Federation University ResearchOnline

https://researchonline.federation.edu.au

Copyright Notice

This article is © Emerald Group Publishing and permission has been granted for this version to appear here. Emerald does not grant permission for this article to be further copied/distributed or hosted elsewhere without the express permission from Emerald Group Publishing Limited.

The final published version of this article can be found at:
https://doi.org/10.1108/BPMJ-07-2020-0331

See this record in Federation ResearchOnline at:
https://researchonline.federation.edu.au/vital/access/manager/Index

ENABLING SITUATIONAL AWARENESS OF BUSINESS PROCESSES

Xiaohui Zhao[1], SiraYongchare[2], and Namwook Cho[3]

[1]School of Engineering, Information Technology, and Physical Sciences, Federation University, Australia
x.zhao@federation.edu.au

[2]Department of IT and Software Engineering, Auckland University of Technology, New Zealand
sira.yongchareon@aut.ac.nz

[3]Department of Industrial & Information Systems Engineering, Seoul National University of Science and Technology, South Korea
nwcho@seoultech.ac.kr

ABSTRACT

The advancement of sensing technologies promises various smart applications. Once integrated into business process management, it will empower business process with the awareness of run-time dynamics like environmental conditions, customer behaviours, object movements, etc. With such awareness, business processes can adapt their responses to the changing conditions, and thereby evolve to be more intelligent and adaptive. Undoubtedly, such business processes will improve customer experience, enhance the reliability of service delivery and lower the operational cost for a more competitive and sustainable business. On the way to exploring such situation aware business process management, this paper proposes a conceptual method of depicting the context of a business process and the related mechanism of perceiving contextual dynamics. The applicability of the method and the improvement to process performance are illustrated and evaluated through process simulations.

KEYWORDS

business process modelling; context awareness; event processing

1. INTRODUCTION

Recent years have witnessed the rapid growth of sensing technologies, such as Radio Frequency Identification (RFID) and sensor network. These technologies have started changing our everyday lives and business operations with smart appliances, building automation, self-adapting logistics chain, and a lot more innovative applications (Dey, 2001, Hong et al., 2009). For business leaders, the advancement of these technologies enables the real-time visibility into environmental conditions (e.g., temperature and humidity), customer behaviours (e.g., browsing time for specific product and the number of interested users), object movements (e.g., a pallet of goods going through the entrance of a warehouse and the location of a specific product), and many other situational dynamics, to understand and control the business on day-to-day basis. From the perspective of business process management, with such capability business processes can perceive customer related or item level changes at run time, and also intelligently respond to these changes to pursue optimal performance (Rosemann and Recker, 2006). As predicted by Gartner (2012) and other industry leaders (Sharp, 2018), the next generation of business process management (BPM) would integrate situational awareness and real-time business analytics together to realise intelligent business operations.

In spite to the high industry demand on integrating situational awareness into business process management, the current application of situational awareness is largely limited to very specific areas with rather simple functions, such as self-switching on/off auto teller machines (ATMs), building automation (turning on/off air conditioners according to room temperature and residents' work time), and product tracing in distribution centres (Boukadi et al., 2009). These sporadic applications fail to connect the perceived changes together to get a full spectrum of the running situation and feed it to backend decision making system to get the best response to the changes. From a BPM perspective, there is still a long way in front before situation sensitive and intelligent business process management can be fully realised. Although some initial efforts have been done with a focus on the technical architecture of event processing for business processes (such as encapsulated event stream processing unit (SPU) (Hoffman, 1984), and the extension to the current process modelling languages, e.g., BPEL and BPMN, to enable service invocation by events (B.Juric, 2010, Yousfi et al., 2016). Tremendous efforts on architecting the system structure, modelling contexts and maintaining the run-time context model, formalising the context sensing and adapting algorithms, etc., are still on large demand by such new generation BPM. Aiming at narrowing the gap, this paper has explored the fundamental supports to the aforementioned functions with a focus on context modelling and situation perception. On the basis of our previous work on RFID enabled applications (Zhao et al., 2010, Zhao et al., 2012) and context–aware business process management (Zhao et al., 2018), the reported work has particularly contributed in the following areas:

- Formalise the context presentation for a business process with a focus on the rules and entities to support context perception;
- Propose a system architecture to illustrate the structure an d constitution of a supporting system for intelligent and situation aware business process management;
- Develop real-time event elicitation and interpretation mechanisms to operationalise the perception of contextual dynamics and real-time responses; and
- Evaluate the applicability of the proposed approaches and the performance improvement to business processes.

The rest of the paper is organised as follows: Section 2 reviews the work related to context awareness and the combination of it into business process management; Section 3 discusses a new method on business process context modelling and the mechanisms for interpreting contextual events; Section 4 demonstrates the applicability of the proposed method with a running example, and evaluates the performance improvements though process simulations; and finally the concluding remarks are given in Section 5 along with a discussion of our future work.

2. Related Work

The concept of context has been existing for quite a long time, yet it has ever been brought to business process modelling for the first time by Roseman et al. (2008). Their work included a high level definition of process context and a goal-oriented process modelling approach which helped to conceptualise, classify and integrate the process context. Later on Saidani et al. (2015). have proposed an approach to eliciting, categorising, adapting, and measuring context-related knowledge in order to identify process context and formalise its presentation. Ontology was deployed in their approach to make the model extensible to various business domains. In their ontology-based approach, the common elements that can affect business processes and the domain specific elements are presented at two different levels to cater for genericity and speciality of modelling, respectively. Besides, Mattos et al. (2014) have also developed a formal presentation to characterise the context of a business process activity in a specific domain using ontology formalism. This formal presentation consists of three layers for conceptual notions, process elements and domain classes, respectively, which are connected through the defined relations among them. This approach organises contextual entities in a hierarchical structure from abstract concept to specific classes, and thus it can adapt to different domains by changing and remapping the entities in the domain meta model. Aiming at making current process modelling languages capable of representing context-aware business processes, Yousfi et al. (2016) have extended BPMN specification with new elements of sensor/reader/collector tasks and corresponding events in their work on uBPMN. So far these works mainly look at how to define and model process contexts but not how to realise it to help business process to best respond to situational changes. The approaches proposed by these researchers also focus on identifying the types of entities and the relations among them involved in a context. Ahead of these work, more efforts are needed to incorporate such knowledge of process context into run-time business process management.

As a classical topic in business process management, process adaptation focuses on customising a process instance to make it applicable to a particular situation. In most existing work, process adaptation is conducted at deign time by defining conditional rules and constraints in the process model to allow different paths to be chosen at run time. For example, Heinrich et al. (2015) proposed a planning approach to automate the construction "exclusive choices" while designing a business process model. It makes use of a set of possible states in relation to the desired goal to consider multiple paths under a set of specific variable conditions. In this perspective, all default deviations and decisions have to be detected and modelled prior to execution. To accommodate the unpredictable situations at run time, some researchers explored the ways of supporting dynamic process adaptation at run time. Beest et al. (2014) proposed an approach for automating business process reconfiguration at runtime using AI planning techniques. They focus on the so-called "erroneous state," which is caused by interference in the process instance while running. On the basis of work system theory and theory of adaptive systems, Nunes et al. (2018) have broken down an unexpected situation into a number of known contextual elements and used them to automate the decision of re-planning the process flow. These works mainly target at how to plan the process adaptation to cater for the situational changes to the business environment. There is still a gap between perceiving the situational changes and responding to them, a seamless incorporation of sensing capability into process planning is highly sought after.

A promising way to such incorporation is through the event processing over sensor event streams, where situational dynamics can be perceived as events occurring in the context, as indicated by Janiesch et al. (2017) (please refer to Challenge 13. Bridging the gap between event-based and process-based systems in their work). In this direction, Schönig et al. (2020) have proposed an approach for Internet of Things (IoT)-aware business process execution. Though the approach does not focus on dynamic process adaptation, it does explore IoT data provenance, interaction between IoT data and business processes, and wearable user interface with context-specific IoT data provision. To attempt the context-aware business process adaption, Hu et al. (2013) have developed a rule-based method of generating activity sequences, on the basis of classic work on adaptive workflows by Reichert and Dadam (1998) and workflow views by Zhao and Liu (2010). These works showed different ways of attempting the run-time adaption to perceived situational changes. Further to these works, we have exerted to model the situational changes into a continuous event stream, and use event stream processing techniques to trace the dynamics of a process context and trigger process reactions to the perceived changes. Towards situation aware business process management, our work provides a full support including how to represent the context, how to maintain the context at run time, and how to interact with or intervene the context.

3. Facilitating Process Context Model

In business process management domain, the scope of context can span from technical details around a business process to strategic plans and global circumstance. Roseman et al. (2008) have defined four layers of business process context, viz., immediate layer, internal layer, external layer and environmental layer. From a high managerial level, a business process may experience turbulences by the introduction of new taxes, changes in national security or foreign policies, etc., which often happen in an unexpected manner. Organisations definitely expect to cope with such changes by adapting their business processes to the changing situation. Narrowing down to a more technical level, the resources and actors required by a business process may be altered, changed in capacity or performance, or even become unavailable in an unpredictable manner. The business partners in a business process, e.g., customers or suppliers, may also behave act unexpectedly, which further complicates the turbulences to the business process. Thus a situation aware business process adaptable to changes is highly sought after.

To support such business processes, business process management system (BPMS) needs to evolve to a context variant system whose behaviours depend on the process input as well as the context. Such as BPMS is expected to: (1) model the context of a business process including involved entities and the interdependencies among them; (2) capture the dynamics of a real business process context and record them in the context model; and (3) plan how the underlying business process should respond to the changed context. According to these three functions, this section is to present an approach to facilitating the situational awareness of business processes with a system architecture, a process context model and an event perceiving mechanism.

3.1 System Architecture

The proposed system architecture encompasses event handling, real-time decision making, context modelling and repository, plus traditional business process modelling and execution. This architecture has referenced the one proposed by Bolchini et al. (2011), which was done in the ubiquitous computing discipline for event handling, in order to inherit the strength in encapsulating event collection and easy connection to applications at upper levels. A lot of extensions and modifications are applied to the reference architecture to accommodate situation aware BPM.

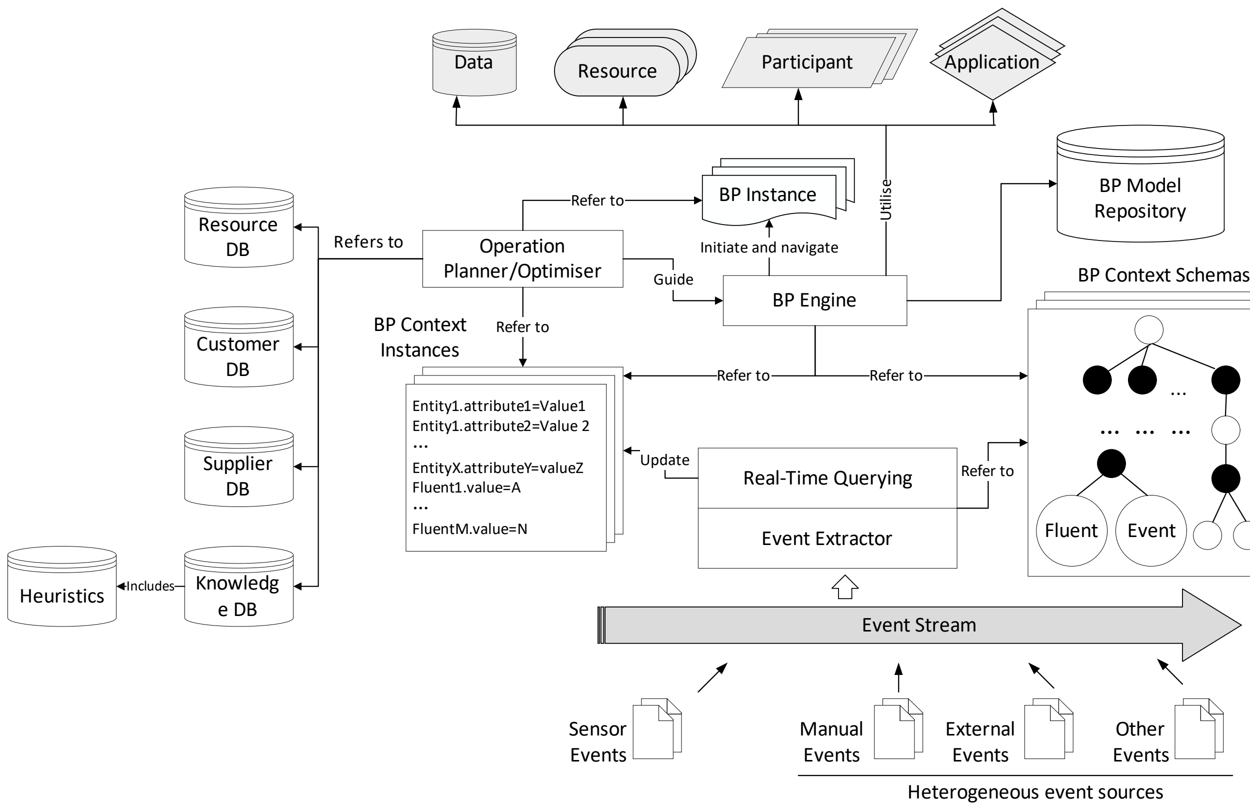


Figure 1. Architecture for context-aware BPM system.

Figure 1 illustrates the architecture of the supporting system for situation aware BPM. At build time a business process context is defined as a schema, and a schema includes the involved entities in the context and the interdependencies among these entities. As two pre-defined entities, fluent and event are used to specify how events influence entities. At run time, a context schema is instantiated with proper attribute values for the entities defined in the schema. The attribute values of the entities in a context instance reflect the process context at that time. According to this specific context, the Operation Planner/Optimiser guides the business process engine to best respond to the perceived situational changes by referring to the stored knowledge and heuristics. At the bottom, real-time sensor events and events from other sources are fed into a continuous event stream, and the system keeps running queries over the event stream to check if events match certain patterns, where different patterns indicate different business meanings. If a certain business meaning is identified from the event stream, the attribute values of related entities in a context instance will be updated to reflect the perceived change. Business process engine enforces the instantiation and execution of business processes, and navigates the business process instance to done by proper actors with proper resources.

3.2 Process Context Model

Our process context model is built upon ontology. In the field of computing, ontology refers to a set of concepts and categories in a subject area or domain that shows their properties and the relations between them. It is widely used to describe concepts, properties, constraints and relationships for the purpose of facilitating information sharing and reuse (Weber, 2003). Ontology-based models are specialised in distributed composition, partial validation, information richness and quality, incompleteness and ambiguity, formality and applicability, in comparison to other options for context modelling, such as simple key-value models, markup scheme models (e.g., Standard Generic Markup Language and User Agent Profile), graphical models (e.g., UML diagrams), and logic based models (Strang and Linnhoff-Popien, 2004, Bettini et al., 2010). Due to these merits, ontology is selected to represent the entities involved in a business process context and the semantics of the context. As the contexts of different business processes vary significantly, particularly if the business processes are from different domains, it is unrealistic to pursue a universal context across all business processes. Instead, we define a high-level ontology to generalise the model of business process context.

Definition (**Business Process Context**) An ontology for the context of a business process is defined as a 6-tuple (*C, A, HR, L, FC, FR*), where

— *C* is the set of concept. Two pre-defined concepts in *C* are 'fluent' and 'event'.
— *A* is the set of all attributes that belong to the concepts in set *C*.
— *HR* is the set of non-taxonomic relations,
— *L* is the set of terms (lexicals) that refer to concepts and relations,
— $FC = \{(l, c) | l \in L \text{ and } c \in HC\}$ maps the terms in *L* to the corresponding concepts,
— $FR = \{(l, r) | l \in L \text{ and } r \in HR\}$ maps the terms in *L* to the corresponding relations.

A *concept* can be an entity or an attribute. An *entity* in a business process context refers to something or someone that has interactions with or has influences to the business process. Such an entity is characterised by its *attributes*, for example, entity machine operator has attributes such as name, position in the organisation, responsibility, and available time.

Relations specify how concepts relate to each other. Two traditional relations "is a" and "is a part of" are applied to depict the taxonomy of entities and how entities combine together to form composite concepts. Customised relations can be added to better characterise the relationships among entities in a given domain, e.g., "places orders of" and "checks credits of" for a sales order handling process.

Terms further describe the relationships between entities. Typically, the rules on how these entities influence each other are specified by terms.

The structure of such an ontology defines the constitution of a business process context, and the attribute value changes of entities reflect the dynamics of the context. Due to the relations among entities, a change to one entity's attribute value may result in a chain of changes to more entities. This well reflects the complex nature of business process context.

3.3 Event Perceiving

In a senor deployed environment, situational changes are originally captured by sensors as sensor events. Though some attributes, such as current prices of products and customers' credit limits, still require manually updating, we can feed these manual changes as some event to the business process context. After combine these sensor events and manual events together, we can feed an event stream into the system as the raw input of situational information. According to above discussion, in order to perceive situational changes, the system is required to:

1. Read an event stream and elicit information out of it;
2. Interpret the business meaning from a series of events by recognising specific event patterns;
3. Integrate with the proposed ontology-based process context model seamlessly.

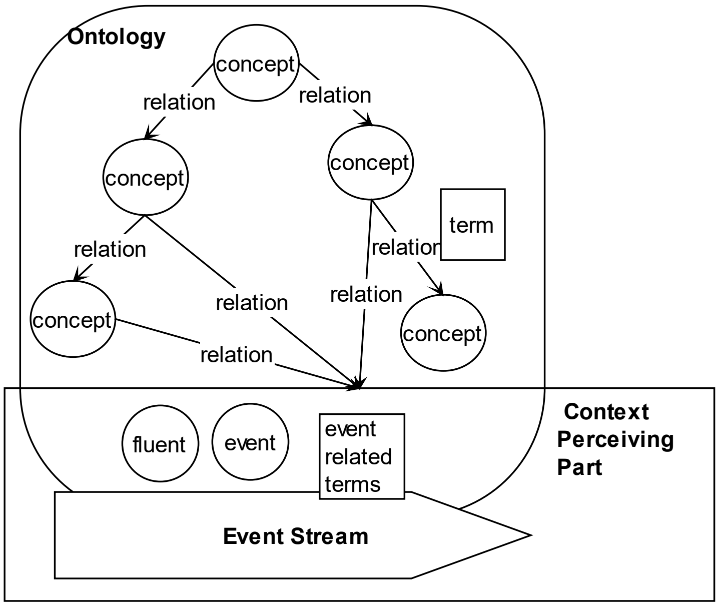


Figure 2. General ontology based process context model.

As shown in Figure 2, an ontology-based process context model consists of a set of concepts together with relations and terms. Special concept 'event' stays as a bottom node in the ontology, indicating events act as first line entities to bring in contextual information. When the input event stream is being received, events in the stream will be interpreted by related terms to trigger related attribute value changes. Such changes may occur in a cascaded manner along the relations defined among entities to escalate situational changes to upper entities. As the events keep coming in, attributes change their values accordingly to reflect the situational dynamics.

By nature sensor events in the event stream are often primitive events which only carry very basic information. For example, the event generated by typical RFID sensor includes nothing but ID of the sensor, ID of the observed RFID tag and the observation time. This information is certainly too simple to trigger any high-level situational changes or any responses by the business process, not to mention the accuracy rate of sensor events. For example, a single discrete event of identifying higher temperature does not necessarily mean the temperature is getting higher. The trend of temperature increase can only be confirmed after a series of such events are received and they consistently represent the temperature is getting higher. To find out such meaningful situational information from senor events, we need to continuously run queries over the event stream to check if a certain event pattern is matched. In addition, such patterns are often timed, and thus the queries need to scan both current and historical events rather than a fixed set of events. To support such queries, we base our event interpretation mechanism on event calculus.

Preliminaries of Event Calculus

Event calculus has been first presented by Robert Kowalski in 1986 and later on extended by Murray Shanahan and Rob Miller as a logical language for representing and reasoning about events and their effects (Shanahan, 1999). In event calculus, *fluents* are formalised by means of functions. The changes of the values of such fluents reflect the status of a situation. A separate predicate *HoldsAt* is used to tell which fluents hold (are true) at a given time point. Events are represented as terms. The effects of events are given using the predicates Initiates and Terminates. Some domain independent rules serve as a universal set of rules applicable to different application domains, which ensure event calculus work as intended. Table 1 lists the primary event calculus predicates.

Table 1. Event calculus predicates

Predicates	Explanation
Initiates(e, f, t)	Fluent f starts to hold (turns true) as the effect of the occurrence of event e at time t.
Terminates(e, f, t)	Fluent f ceases to hold (turns false) as the effect of the occurrence of event e at time t.
$\mathit{Initially}_P(f)$	Fluent f holds (is true) from the initial time point.
$\mathit{Initially}_N(f)$	Fluent f does not hold (is false) from the initial time point.
$t_1<t_2$	Time point t_1 is prior to time point t_2.
Happens(e, t)	Event e takes place at time t (Here we confine that e is a spontaneous event, which means the action delegated by event e occurs and finishes at the same time point t.)
HoldsAt(f, t)	Fluent f holds (is true) at time t.
Clipped(t_1, f, t_2)	Fluent f has been made false between time points t_1 and t_2.
Declipped(t_1, f, t_2)	Fluent f has been made true between time points t_1 and t_2.

Domain-independent Event Calculus (EC) Axioms. Event calculus formalises the correct evolution of the fluent via formulas that tell the values of each fluent after an arbitrary action has been performed. Event calculus solves the classic logic frame problem (see [15] for the details of frame problem) in a way that a fluent is true at time t is and only if it has been made true in the past and has not been made false in the meantime as shown by the following formula (EC3).

(EC1) $\mathit{HoldsAt}(f, t_2) \leftarrow \mathit{Happens}(e, t_1) \wedge \mathit{Initiates}(e, f, t_1) \wedge t_1<t_2 \wedge \neg\mathit{Clipped}(t_1, f, t_2)$;
This formula means that fluent f is true at time t_2 if:

1. An event e has taken place: *Happens*(e, t_1)
2. This took place in the past: $t_1<t_2$
3. This event has the fluent f as an effect: *Initiates*(e, f, t_1)
4. The fluent has not been made false in the meantime: $\neg$*Clipped*(t_1, f, t_2);

A similar formula (EC2) is used to formalise the opposite case in which a fluent is false at a given time.

(EC2) $\neg\mathit{HoldsAt}(f, t_2) \leftarrow \mathit{Happens}(e, t_1) \wedge \mathit{Terminates}(e, f, t_1) \wedge t_1< t_2 \wedge \neg\mathit{Declipped}(t_1, f, t_2)$;

The *Clipped* and *Declipped* predicates, stating that a fluent has been made false or true during an interval, can be axiomatised as follows:

(EC3) $\mathit{Clipped}(t_1, f, t_3) \leftrightarrow \exists e, t_2\ [\mathit{Happens}(e, t_2) \wedge t_1< t_2<t_3 \wedge \mathit{Terminates}(e, f, t_2)]$;

(EC4) $\mathit{Declipped}(t_1, f, t_3) \leftrightarrow \exists e, t_2\ [\mathit{Happens}(e, t_2) \wedge t_1< t_2< t_3 \wedge \mathit{Initiates}(e, f, t_2)]$;

In order to describe fluents' behaviours at the initial time, the following two axioms specify a general principle of persistence for fluents. Thus fluents change their values only via the occurrence of initiating and terminating actions.

(EC5) $\mathit{HoldsAt}(f, t) \leftarrow \mathit{InitiallyP}(f) \wedge \mathit{Clipped}(0, f, t)$;

(EC6) $\neg\mathit{HoldsAt}(f, t) \leftarrow \mathit{InitiallyN}(f) \wedge \neg\mathit{Declipped}(0, f, t)$.

The full set of the six domain independent axioms is represented as *EC*, where $EC=\bigwedge_{i=1}^{6} EC_i$.

Uniqueness-of-names (UNA) Axioms are particularly used to guarantee that each event is unique and thus there are no overlapping effects between events. For example, UNA[unloaded, loaded, broken] indicates the events unloaded ≠ loaded, loaded ≠ broken and unloaded ≠ broken. For simplicity, we do not introduce the details of the uniqueness-of-names axioms, but use symbol Ω to represent the conjunction of these UNA axioms.

Event Calculus based Model for Perceiving Contextual Dynamics

Based on our previous work on event processing (Zhao et al., 2012), we create a model to specify the rules for eliciting business meanings from a series of events and trigger the responses to the perceived changes on the basis of event calculus. A particular concept of scenario is defined to include the rules that are specially designed for a given business process context, and these rules will drive the business process to perceive and respond to the contextual changes, as shown in Figure 3. Concept environment is created to refer to an actual series of events with the initial settings of involved fluents which defines a specific running environment for business processes. A scenario works in an environment, and this delegates that a single business process or a group of business processes works or work in a dynamic environment. Queries are continuously running to check the values of fluents, and the business process(es) relies/rely on these to determine the changes of the context, and triggers will start the responding actions according to the query results.

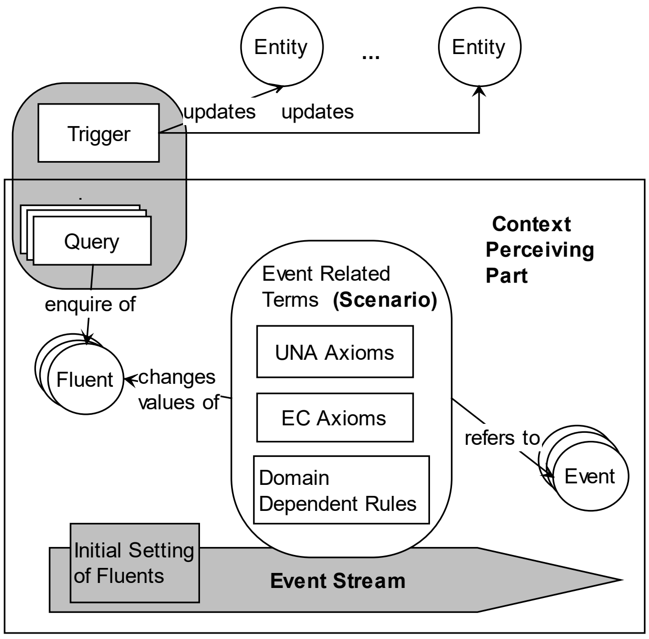


Figure 3. Context perceiving part.

Definition (**Event Occurrence**) Using predicate Happens an event occurrence is simply characterised as Happens(e, t), which indicates that event e occurs at time *t*.
Definition (**Domain-dependent Rule**) The domain-dependent rule is represented as a conjunction of expressions constituting the aforementioned fluents and predicates. Syntactically, a domain-dependent rule *r* can be defined as $P \leftarrow \bigvee_{i=0}^{n}[(\bigwedge_{j=0}^{m} exp_{ij}) \vee exp_i]$, where
— $P \in \{Initiates(e, f, t), Terminates(e, f, t), HoldsAt(f, t)\} \cup$ {domain-specific predicates};
— exp_{ij} and $exp_i \in \{Happens(e, t), HoldsAt(f, t)\} \cup$ {domain-specific predicates}.
Definition (**Scenario**) A scenario *S* describes the logic how a business process perceives the context, and syntactically, *S* is defined as triple (*R*, *EC*, *Ω*) where
— *R* is the set of domain dependent rules;
— *EC* is the set of *EC* axioms;

— Ω is the set of UNA axioms.

Definition (**Environment**) An environment *env* denotes the execution environment for a scenario. Technically, it is characterised by the initial settings of the involved fluents and the actual event stream. Correspondingly an environment *env* is defined as tuple (I, Δ) where

— I is the set of the initial values of the fluents specified using predicates $Initially_N$ and $Initially_P$;

— Δ is a sequence of events.

Definition (**Query**) For scenario S in environment env a query q_t can evaluate the values of specific fluents at a given time point t. Syntactically, a query q_t is defined as $_{\leftarrow\Delta(t_0,t_1)\wedge I_{t0}}\rho_t$, where

— ρ_t is a Boolean statement for the query to evaluate, and it is defined as a conjunction of several *HoldsAt*(f, t) predicates, i.e., $\rho_t = \bigwedge_i HoldsAt(f_i, t)$.

— $\Delta(t_0, t)$ is the set of events that have occurred from time point t_0 to t in environment *env*;

— I_{t0} denotes the initial values of fluents defined in *env* at time t_0;

Definition (**Trigger**) When a query detects a contextual change (i.e., the value of the Boolean statement to evaluate is changed), a trigger tg starts the reaction to it by updating an entity's attribute value in the ontology. Syntactically, *tg* is defined as $|_{\leftarrow\Delta(t_0,t_1)\wedge I_{t0}}\rho_t| \Rightarrow$ *update*(*e.att, x*), where

— symbol "| |" denotes the boolean result of the query within the bars;

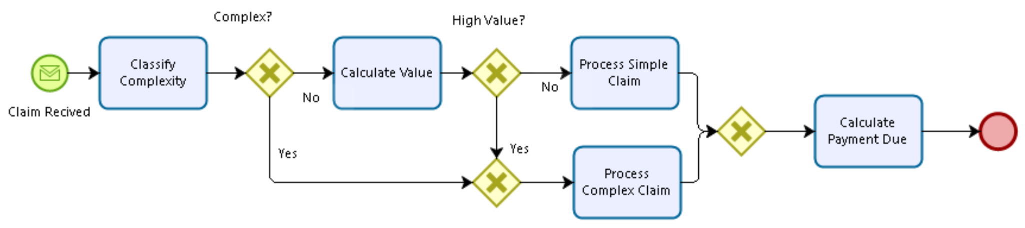
— *update*(*e.att, x*) denotes the action of updating the value of attribute *att* of entity *e* to *x*, if the query returns true.

4. Evaluation And Discussion

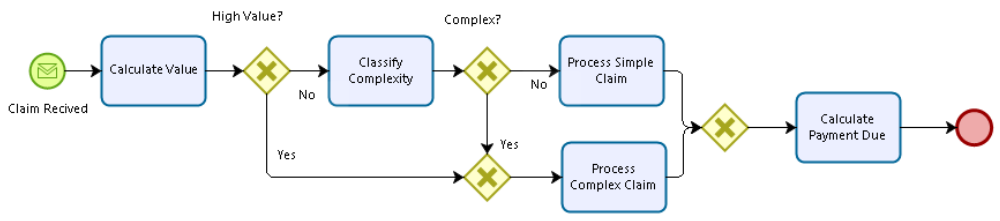
This section is to test the applicability of the proposed approach and evaluate the performance improvement by the situational awareness through business process simulation.

4.1 Setup of the Evaluation Scenario

A simplified insurance claim handling process is used to for the simulation. An instance of this process is created to handle the claim once a claim is received. As shown in Figure 4, this claim handling process can be done in two ways. In the first way as shown in Figure 4 (a), the process will classify if the received claim is a complex case or not, if not the process will continue to check if the process is of high value or not. Either if it is a complex claim or it is of high value, this claim will be handled as a complex claim by a senior staff, otherwise it will be handled as a simple one by either a junior or senior staff. Finally the payment due will be calculated. Except task "Process Complex Claim", all tasks in the process can be done by either a junior or senior staff. In the other way as shown in Figure 4 (b), the process will check the value of the claim before the complexity. The activity handling times and required resources are given in Table 2, where we can see the time for processing a complex claim is much longer than doing a simple one.



(a)



(b)

Figure 4. Insurance claim handling processes

Table 2. Process settings

Activity Name	**Activity Time (minutes)**	**Require Resources**
Classify Complexity	*N*(10, 9)	One junior staff or one senior staff
Calculate Value	*N*(10, 9)	One junior staff or one senior staff
Process Simple Claim	*N*(40, 225)	One junior staff or one senior staff
Process Complex Claim	*N*(200, 2500)	One senior staff
Calculate Payment Due	*N*(10, 9)	One junior staff or one senior staff

*$N(\mu, \sigma^2)$ indicates a normal distribution where μ is the mean, and σ is the deviation.

Though the two process models look very similar to each other, their performances are quite different in different circumstances. In the case that the incoming claims are predominated by simple claims, process model (a) will be more efficient because it starts handling complex claims earlier and thereby increases the chance of concurrent process instances. Accordingly, in the case that the incoming claims are predominated by low value claims, process model (b) will bring more efficiency.

The process context consists of senior staff, junior staff, and claim itself. A claim has two important attributes, "claim value" and "complexity", where the former can have value of "high" or "low" and the latter can have value of "simple" or "complex" to indicate the type of the claim.

The situational awareness is set up to continuously check the number of received claims of a certain type in a given period of time. If the number goes beyond a threshold, the process will determine that a particular type of claims is dominating and adapt to it be switching between the two process models to pursue the best efficiency.

In terms of events, if a complex claim is identified by a business process, event "*complexClaim*" will be thrown to the event stream. And an event "*highValueClaim*" is thrown out is a high-value claim is identified. Once the business process management system decides to switch the process model from type (a) to type (b), an event "*aToB*" will be thrown, and so is event "*bToA*" if the process model will switch from type (b) to type (a).

complexClaim –	A complex claim is identified;
highValueClaim –	A high-value claim is identified;
aToB –	Process switches from model (a) to model (b);
bToA –	Process switches from model (b) to model (a).

The following fluents are created to characterise the status of the fault checking.

A1, A2 –	The recommendation for process model switch (model (a) to (b), or (b) to (a), respectively) has been sent;
S1, S2 –	The first *N*1 or *N*2 claims have not been checked yet for the purpose of monitoring complex claims or high-value claims, respectively;
F1, F2 –	The first *N*1-1 or *N*2-1 claims have been checked already for the purpose of monitoring complex claims or high-value claims, respectively;
IsOfTypeA –	The current process model is of type (a);
IsOfTypeB –	The current process model is of type (b).

In addition to these fluents, the following variables are used.

N1, N2 –	Integer variables, to specify the threshold for triggering recommendation *A*1 and *A*2, respectively;
T –	Temporal variable, to specify the threshold time period;
Num1, num2 –	Integer variables, to record the number of complex claims and high-value claims, respectively;
queue1, queue2 –	Queues to store the times of finding complex claims and high-value claims, respectively;
tx, ty –	Time typed variables.

The corresponding scenario *S*=(*R, EC, Ω*) constitutes axioms EC and Ω, and the domain-dependent rule set *R*, where *R* comprises the following rules:

(R1) $Terminates(A1, t) \wedge Terminates(S1, t) \wedge Terminates(F1, t) \wedge queue1.clear() \wedge Initiates(IsOfTypeB, t) \wedge Terminates(IsOfTypeA, t) \wedge num1{=}0 \leftarrow Happens(aToB, t)$;

(R2) $queue1.add(t) \wedge tx{=}t \wedge num1{+}{+} \wedge Initiates(S1, t) \leftarrow Happens(complexClaim, t) \wedge \neg HoldsAt(S1, t) \wedge \neg HoldsAt(F1, t)$;

(R3) $queue1.add(t) \wedge num1{+}{+} \leftarrow Happens(complexClaim, t) \wedge HoldsAt(S1, t) \wedge \neg HoldsAt(F1, t)$;

(R4) $queue1.add(t) \wedge Initiates(F1, t) \leftarrow Happens(complexClaim, t) \wedge HoldsAt(S1, t) \wedge \neg HoldsAt(F1, t) \wedge num1{=}N1{-}2$;

(R5) $Initiates(A1, t) \wedge queue1.pop() \wedge queue1.add(t) \wedge tx{=}queue1.pop() \leftarrow Happens(complexClaim, t) \wedge HoldsAt(F1, t) \wedge \neg HoldsAt(A1, t) \wedge HoldsAt(IsOfTypeA, t) \wedge (t{-}tx{<=}T)$;

(R6) $queue1.pop() \wedge queue1.add(t) \wedge tx{=}queue1.pop() \leftarrow Happens(complexClaim, t) \wedge HoldsAt(F1, t) \wedge (t{-}tx{>}T)$.

(R7) $Terminates(A2, t) \wedge Terminates(S2, t) \wedge Terminates(F2, t) \wedge queue2.clear() \wedge Initiates(IsOfTypeA, t) \wedge Terminates(IsOfTypeB, t) \wedge num2=0 \leftarrow Happens(bToA, t)$;
(R8) $queue2.add(t) \wedge ty=t \wedge num2{+}{+} \wedge Initiates(S2, t) \leftarrow Happens(highValueClaim, t) \wedge \neg HoldsAt(S2, t) \wedge \neg HoldsAt(F2, t)$;
(R9) $queue2.add(t) \wedge num2{+}{+} \leftarrow Happens(highValueClaim, t) \wedge HoldsAt(S2, t) \wedge \neg HoldsAt(F2, t)$;
(R10) $queue2.add(t) \wedge Initiates(F2, t) \leftarrow Happens(highValueClaim, t) \wedge HoldsAt(S2, t) \wedge \neg HoldsAt(F2, t) \wedge num2=N2\text{-}2$;
(R11) $Initiates(A2, t) \wedge queue2.pop() \wedge queue2.add(t) \wedge ty=queue2.pop() \leftarrow Happens(highValueClaim, t) \wedge HoldsAt(F2, t) \wedge \neg HoldsAt(A2, t) \wedge HoldsAt(IsOfTypeB, t) \wedge (t\text{-}ty<=T)$;
(R12) $queue2.pop() \wedge queue2.add(t) \wedge ty=queue2.pop() \leftarrow Happens(highValueClaim, t) \wedge HoldsAt(F2, t) \wedge (t\text{-}ty>T)$.

Here, (R1) resets the fluents and variables when the process model is switched from model (a) to model (b); (R2) shows how the first identified complex claim is recorded and (R3) shows how to record the following identified complex claims before the second last to the threshold; when the second last complex claim to the threshold is found, (R4) turns fluent *F*1 to be true; When next complex claim is found, if the *N*1 complex claims are found within the given period *T* (R5) turns on the recommendation for process model switch, if not it will update the times stored in the queue to continue monitoring the event stream as shown in (R6). (R7-12) do the same job for high-value claims.

4.2Simulation and Results

With the awareness to the frequency of certain types of received claims, the business process can proactively adapt to it by switching between the two process models at run time to pursue the best efficiency. In order to evaluate the performance gain by such awareness, we have conducted process simulations to compare the situation aware business process and traditional static business process in terms of total cost, average process time and resource utilisation.

The modelled rules are implemented in Python. The queries are evaluated using IBM Discrete Event Calculus Reasoner, and relsat 2.02 is selected as the propositional satisfiability solver. The event input stream is simulated by a large event log file, where the claims arrival rate is of a Poisson distribution with mean interval value of 30 minutes. The thresholds for complex claims and high-value claims are set 5 (i.e., *N*1=*N*2=5), and the threshold time period (i.e., *T*) is set 750 minutes. To better visualise the performance improvement to the business process, we run a series of simulations of the business process in BizAgi Modeler 3.1.1 (32-bit) software. Table 3 lists the available resources and their costs.

Table 3. Resource settings

Resource Role	**Quantity**	**Cost (per hour)**
Junior Staff	2	\$30
Senior Staff	3	\$50

The test case for the simulation includes 2000 claims, and the percentages of different types are listed for the first 1000 cohort and the last 1000, respectively, in Table 4. The situation-aware business process first starts with process model (a), and when it starts handling the last 1000 claims, it can sense the changes of claim constitution, and then switch from process model (a) to model (b).

Table 4. Test Case of Claims

Claim	**Percentages of different claim types**			
	High-value and complex	**Low-value and complex**	**High-value and simple**	**Low-value and simple**
First 1000 claims	1%	6%	27%	66%
Last 1000 claims	1%	15%	4%	80%

The 2000 claims have been used to test process model (a), model (b) and the situation-aware process, separately. When a new claim arrives, the process may create a new instance to handle, subject to resource availability. Thus we collect the execution times of all these process instances to calculate total time and average time per claim, and similarly work out the cost per claim. Figure 5 gives a screenshot of the running simulation.

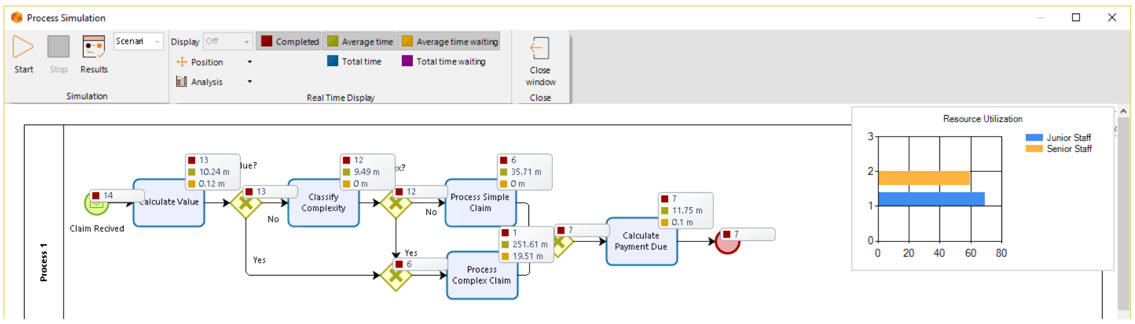
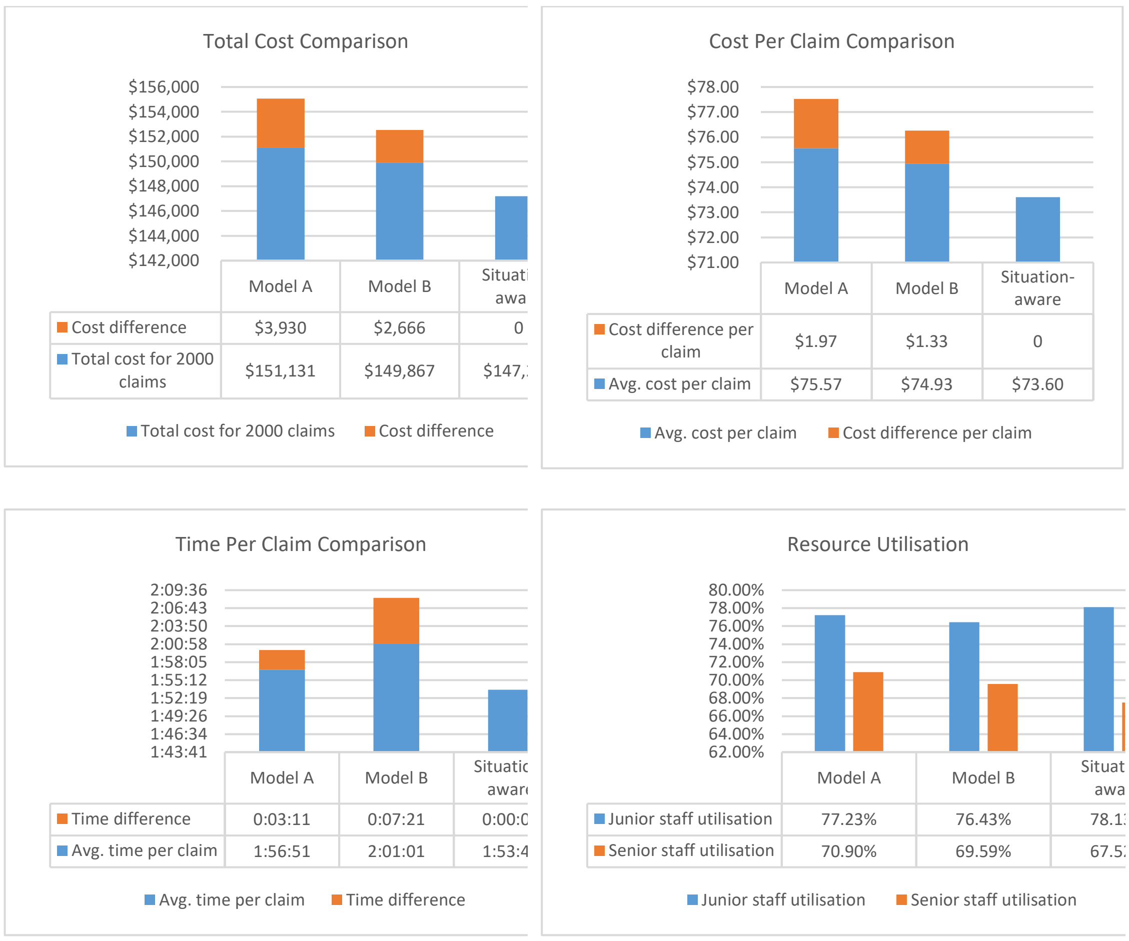


Figure 5. Simulation Screenshot

Figure 6 lists the comparisons among the three process models in terms of different measures. The orange blocks in the graphs on the first row indicate the cost difference against the situation-aware model. From these two graphs, we can easily see the cost efficiency from situational awareness. Compared to process model (a) and (b), the situation-aware process can save \$1.97 and \$1.33 per claim, respectively. The saving becomes considerable when the number of claims goes up, like saving \$3930 and \$2666 for the 2000 claims against process model (a) and (b). In regard to average process time, process model (a) and (b) have the average process time of 1h 56m 51s and 2h 01m 01s, respectively, which are 3m 11s and 7m 21s longer than the situation-aware process. The saving in time becomes considerable when we look at thousands of claims. The resource utilisation comparison shows the situation-aware process owns relatively higher utilisation of junior staff and lower utilisation of senior staff. This indicates the situation-aware process is better at scheduling resources to increase the utilisation of cheap resources while lower it of expensive resources. This also justifies the high cost efficiency of situation-aware process. An interesting finding here is the situation-aware process gains efficiency from both time and resources, and this can be explained as the situation-aware process tends to do the distinct split of claims first to enable the early handling of complex claims, which contributes to the potential parallelism of running process instances.



	Model A	Model B	Situati awa
🟧 Cost difference	$3,930	$2,666	0
🟦 Total cost for 2000 claims	$151,131	$149,867	$147,

	Model A	Model B	Situation-aware
🟧 Cost difference per claim	$1.97	$1.33	0
🟦 Avg. cost per claim	$75.57	$74.93	$73.60

	Model A	Model B	Situatio awar
🟧 Time difference	0:03:11	0:07:21	0:00:0
🟦 Avg. time per claim	1:56:51	2:01:01	1:53:4

	Model A	Model B	Situat awa
🟦 Junior staff utilisation	77.23%	76.43%	78.1
🟧 Senior staff utilisation	70.90%	69.59%	67.5

Figure 6. Process simulation results

Though the process simulation is by nature with some limitations and assumptions on activity time estimation, resource availability, claim arrival rate, etc., the results indicatively reflect the improvement to business efficiency in terms of time, cost and resource use. This evidences the effectiveness of situation-aware business process management.

5. Conclusion and Future Work

Aiming at integrating situational awareness into business process management, the reported work contributed to modelling process context, perceiving contextual changes and adapting business process to such changes. In particular, an ontology based process context model has been proposed to formally present the context constitution and the inter-relations within it. On the basis of event calculus a formal mechanism on querying contextual event streams and recognising event patterns is discussed to operationalise situational awareness for business process management. As a bridging work, these achievements have created a foundation for further development of business process automation and the integration of real-time data analysis and decision making, which can ultimately help realise business hyperautomation.

The reported work certainly is with some limitations and spaces for improvement. For example, the ontology based process context model allows user defined entities and relations, which may result in that different people may develop different models for the same context. Besides, once a contextual change is perceived, it is often not clear how to best respond to the change. In future, we will work on incorporating real-time business analytics into situation-aware business process management. Such decision making is required to be conducted at near real time and be context-sensitive. This will better help organisations to seek the optimal business efficiency with highly intelligent and adaptive business processes.

REFERENCES

APPEL, S., KLEBER, P., FRISCHBIER, S., FREUDENREICH, T. & BUCHMANN, A. 2014. Modeling and Execution of Event Stream Processing in Business Processes. *Information Systems,* 46**,** 140-156.

B.JURIC, M. 2010. WSDL and BPEL extensions for Event Driven Architecture. *Information and Software Technology,* 52**,** 1023-1342.

BEEST, N. R. T. P. V., KALDELI, E., BULANOV, P., WORTMANN, J. C. & LAZOVIK, A. 2014. Automated runtime repair of business processes. *Information Systems,* 39**,** 45-79.

BETTINI, C., BRDICZKA, O., HENRICKSEN, K., INDULSKA, J., NICKLAS, D., RANGANATHAN, A. & RIBONI, D. 2010. A Survey of Context Modelling and Reasoning Techniques. *Pervasive and Mobile Computing* 6**,** 161-180.

BOLCHINI, C., ORSI, G., QUINTARELLI, E., SCHREIBER, F. A. & TANCA, L. 2011. Context Modelling and Context Awareness: steps forward in the Context-ADDICT project. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering,* 34**,** 47-54.

BOUKADI, K., CHAABANE, A. & VINCENT, L. 2009. Context-Aware Business Processes Modelling: Concepts, Issues and Framework. *13th IFAC Symposium on Information Control Problems in Manufacturing.* Moscow, Russia.

DEY, A. K. 2001. Understanding and Using Context. *Personal and Ubiquitous Computing,* 5**,** 4-7.

GARTNER 2012. Gartner Says Intelligent Business Operations Is the Next Step for BPM Progams.

HEINRICH, B., KLIERA, M. & ZIMMERMANN, S. 2015. Automated planning of process models: Design of a novel approach to construct exclusive choices. *Decision Support Systems,* 78**,** 1-14.

HOFFMAN, E. G. 1984. *Fundamentals of Tool Design*, Society of Manufacturing Engineers.

HONG, J., SUH, E. & KIM, S. 2009. Context-aware Systems: A Literature Review and Classification. *Expert Systems with Applications,* 36**,** 8509-8522.

HU, G., WU, B. & CHEN, J. 2013. Dynamic Adaptation of Business Process Based on Context Changes: A Rule-Oriented Approach. *International Conference on Service-Oriented Computing Workshops.* Berlin, Germany.

JANIESCH, C., KOSCHMIDER, A., MECELLA, M., WEBE, B., ANDREABURATTIN, CICCIO, C. D., GAL, A., KANNENGIESSER, U., MANNHARDT, F., JANMENDLING, OBERWEIS, A., REICHERT, M., RINDERLE-MA, S., SONG, W., SU, J., TORRES, V., WEIDLICH, M., WESKE, M. & ZHANG, L. 2017. The Internet-of-Things Meets Business Process Management: Mutual Benefits and Challenges.

MATTOS, T., SANTORO, F. M., REVOREDO, K. & NUNES, V. T. 2014. A Formal Representation for Context-aware Business Processes. *Computers in Industry,* 65**,** 1193-1214.

NUNES, V. T., SANTORO, F. M., WERNER, C. M. L. & RALHA, C. G. 2018. Real-Time Process Adaptation: A Context-Aware Replanning Approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* 48**,** 99-118.

REICHERT, M. & DADAM, P. 1998. ADEPTflex -Supporting Dynamic Changes of Workflows without Losing Control. *Journal of Intelligent Information Systems,* 10**,** 93-129.

ROSEMAN, M., RECKER, J. & FLENDER, C. 2008. Contextualisation of Business Process Management. *International Journal of Business Process Integration and Management,* 3**,** 47-60.

ROSEMANN, M. & RECKER, J. Context-aware Process Design: Exploring the Extrinsic Drivers for Process Flexibility. International Conference on Advanced Information Systems Engineering, 2006 Luxembourg. 149-158.

SAIDANI, O., ROLLAND, C. & NURCAN, S. 2015. Towards a generic context model for BPM. *Hawaii International Conference on System Sciences.* Hawaii.

SCHÖNIG, S., ACKERMANN, L., JABLONSKI, S. & ERMER, A. 2020. IoT meets BPM: a bidirectional communication architecture for IoT-aware process execution. *Software and Systems Modeling,* 19**,** 1443-1459.

SHANAHAN, M. 1999. The Event Calculus Explained. *Artificial Intelligence Today.* Springer.

SHARP, D. 2018. Situational awareness: Real-time decision making to improve business operations.

STRANG, T. & LINNHOFF-POPIEN, C. A Context Modeling Survey. Workshop on Advanced Context Modelling, Reasoning and Management, International Conference on Ubiquitous Computing, 2004 Nottingham, England.

WEBER, R. 2003. Conceptual Modelling and Ontology: Possibilities and Pitfalls. *Journal of Database Management,* 14**,** 1-20.

YOUSFI, A., BAUER, C., SAIDI, R. & DEY, A. K. 2016. uBPMN: A BPMN extension for modeling ubiquitous business processes. *Information and Software Technology,* 74**,** 55-68.

ZHAO, X. & LIU, C. 2010. Steering Dynamic Collaborations between Business Processes. *IEEE Transactions on Systems, Man and Cybernetics: Part A,* 40**,** 743-757.

ZHAO, X., LIU, C. & LIN, T. 2010. Incorporating Business Process Management into RFID-enabled Application Systems. *Business Process Management Journal,* 16**,** 932-953.

ZHAO, X., LIU, C. & LIN, T. 2012. Incorporating Business Logics into RFID-enabled Applications. *Information Processing & Management,* 48**,** 47-62.

ZHAO, X., YONGCHAREON, S., SHEN, J., CHO, N. & DEWAN, S. 2018. Enabling Intelligent Business Processes with Context Awareness. *IEEE International Conference on Services Computing.* San Francisco, US.