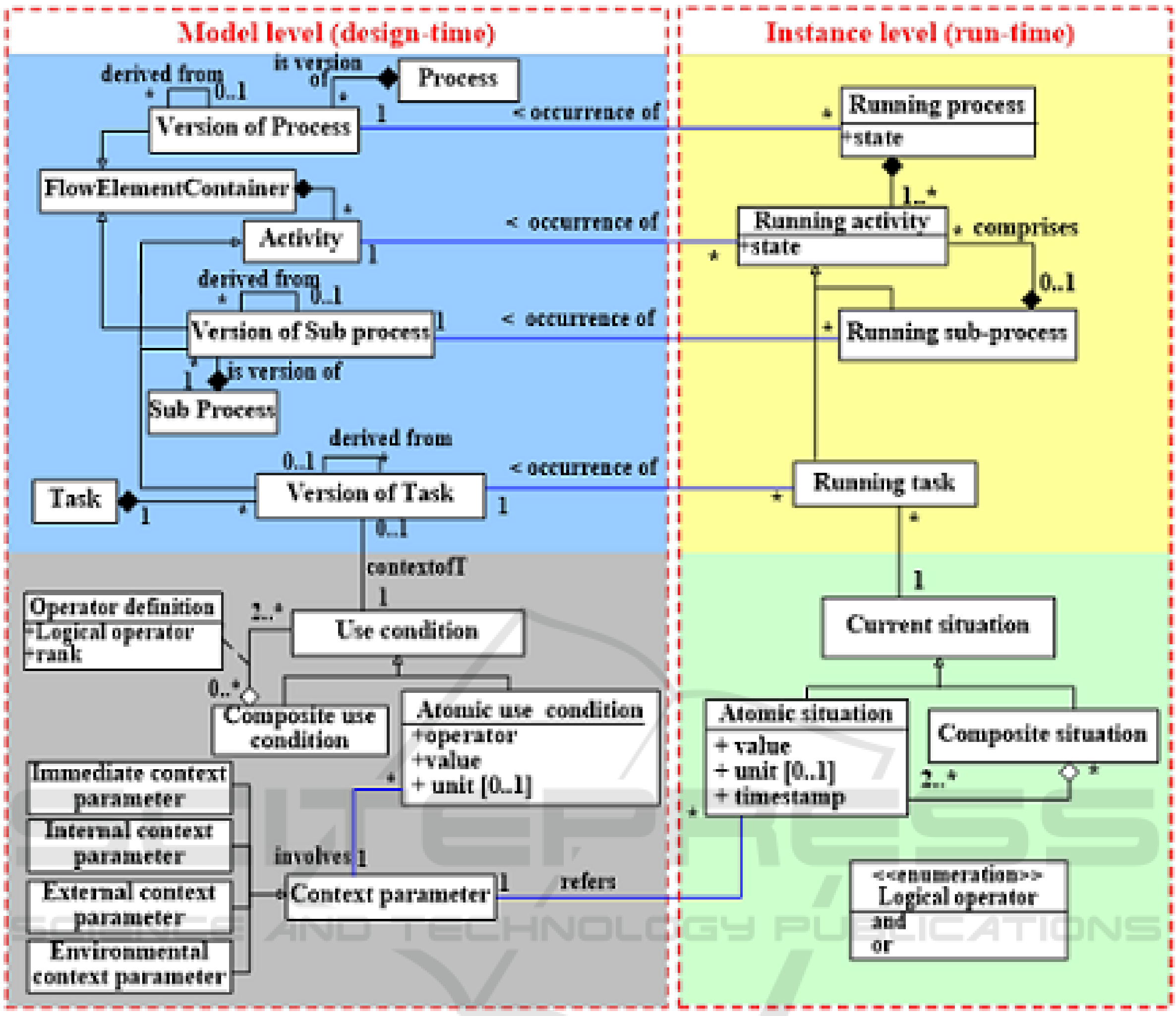

Figure 1: BPMN4V-Context meta-model as a UML class diagram.

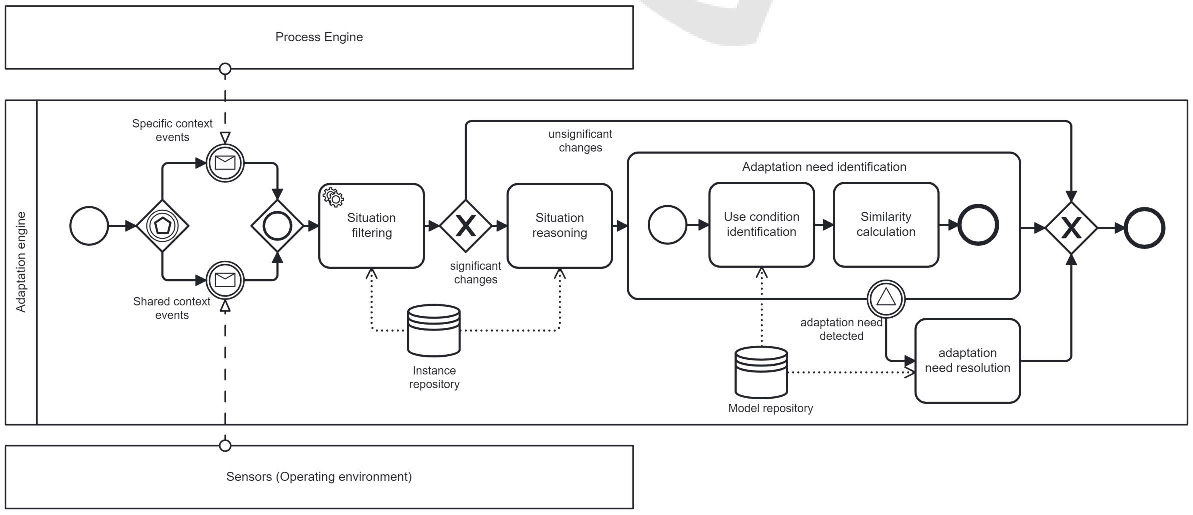

Figure 2: Adaptation need detection process as a BPMN diagram.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

uncertain bout small changes. To overcome this problem, our *adaptation engine* first executes a situation filtering activity to ensure that the reasoning activity will process only the significant data changes. This *Situation filtering* activity filters the received data and keeps only the significant changes of context parameters values. To do so, this activity compares the absolute value of the difference between the received value of each low-level context parameter and the last stored value in the *Context repository* to a certain variation threshold (*var-threshold*) stored also in the *Context repository*. So, it verifies if the received change of each context parameter value is significant (*i.e.*, the absolute value of the difference is higher than the variation threshold) or insignificant. In case all the received context changes are insignificant, the *Adaptation engine* does not trigger the situation reasoning, nor the adaptation need identification. Consequently, both the reasoning and analysis times are reduced, only by processing and analyzing the significant context changes. Otherwise (*i.e.*, when there is one or more significant context changes are received), both activities "*Situation reasoning*" and "*Adaptation need identification*" are triggered for enhancing this situation by high-level context parameters values and detecting the need for process adaptation, respectively. More precisely, the first activity includes the following both internal operations to deduce the high-level context parameters from filtered low-level ones:

- The first operation, *Situation aggregation*, uses aggregation functions (*i.e.*, average, count, max, sum, etc.) of (low-level/high-level) context parameters to deduce high-level context parameters. For instance, the value of the high-level context parameter *precipitation amount* is the aggregation *sum* of the precipitation measurements acquired by the "precipitation sensors" located at different sources points of a river.
- The second operation, *Situation deduction*, executes the rules that ensure deducing new high-level context parameters using low-level/high-level context parameters. Let us remember that the deduction rules are defined by domain experts in the *Context repository*. The aim of this operation is to deduce a set of high-level context parameters enhancing the current situation of running processes, consequently helping the process adaptation.

After executing these operations, the ***Adaptation needs identification*** activity is triggered to identify the possible adaptation needs of the considered process instance. This activity is modeled as sub-process in the BPMN diagram, which includes the following atomic activity: "*Use condition identification*" and "*Similarity calculation*". The ***Use condition identification*** activity allows to obtain the use condition corresponding to the considered process version V_P; it generates an appropriate query to access the *Models repository* implementing the *Model part of the BPMN4V-Context meta-model* and retrieve the use condition C of V_P. Then, the second activity, Similarity calculation, calculates the similarity between the identified use condition C and the enhanced current situation S of V_P. The algorithm called *Sim*, shown below, implements the similarity calculation. It receives both use condition C and enhanced current situation S of a process version V_P as input, and returns a similarity percentage "sim_p" of V_P. It calculates the similarity value "sim" of V_P as the number of its atomic use conditions verified by the values of the context parameters of the current situation. It compares the value of each context parameter in the current situation "VPC_{CS}", and the value of the context parameter specified in the atomic use condition "VPC_{UC}". Note that this comparison may require a conversion of VPC_{CS} or its replacement with the appropriate synonym when VPC_{UC} is expressed with a different unit of measure or a synonym of VPC_CS, respectively. If an atomic use condition is verified, then sim is incremented by 1. Finally, the algorithm returns the similarity percentage sim_p of the atomic usage conditions. sim_p is equal to the number of the verified atomic usage conditions of V_P divided by its total number of atomic usage conditions. This algorithm uses the following functions:

- **convertUnit (vs, us, uc):** it converts the context parameter value vs if necessary from the unit of measurement us defined in the atomic situation to the unit of measurement uc defined in the atomic use condition. For example, the current value of the context parameter temperature is measured using the *Fahrenheit unit* (*e.g.*, 77 °F), whereas the unit of *temperature* is defined in *degrees Celsius* in the atomic use condition; then, the current value of the context parameter *temperature* is converted into degrees Celsius (*e.g.*, 23 °C) by this function.
- **synonyms (v):** it resolves the possible synonymy problem when the value v of a context parameter is expressed by a synonym of that defined in the atomic use condition, using a dictionary of equivalent representations. For instance, the value of the context parameter *resource availability* can be 0 or 1 in the current situation, whereas the defined value of *resource availability* in the atomic use condition is rather "available" or "unavailable".
- **checkCondition** (v_s, op_c, v_c)**:** It returns *true* if the value vs of the context parameter involved in the atomic situation is verified in the use condition,

otherwise, it returns *false*. More precisely, it compares the current value vs of the context parameter to the defined value v_c in the atomic use condition using the operator op_c.

- **card (cs):** It returns the number of atomic situations involved in the current situation *cs*.

Algorithm 1: Similarity calculation: Sim.

```
Require: $S$: Current situation,$C$: Use condition
  Local
  $sim \leftarrow 0$: real, $ps$: Atomic situation, $c$: Atomic use condition
  begin function
  for each $c$ in $C$ do
      if isCompositeCond ($c$) then
          ▷ Case where c is a composite use condition
          SimilarityCalculation (S, c)
      else
              ▷ Case of $c$ is an an atomic use condition
          for each $s$ in $S$ do
              if (c.getContextPar () = s.getContextPar ()) then
                  if ($s.getUnit() \neq c.getUnit()$) then
                      $vs$ ⇐convertUnit (s.getValue (), s.getUnit (), c.getUnit ())
                  end if
                  if checkCondition ($vs$, c.getOperator (), Synonyms (c.getValue ())) then
                      $sim \Leftarrow sim+1$
                  end if
              end if
          end for
      end if
  end for
  Return $sim/card(S)$
  end function
```

It should be noted that *Algorithm Sim* returns 1 as similarity value when all atomic situations featuring the current situation verify the use condition of the considered process version. This means that the current situation of the considered process version exactly matches the use condition of this process version. But when this algorithm returns a similarity value ¡ 1, a need for process adaptation is identified. In this case, a signal event (*need process instance adaptation* event) is triggered for defining the operations required to address this need using the hybrid approach recommended in (Oukharijane et al., 2020; Oukharijane et al., 2021).

5 CASE STUDY: THE FLOOD MANAGEMENT PROCESS

This section presents the Flood Management Process (FMP) case study and illustrates the applicability of the approach. First, it shows how to model the Flood Management Process and their context then, it uses this case study to demonstrate the execution of a sample adaptation need detection using the recommended approach.

5.1 Knowledge Modeling for FMP

This section presents a real case study related to the process of flood crisis management of the major French river "the Loire", on the city of Orleans. As shown in Figure 3, this process is triggered in urbanized impacted area when the water level of the river "Loire" in France rises above 3 m^3. In response to this event, the crisis cell decides whether or not to evacuate people from the flooded zones by assessing the flooding situation. In case an evacuation is needed, the Prefect emits an evacuation order, then the COD, which is the operational committee set up within the crisis cell, and the town hall inform the media and the population about the flood. After that the gendarmes proceed to the evacuation of people from the flooded zones. Finally, the crisis cell reports on the evacuation, and the Prefect sends the report to the interior ministry.

Like any other process, the FMP might be subject to different operating environment variations, such as water level, water flow rate, amount of precipitation, that could interrupt its functioning. In order to supervise this environment, we refer to the following context parameters among others:

- **Water level**, which indicates the level of water rising in the Loire. In the Orléans city, this parameter refers to the average water levels recorded at the following two stations: the Pont Royal station and the Quai du Roi station.
- *Water velocity*, which indicates the speed of the water in the Loire in the different source points of water to the Loire. This parameter refers to the average water velocities recorded at Pont Royal and Quai du Roi stations.
- **Impacted area,** which features the size of the population potentially impacted,
- **Road state,** which can be *not flooded, flooded and derivable* or *flooded and blocked.* When the roads are not flooded, people evacuate themselves. On the other hand, when the roads are flooded but still derivable, the evacuation method is carried out by

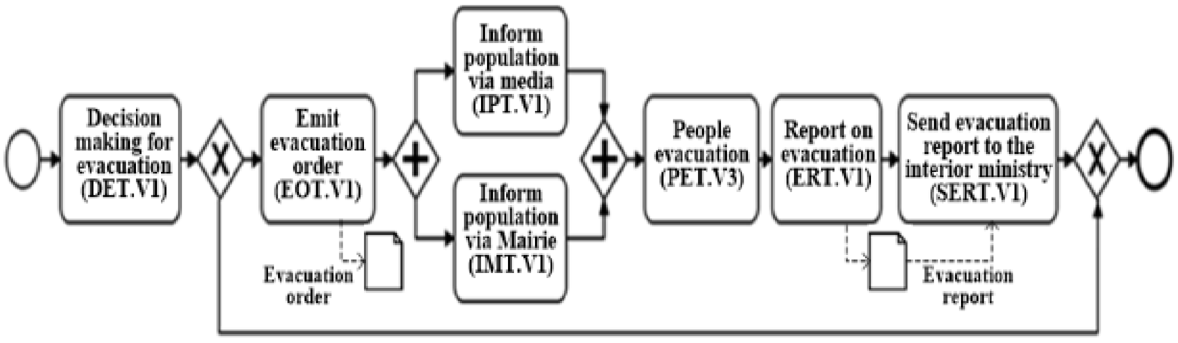
Decision making for evacuation (DET.V1)
Emit evacuation order (EOT.V1)
Inform population via media (IPT.V1)
Inform population via Mairie (IMT.V1)
People evacuation (PET.V3)
Report on evacuation (ERT.V1)
Send evacuation report to the interior ministry (SERT.V1)
Evacuation order
Evacuation report

Figure 3: Flood management process model.

land, which means that the evacuation task is carried out using specific vehicles with the help of gendarmes. However, when the roads are blocked because they are highly flooded, people's evacuation must be carried out by firefighters with zodiacs or helicopters in the most extreme cases.

- **Risk level**, which can be *low, moderate* or *high*. Indeed, the risk level of flooded zones is deduced from the following context parameters: *water level* and *water velocity*. Thus, when the water level is below of 2 m^3 and the water velocity is below of 2,500 m^3/s, thus the risk level is ”low”, whereas when the water level is more or equal to 2 m^3 and a water velocity is below of 2,500 m^3/s, so the risk level is ”moderate”. While, when the water level is higher than 3 m^3 and water velocity is more than 2500 m^3/s, thus the risk level is “high”.

On the basis of the previous context description, the parameters *Impacted area, Water level of a station, Water velocity of a station* and *Road state* represent the low-level context that must be measured by sensors and process engine listeners. For each of these parameters, of the numeric type, a threshold value was defined in Table 1 in order to verify if the received change is significant or not. We note that these context parameters can be measured at different station points; for example, for the Orleans city the water level is measured at both the Pont Royal and the Quai du Roi stations.

As for the high-level context parameters, such as the Water level, Water velocity, Risk level, Personnel's role and Type of equipment, they are deduced from the low-level ones according to the aggregations functions and deduction rules identified by domain experts and which are given below:

On the other hand and according to cell crisis, the flood management process defined in Figure 3 is used in the following context (*i.e.,* use condition): Impacted area is urbanized, Water level is more than 3 m^3, Water velocity is below of 2500 m^3/s, road state is flooded and derivable, Type of equipment is vehicle, personnel role of evacuation is gendarme and Risk level is moderate.

5.2 Context-Based Approach Simulation

In this section, we illustrate the simulation of our context-based approach for the detection of the adaptation need. In order to demonstrate this, we suppose that the flood management process (*cf.,* Figure 3) is running for Orleans city affected by Loire's floods. For this illustration, we refer to the following changes from the operating environment that need adaptation: *Water level of Pont Royal station = 3 m^3* and *Water level of Quai du Roi station = 4 m^3* and *Water velocity of Pont Royal station = 3100 m^3/s* and *Water velocity of Quai du Roi station = 3300 m^3* and *Impacted area = “Urbanized”* and *Road state = “Flooded and derivable”*.

Once the adaptation engine receives these changes, it starts its *Situation filtering* activity to filter and keep only the significant changes. Then, it triggers the ”Situation reasoning” activity to enhance the current situation thanks to its both operations. The first operation enhances the situation thanks to aggregation functions (average) and deduces the values of the water level and velocity of the Loire from the measured values at both Pont Royal and Quai du Roi stations. These functions led respectively to the addition of the following average values: ”3.5 m^3” and ”3200 m^3/s” for the two high-level context parameters ”Water level” and ”Water velocity” describing the average water level and velocity of the Loire. Then the second operation enhances the situation thanks to the deduction rules R3 and R5 defined in Table 2. Once the deduction is made, the current situation will be enhanced by new high-level context parameters ”*risk level*”, *Personnel role* and ”*Type of equipment*” having respectively “high”, “Gendarme” and ”vehicles” values.

Table 1: Variation threshold of low-level parameters.

Context parameter	Threshold value
Water level of a station	0.5 m^3
Water velocity of a station	150 m^3/s

Table 2: An expect of the deduction rules.

N	Deduction rule
R1	If water level < 2 m^3 and Water velocity < 2500 m^3/s then Risk level = ”low”
R2	If Water level ≥ 2 m^3 and Water velocity < 2500 m^3/s then Risk level = ”moderate”
R3	If Water level > 3 m^3 and Water velocity ≥ 2500 m^3/s then Risk level = ”high”
R4	If Road state = “Not flooded” Then Type of equipment = “Vehicle”
R5	If Road state = “Flooded and derivable” Then Personnel role = “Gendarme” and Type of equipment = “Vehicle”
R6	If Road state = “Flooded and blocked” Then Personnel role = “Firefighter” and Type of equipment = “Zodiac”

According to this enhanced situation and the use condition defined in the previous section, the Analyzer of the running process version calculates the similarity using Algorithm 1. The latter gives a similarity in the result equal to 0.67, which implies that the conditions related to the context parameters of *water velocity*, the *risk level* and the type of equipment are not confirmed, therefore, the running process needs adaptation.

6 SIMULATION AND EVALUATION RESULTS

We evaluate in this section our contributions, which ensure the context monitoring and analysis functionalities, by simulations. To this end, we perform two performance tests. In the first test, we evaluate the reasoning execution times consumed by the adaptation engine using or not using the Situation Filterer to verify that the use of the Situation Filterer activity reduces the reasoning time, and thus enhances the performance of adaptation engine. Whereas, the second test evaluates the impact of the reasoning on context parameters to improve the outcome of the analyzer (*i.e.*, adaptation need, no adaptation need) using precision, recall, f-measure and error rate metrics.

6.1 The First Performance Test

Our aim in this sub-section is to report on an evaluation that measures the advantages of using Situation filtering activity to enhance the performance of context monitoring. To conduct this evaluation, we vary the number of sensors/listeners from 50 to 200, and we realize two experiments:

- First experiment (without filtering the received context changes from sensors/listeners): in this experiment, all the received context changes are interpreted in order to enrich the current situation with high-level context parameters.
- Second experiment (with filtering the received context changes from sensors/listeners): in this experiment, the adaptation engine uses the variation thresholds of low-level context parameters in order to filter the received changes by eliminating all the insignificant ones. Therefore, only notable changes are interpreted for inferring high-level context parameters values.

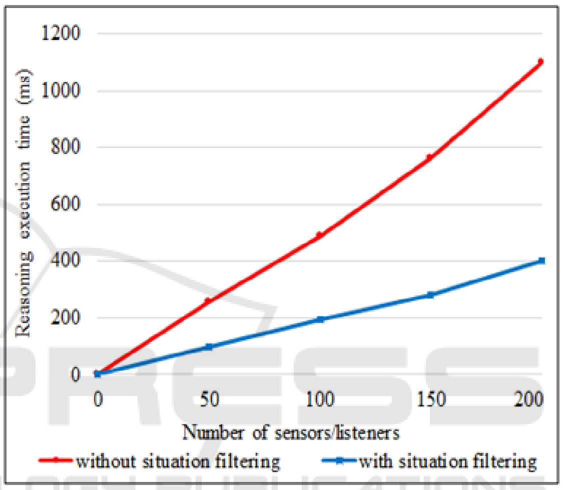

Figure 4: Comparison of the reasoning execution times with and without performing an upstream situation filtering.

Figure 4 illustrates the benefits of an upstream situation filtering on the reduction of the reasoning execution times consumed by the Situation Reasoning activity, respectively.

In the first experiment (without using an upstream situation filtering), we let the adaptation engine interprets all the received changes from sensors and listeners without filtering them. The results of this experiment are displayed by the red charts which confirm that the reasoning execution times grow exponentially with the number of sensors and listeners.

For the case of 200 sensors and listeners for example, the time taken by the Situation reasoning activity is about 1096.8 ms (*cf.* figure 4, red chart).

In the second experiment (with using an upstream situation filtering), we let the adaptation engine filters all the received changes of context parameters values by using the variation threshold (Var-threshold) of each context parameter and keeps only the significant ones. Figure 4 illustrates that the reasoning execution times are remarkably reduced, since the context parameter value is updated only if the difference between the received value and the stored value in the Context repository is higher than its variation threshold. In the case of 200 sensors and listeners, the reasoning execution time is decreased from more than 1096.8 ms (cf. figure 4, red chart) to about 400.3 ms (cf. figure 4, blue chart) by performing a situation filtering upstream of the situation reasoning, for example. Indeed, although the number sensors/listeners, sending changes to the adaptation engine, is increased, the monitoring performance is not seriously affected; this is the contribution of the Situation filtering activity that eliminates all insignificant context changes.

6.2 The Second Performance Test

As for the second test, we evaluate the effectiveness of our approach for the detection of the adaptation needs by reporting on an assessment that measures the advantages of implementing the Situation reasoning activity in the adaptation engine to enhance the current situation, and thus increase the accuracy of the adaptation needs. In this evaluation, we expose the possible scenarios for the detection of the adaptation needs. For each of these scenarios, we compared the provided results with our approach in which the Situation reasoning activity is implemented against the results provided without implementing the Situation reasoning activity. These scenarios, five in number, are as follows:

- SC1 (reasoning is not needed): detecting adaptation needs considering (i) a use condition that contains only low-level context parameters and (ii) a sensed situation containing all the low-level context parameters of the considered use condition.
- SC2 (unit translation is needed): detecting adaptation needs considering (i) a use condition that contains only low-level context parameters and (ii) a sensed situation containing low-level context parameter values expressed with units different from those specified in the considered use condition.
- SC3 (value translation is needed): detecting adaptation needs considering (i) a use condition that contains only low-level context parameters and (ii) a sensed situation containing low-level context parameters values which are synonyms of those of the considered use condition.
- SC4 (high-level context parameter deduction is needed): detecting adaptation needs considering (i) a use condition that contains both low and high-level context parameters and (ii) a sensed situation that contains all low-level context parameters of the defined use condition of the supervised task.
- SC5 (all the reasoning types are needed): detecting adaptation needs considering (i) a use condition that contains both low and high-level context parameters and (ii) a sensed situation containing low-level context parameter values expressed with units different from those specified in the considered use condition and others which are synonyms of those of this use condition.

To evaluate the effectiveness of our context-based approach for the detection of the adaptation needs, we have considered several changes related to the current situation of the flood management environment. For each of these scenarios, we have classified the outcome of adaptation engine using a confusion matrix. In this matrix, each column represents the outcome of the implemented adaptation engine (adaptation need, no adaptation need), where each row represents the real outcome. However, the cells of this matrix represent (i) *True positives (TP),* i.e., number of cases with adaptation needs that are correctly identified, (ii) *False negatives (FN),* i.e., number of cases with adaptation needs that are not identified, (iii) *False positives (FP),* i.e., number of cases with adaptation needs that are incorrectly identified and (iv) *True negative (TN)*, i.e., number of cases with no adaptation needs that are correctly identified. Then, we have measured the following metrics: (1) **Precision** defines how many adaptation needs are correctly identified among all the identified adaptation needs, (2) **Recall** defines how many adaptation needs are correctly identified among all the adaptation ones (correctly identified and not identified),(3) **F-measure** refers to the balanced mean between the precision and recall metrics, and (4) **Error rate** is calculated as the number of all incorrect cases divided by the total number of test cases. The best error rate is 0, whereas the worst is 1.

Table 3 summarizes the results obtained using these metrics for the two adaptation need detection approaches that respectively implement or not the situation reasoning.

Let us comment some of these results. For the first scenario SC1 and as indicated in Table 3, the precision and recall metrics are both equal to 1 for

Table 3: Evaluation outcomes of the two context-based approaches for the detection of the adaptation needs.

Scenarios	Results without considering situation reasoning				Results with considering situation reasoning			
	Precision	Recall	F-measure	Error	Precision	Recall	F-measure	Error
SC1	1	1	1	0	1	1	1	0
SC2	0.25	0.4	0.3	0.82	1	1	1	0
SC3	0.33	1	0.5	0.67	0.75	1	0.85	0.14
SC4	1	0.29	0.45	0.5	1	1	1	0
SC5	0	0	0	1	0.8	0.85	0.82	0.29

both approaches as the both versions of adaptation engine (with and without implementing the Situation reasoning activity) outcomes match exactly the real outcomes for all scenario test cases. This is due to the fact that each context parameter exists in both the use condition and current situation with the same unit and representation.

Regarding the result of SC2, detecting the adaptation needs with a translated situation thanks to the Unit translator is more efficient than detecting the adaptation needs without translating the situation received from the sensors. Indeed, the Unit translator function takes care of translating the units of the received situation from the sensors to the units defined in the use condition. Thus, the precision, recall and f-measure metrics for adaptation engine implementing this function are all equal to 1, and the error rate metric is equal to 0. However, to detect the adaptation needs considering the situation received from the sensors without unit translation, the precision metric is equal to 0.25 and the recall is equal to 0.4. This means that (i) a set of incorrect adaptation needs are identified for some scenario test cases and (ii) a set of adaptation needs that must be detected are not identified for some other cases. As a consequence, the values of the f-measure and error rate metrics are equal to 0.3 and 0.82, respectively. As for the result of SC3, detecting the adaptation needs considering an enhanced situation thanks to the Value translator gives very acceptable results, since the values of the precision and recall are on average equal to 0.75 and 1, respectively. On the other hand, detecting the adaptation needs considering only the situation received from sensors (without enhancement) sends back the values of 0.33 for the precision and 1 for the recall. As the recall metrics are equal to 1 for both approaches of the detection of the adaptation needs, it turns out that all adaptation needs that should have been detected are identified in the different simulated test cases from SC3. As a consequence, the values of the f-measure and error rate metrics are equal to (i) 0.85 and 0.14, respectively, for our approach in which the situation reasoning is considered and (ii) 0.5 and 0.67 for the approach in which the situation reasoning is not considered. For the result of SC4, detecting the adaptation needs considering an enhanced situation with high-level context parameters thanks to the Situation reasoning activity returns a value equal to 1 for the precision and recall metrics. On the other side, detecting the adaptation needs considering a situation without enhancement returns values equal to 1 and 0.29, respectively, for the precision and recall metrics. It should be noted that the precision metrics are always equal to 1, since all adaptation needs are correctly identified in the different simulated test cases from SC4, because they are detected on the basis of only the low-level context parameter values. As a consequence, the values of the f-measure and error rate metrics considering the situation reasoning is better than the values of these metrics without considering the situation reasoning. These results highlight the importance of the Situation reasoning activity. Finally, regarding the result of SC5, detecting the adaptation needs considering an enhanced situation is more efficient than detecting the adaptation needs without implementing the Situation reasoning and translating. Indeed, the Situation reasoning activity (i) detects and resolves situation problems related to the used units and to the synonymous values, and (ii) enriches the current situation with high-level context parameter values. Thus the error rate of the adaptation need detection is considerably reduced (from 1 to 0.29).

7 CONCLUSION

The complexity of processes, the dynamism of the environments in which they are operated, and the need for process adaptation to context changes are growing rapidly. So, process adaptations are often performed manually by process designers or process owners. However, a manual adaptation is a costly, time consuming and error prone task. For that reason, this paper recommends an approach for real-time adaptation need detection. Its contributions are as follows:

- It is context-based, which has enabled it to improve the detection of the adaptation needs. The context is represented by parameters from any type of contexts defined in Rosemann's taxonomy.

- It recommends well modeling the knowledge required for the adaptation need detections, as allowed by the two levels of BPMN4V-Context meta-model.
- It recommends (i) the use of *sensors* and the *push* mode to support a real-time monitoring of the operating environment of processes, (ii) the context changes filtering in order to only analyze significant changes, and (iii) the reasoning on context parameters to enhance the current situation.
- It allows comparing the operating environment of process model versions and the use conditions of these versions. Therefore, it serves as a basis for identifying adaptation needs.
- It ensures the translation of the context parameters values related to units and synonyms before analyzing. This translation makes it possible to avoid divergences of representation of these values in the current situation and in the use conditions, and thus to improve decision-making for process adaptations.

However, our approach can be improved. As future work, we plan to incorporate ontologies to better capture the semantics of the current situation of running processes and take advantage of semantic aspects. In addition, we plan to study how to structure and exploit historical data and how to conduct predictive analysis of the current situation of the operating environment in order to predict the future adaptation needs before they arise. Moreover, we have to evaluate the usability of our contributions.

REFERENCES

Ayora, C., Torres, V., Pelechano, V., and Alférez, G. H. (2012). Applying CVL to business process variability management. In *VARiability for You Workshop: Variability Modeling Made Useful for Everyone*, pages 26–31, Innsbruck, Austria. ACM.

Ayoub, A. and Elgammal, A. (2018). Utilizing Twitter data for identifying and resolving runtime business process disruptions. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 189–206. Springer.

Da, K., Montaury, P., Dalmau, M., Montaury, P., Roose, P., and Montaury, P. (2014). Kalimucho : Middleware for mobile applications. In *29th Annual ACM Symposium on Applied Computing*, pages 413–419.

De Lemos, R., Giese, H., Müller, H. A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N. M., and Vogel, T. (2013). Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer.

Ferro, S. and Rubira, C. (2015). An architecture for dynamic self-adaptation in workflows. In *International Conference on Software Engineering Research and Practice (SERP)*, pages 35–41.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1993). *Design patterns: Abstraction and reuse of object-oriented design*. Springer.

Huebscher, M. C. and McCann, J. A. (2008). A survey of autonomic computing—degrees, models, and applications. *ACM Computing Surveys (CSUR)*, 40(3):1–31.

IBM (2006). An architectural blueprint for autonomic computing. *IBM White Paper*, 31.

Masoumi, A., Marrella, A., and Soutchanski, M. (2013). Towards a Planning-Based Approach to the Automated Design of Chemical Processes. In *International Conference of the Italian Association for Artificial Intelligence*, volume 1101, pages 61–70, Trento, Italy.

Monteiro, P., Rodrigues, J., Oliveira, J., and Souza, J. (2008). ABPM: Autonomic Business Process Manager. In *Latin American Autonomic Computing Symposium*, GRamado.

Oliveira, K., Castro, J., España, S., and Pastor, O. (2013). Multi-level autonomic business process management. In *International Conference on Enterprise, Business-Process and Information Systems Modeling*, pages 184–198, Valencia, Spain. Springer.

Oukharijane, J., Ben Said, I., Chaâbane, M. A., Andonoff, E., and Bouaziz, R. (2019). Towards a New Adaptation Engine for Self-Adaptation of BPMN Processes Instances. In *14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, pages 218–225, Heraklion, Crete, Greece.

Oukharijane, J., Chaabâne, M. A., Ben Said, I., Andonoff, E., and Bouaziz, R. (2020). A Hybrid approach based on Reuse Techniques for Autonomic Adaptation of Business Processes. In *International Conference on System and Software Reuses*, pages 1–17.

Oukharijane, J., Chaabâne, M. A., Ben Said, I., Andonoff, E., and Bouaziz, R. (2021). Self-adaptive business processes: a hybrid approach for the resolution of adaptation needs. *Innovations in Systems and Software Engineering*, pages 1–23.

Rinderle, S., Reichert, M., and Dadam, P. (2004). Correctness criteria for dynamic changes in workflow systems-a survey. *Data & Knowledge Engineering*, 50(1):9–34.

Rosemann, M., Recker, J., and Flender, C. (2008). Contextualisation of business processes. *International Journal of Business Process Integration and Management*, 3(1):47–60.

Seiger, R., Huber, S., Heisig, P., and Assmann, U. (2019). Enabling Self-adaptive Workflows for Cyber-physical Systems. *Software & Systems Modeling*, 18(2):1117–1134.