


On Modelling Process Aspects With Deontic Event-Calculus

Mustafa Hashmi, La Trobe University, Australia*

 <https://orcid.org/0000-0002-6376-082X>

ABSTRACT

Intuitive and faithful modelling of the compliance requirements about the process aspects is a prerequisite for automated compliance checking. Several formalisms with varying degrees of expressiveness for modelling compliance requirements have been reported in the literature. Deontic event-calculus (DEC) is a normative variant of event-calculus (EC) formalism with predicates to modelled normative requirements. However, currently, DEC does not support capturing normative requirements about the process aspects. In this paper, the authors extend DEC with new deontic predicates to model process aspects of data, time, control flow, and resources. The extended deontic predicates enable DEC to intuitively represent the compliance requirements relevant to aspects of a business process. In addition, the authors report the complexity evaluation of the extended deontic predicates using well-known Halstead's complexity metrics. Evaluation result demonstrates that the complexity of modelling the compliance rules with DEC predicates is significantly lower even when the complexity of the standard EC is exponential.

KEYWORDS

Business Process Aspects, Business Process Management, Business Processes, Compliance Requirements, Regulatory Norms

1. INTRODUCTION

Business process compliance provides a means to ensure business practice and processes are aligned with the set of norms that stem from a comprehensive set of rules and legislations related to it. It is a core ingredient of any enterprise which provide a high-level overview of the state-of-the-affairs about how well the enterprise is functioning. It is used to ensure that business processes behave according to the conditions prescribed by the norms, where these conditions may be relevant to one or more aspects of a business process. Any divergent/misbehaviour of the business process may lead to non-compliance, which may subsequently result in severe penalties (Hashmi et al., 2016).

In general, a business process can be understood as a set of logically ordered activities aiming to achieve a specific goal. An aspect of a business process can be understood as the general characteristics (or properties) that define the (logical) structure of the process and/or constraints that are attached to it and can be broadly classified into four different types according to their nature, namely: (i) *control-flow*, (ii) *data*, (iii) *resources*, and (iv) *temporal (or time)* perspectives. The control-flow aspect is used to define the execution order as well as conditions and constraints of executing such

DOI: 10.4018/IJSSMET.297498

*Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

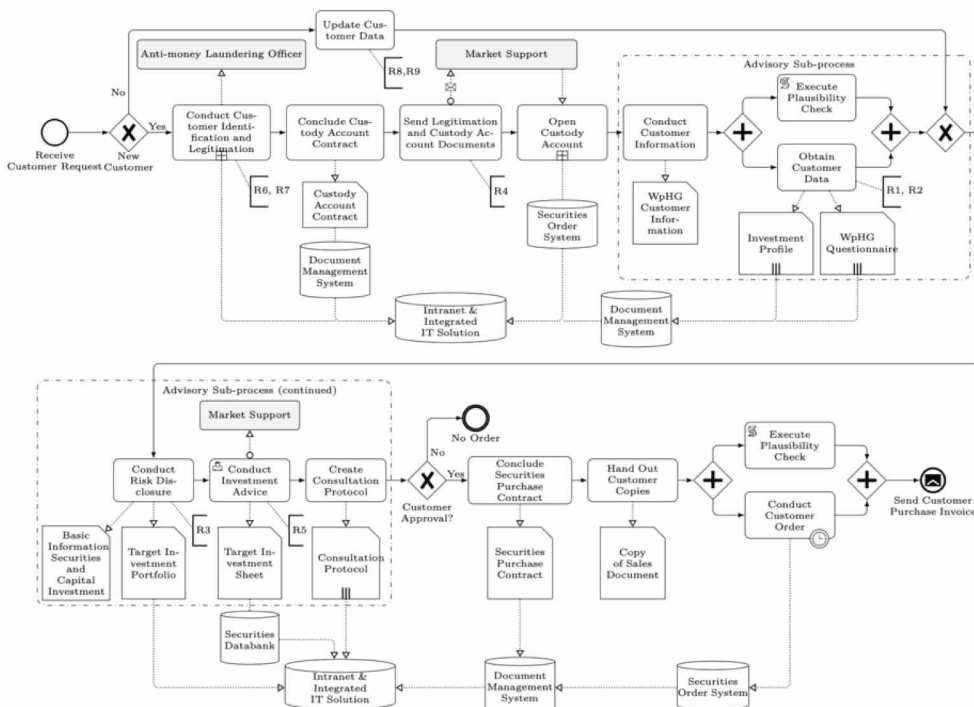
tasks; whereas the resources and data aspects describe the entities (and equipment) and data that are needed in the tasks, respectively. Lastly, the temporal aspect refers to the temporal requirements that are imposed to the enforcement, enactment and retract of a task.

Figure 1 illustrates an example of security purchase and advice business process that is subject to the compliance requirements from various German banking regulations, as summarized in Table 1. The process starts with an investment advice request from a client. If the client is a new customer, the investment consultant will then perform preliminary identification, lodgement and legitimation checks before providing any advice to the client. However, if the client is an existing one, his details will be updated, and a custody account will be set up for receiving securities transactions. The advisory sub-process is initiated by the acquisition of new/updated customer information and ends once the security advice is completed. If the customer accepts and approve the advice, then securities purchase contract will be prepared, and the order is recorded. Finally, an invoice will be sent to the customer to conclude the purchase and/or advice.

As can be seen from Table 1, the first two rules, R1 and R2, are related to the ordering (control-flow) of activities; R3 is related to the resources (the two brochures) being received by the customer, R5 prescribes the requirement for the competence of the consultant, and R6 stipulates a principle of handling suspected money laundering cases, etc.

Over the years, different formalisms, such as Linear Temporal Logic (LTL) (Pnueli, 1977), LTL-FO+ (Halle & Villemare, 2008), Defeasible Logic (DL) (Antoniou et al., 2000), Deontic Logic (DL) (von Wright, 1951), Deontic Logic of violations (Governatori & Rotolo, 2006), π -Logic (Abouzaid & Mullins, 2008), Event-Condition-Action (ECA) (Berndtsson & Mellin, 2009), Computational Tree Logic (CTL) and its variant CTL* (Vardi, 2001) etc., have been reported in the literature to model compliance requirements in the verification process. However, so far, existing approaches focus mainly on the verification of business rules at the process level, little has been done in modelling

Figure 1. Security purchase and advice business process (Adopted from (Zasada & Bui, 2018))



and verifying different aspects that are stipulated in the tasks. Deontic Event Calculus (hereafter, DEC) (Hashmi et al., 2014) is a deontic variant of Event Calculus (EC) (Kowalski & Sergot, 1986). It is a logical formalism that is expressive enough to intuitively represent different types of (deontic) modalities that appear in legal norms (Hashmi et al., 2016). Hence, in this paper, we extend DEC with new predicates to cater the needs of providing support in process aspects modelling. In addition, we also investigate the complexity of the newly introduced predicates.

The structure of the paper as follows: next we discuss related studies conducted in this space in Section 2, following which we tersely revisit the features of DEC in Section 3. In Section 4, we introduce the new predicates and events, and demonstrate how to intuitively model compliance requirements pertaining business process aspects in Section 5. The complexity analysis is presented in Section 6 before concluding the paper with final remarks and pointers for future work in Section 7.

This paper assumes the reader is familiar with EC (Kowalski & Sergot, 1986). All necessary basic notions and notations that appears without definitions are found in (Hashmi et al., 2014; Kowalski & Sergot, 1986).

2. RELATED WORK

Lohrmann and Reichert (2010) has considered process aspects from an economic perspective and identified 11 different types of economic resources, namely: (i) process instances value consumption, (ii) materials, (iii) raw materials, (iv) supplies, (v) labour, (vi) capital goods employment, (vii) use-bound depreciation, (viii) maintenance, (ix) capital goods provision, (x) time-bound depreciation, and (xi) capital employment, that needs to be considered when evaluating the quality of a business process, and has proposed a quality- related measures to this. In comparison to our work, their discussion covers more on the link between business objectives (of each activity) and their associated outcomes/benefits in terms of economic values, while ours is focusing more on the technical aspects of modelling compliance requirements from the process specifications perspective. Apart from this, as pointed out in the paper, whether their measure can be put into practice is still a subject that need to be investigated.

Similar to us, Turetken et al. (2012) divided the process aspects into four different pattern classes, namely: (i) order, (ii) occurrence, (iii) resource, and (iv) time, and has further been divided into 28 patterns. For each pattern, they have also developed a map- ping to linear temporal logic (LTL) or metric temporal logic (MTL) to automatically generate the corresponding statements that can be loaded into their pattern-based compliance framework for compliance verification and monitoring. However, their approach is pretty much concentrated on the temporal and control flow aspects of the business process, and little on the data aspect has been addressed.

Knuplesch et al. (2013), on the other hand, proposed the use of extended Compliance Rule Graph (eCRG), a visual notation, for compliance rules modelling. On the top of the four process aspects that we have addressed in this paper, their approach also covers interaction aspect that possibly appears in cross-organizational scenarios and had discussed how the information flow among different entities can be modeled. Each eCRG generated will then be converted into a corresponding first order logic (FOL) formula, which in turn will be evaluated over process traces. In our framework, we have decided not to separate the interaction aspect to a new class as the types of information flow can easily be modeled as fluent in DEC. Besides, the use of EC based formula also helps in simplifying the notations used in representing requirements related to the temporal aspect.

Montali and colleagues (Montali et al., 2013; Montali et al., 2014) have formalised Declare using EC and extended it with the notion of port for anchoring constraints to atomic and non-atomic activities. In their framework, a port is a tuple $P = \langle E, A, I, D, O \rangle$ where E is an event type that belongs to the lifecycle of activity A , D is the set of (internal) attributes implicitly associated to all activities, and I and O are the sets of input and output data identifiers of the port, respectively, and assuming that data corresponding to such identifiers are the same throughout the whole business process. Instead

of anchoring directly to the activities, data constraints related information will be anchored to the ports. Besides, such anchors can also be used to associate data conditions restricting the range of matching event occurrences. Even though extensions to other aspects seems possible, their analysis concentrated pretty much only on data aspect and little on other aspects of the business process has been addressed.

Pereira and Varajão (2017) have identified eight types of constraints related to the temporal aspect of business processes, namely: (i) activity duration, (ii) process duration, (iii) deadlines, (iv) minimum limit, (v) maximum limit, (vi) fixed date(s), (vii) waiting time, and (viii) negative information, and concluded that present business process management systems provide very limited support to model and manage temporal related information, which have to evolve in the future.

Kumar and Barton (2017) discussed a technique for capturing and validating the compliant behaviour of the temporal constraints on the workflow models. In their approach, the constraints are specified as structural and temporal conditions on the models. However, no support for modelling data and resources constraints is provided. Whereas in the context of a large project, Lam et al. (2020) discussed the process aspects as the compliance dimensions of roles, data, money, and quality but no clear indication on how the normative requirements about these dimensions modelled.

Joost and colleagues (Joost & Hans, 2020), on the other hand, used Event-Calculus to formalise logic for reasoning and modelling the effects of actions resulting from commitment for smart contracts. While commitments by nature may be normative, a commitment might be relevant to data, resource, etc—however, in their work have been discussed as exchange commitments and control commitments. No direct reference is provided associating the business process aspects of data, resources, etc., with the categories of commitments. Besides, they used a generic EC variant that cannot intuitively represent deontic modalities, constraints including contractual commitments (Governatori & Hashmi, 2015; Hashmi et al., 2014).

3. REVISITING DEONTIC EVENT CALCULUS

EC is a discrete event-based logical formalism that was developed to capture the behaviour of a sequence of event occurrences based on the notion of dynamic (mainly, time-varying) properties of the world, called fluents, which was initiated at some time instance and continuously hold until it is terminated, or interrupted, by some event (Miller & Shanahan, 1999, 2002)), and is a first-class object that can be quantified over and appear as arguments to predicates (Shanahan, 1999). Since its inception, EC is one of the widely used and studied logical formalism (Alexiev, 1995; Andrade-Lotero, 2006; Cervesato et al., 1997; Farrell et al., 2004; Sadri & Kowalski, 1995).

DEC, in this regard, is a normative extension of EC proposed by (Hashmi et al., 2014), which addresses the issues of EC's base predicates *Initiates* and *HoldsAt* that fail to capture the effects of legal norms when they enter into force. As mentioned in (Hashmi et al., 2016), the major difference between DEC and the standard EC is on the condition of initiation. That is, each (deontic) fluent Δ in DEC has its own specific set of triggering events, $\text{trigger}(\Delta, N)$, which distinguishably represents the situation that the fluent (Δ) enters into force with a delay N after the triggering event occurs. In what follows, we revisit some of the relevant predicates and events offered by DEC, as shown in Table 2¹.

Accordingly, business rules may prescribe conditions stipulating the deadlines until when the specific legal effects associated with the fluent must be achieved, or otherwise a violation is triggered. To capture such conditions, the deadline triggering event, $\text{deadline}(\Delta, \tau)$, is used, where Δ is a fluent and τ represents the time instance until when the fluent must be achieved. Correspondingly, the violation event, $\text{violation}(\Delta, \tau)$, is used to signify the fluent Δ has been violated at time instance τ .

Depending on the applicability conditions and contexts, these two events can be used in combination with the *Happens* predicate in EC to capture the deontic effects of the rules. For instance, $\text{Happens}(\text{trigger}(\Delta, N), \tau)$ can be used to signify the occurrence of the triggering event with a delay of N time instance, which cannot be achieved with the EC's *Initiates* predicate.

Table 1. Compliance requirements for security purchase and advice process

ID	Compliance Requirements
R1	The customer data must be received before the individual risk assessment can take place.
R2	Customer advisor must provide the two obligatory brochures, WpHG Customer Information and the Basic Information Securities and Capital Investment, to the customer.
R3	The customer advisor must ensure that the customer acknowledges the reception of the two brochures.
R4	After concluding the custody account contract, the customer legitimization and the account documents need to be sent to the market support.
R5	The investment advice needs to be conducted by a customer advisor with a securities competence of level C or above
R6	The customer identification and legitimization must be handled by the customer advisor, while suspected cases of money laundering must be checked by an anti-money laundering officer.
R7	Before concluding a custody account contract, the customer advisor needs to wait until the suspected case of anti-money laundering is resolved
R8	The customer information must be updated with every future customer contact.
R9	Stockless custody accounts are charged with a fee of 5 Euro per year. If the fee is not paid, the account is dissolved by market support. The account is reactivated by a new securities purchase.

The predicate $DHoldsAt(\Delta, \tau)$ signals the situation that the (deontic) fluent Δ is deontically hold at time instance τ . It is different from its EC counterpart $HoldsAt$ on the grounds of its initiation conditions. An obligation can enters into force in two situations: (a) the obligation enters into force when the triggering event occurs, or (b) the obligation enters into force with some delay after the triggering event occurs (Hashmi et al., 2014).

$DTerminates(E, \Delta, N, \tau)$, on the other hand, captures cases where the (deontic) fluent Δ is deontically terminated by the event E at time instance τ with some delay N , which can occur in either (i) *the fluent cannot be achieved before the deadline*, or (ii) *the fluent is violated and such violation cannot be repaired*, and can be captured using the two deontic events violation and deadline mentioned above.

Table 2. DEC events and predicates adopted from (Hashmi et al., 2014)

Item	Description
Deontic Event	
$trigger(\Delta, N)$	The fluent Δ has been triggered with a delay N .
$violation(\Delta, \tau)$	The fluent Δ has been violated at time τ .
$deadline(\Delta, \tau)$	The fluent Δ must be fulfilled before time τ , or a violation is triggered.
Deontic Predicate	
$DHoldsAt(\Delta, \tau)$	The fluent Δ deontically holds at time τ .
$DTerminates(E, \Delta, N, \tau)$	The event E deontically terminates the fluent Δ , with some delay N , at time τ . In some cases, the trigger for the fluent does not only trigger the initiation of the fluent but also its termination. This means we have to write expression with the following form: $DTerminates(trigger(\Delta, N), \Delta, N, \tau)$. Therefore, to ease readability, in such cases we will use the convention of dropping the repeated Δ and N from the arguments of $DTerminates$, using thus: $DTerminates(trigger(\Delta, N), \tau)$.

Below are some examples demonstrating how DEC can be used to model different types of obligations presented in (Hashmi et al., 2014). For instance, the following axioms that describe when a punctual obligation holds.

$$\begin{aligned} \text{D}\underline{\text{HoldsAt}}(\text{O}^{\tau_s} \varphi, \tau_s) &\leftarrow & (A1) \\ \exists \tau_t, N : &\text{Happens}(\text{trigger}(\text{O}^{\tau_s} \varphi, N), \tau_t) \\ &\wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned}$$

$$\begin{aligned} \text{D}\underline{\text{Terminates}}(\text{trigger}(\text{O}^{\tau_s} \varphi, N), \tau_t) &\leftarrow & (A2) \\ \exists \tau_t, N : &\text{Happens}(\text{trigger}(\text{O}^{\tau_s} \varphi, N), \tau_t) \\ &\wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned}$$

Here, the punctual obligation is represented as a deontic fluent $\text{O}^{\tau_s} \varphi$ where O^{τ_s} denotes an obligation to the fluent φ , τ_s is the time that the obligation enters into force, and N is an integer; whereas the triggering event $\text{trigger}(\text{O}^{\tau_s} \Delta, N)$ is used to initiate the obligation. Axiom A2, in the like manner, specifies that the obligation is terminated by the same event that triggered it. Hence, the obligation is terminated at the same time instance as it is initiated, which corresponds to the definition of punctual obligation.

Axiom A3, on the other hand, presents an achievement obligation.

$$\begin{aligned} \text{D}\underline{\text{HoldsAt}}(\text{O}^{\tau_s} \varphi, \tau_s) &\leftarrow & (A3) \\ \exists \tau_t, N : &\text{Happens}(\text{trigger}(\text{O}^{\tau_s} \varphi, N), \tau_t) \\ &\wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned}$$

For an achievement obligation, achieving the contents of obligation at least once is enough to comply. Generally, a deadline is attached to an achievement obligation until which the obligation must be fulfilled to avoid any violations.

$$\begin{aligned} \underline{\text{Happens}}(\text{violation}(\text{O}^{\tau_s} \varphi, \tau_v), \tau_v) &\leftarrow & (A4) \\ \text{D}\underline{\text{HoldsAt}}(\text{O}^{\tau_s} \varphi, \tau_e) \wedge \text{Happens}(\text{deadline}(\text{O}^{\tau_s} \varphi, \tau_e), \tau_e) \\ \wedge (\neg \text{Happens}(\varphi, \tau_e) \wedge \neg \text{HoldsAt}(\varphi, \tau_e)) \wedge \\ \underline{\text{ViolationTerminable}}(\text{O}^{\tau_s} \varphi) \wedge (\tau_v = \tau_e) \end{aligned}$$

Axiom A4 models the violation of an achievement obligation when the deadline of the obligation occurs but the contents of the obligation are not achieved within the deadline. Besides, depending on the obligation conditions, the violation of an achievement obligation may terminate² the obligation. The predicate ‘ViolationTerminable’ is a boolean switch that will be checked whether the obligation is terminable upon violation.

Maintenance obligation is a case of persistent obligation (Hashmi et al., 2016). A maintenance obligation generally holds in an interval, and the contents of maintenance obligation must be complied with for all the instants of the interval in which the obligation is in force. The intuition of a maintenance obligation is captured through persistence obligation:

$$\begin{aligned}
 \text{D HoldsAt}(\text{O}^{\tau_s} \varphi, \tau_s) \leftarrow & \quad (A5) \\
 \exists \tau_t, N : & \text{Happens}(\text{trigger}(\text{O}^{\tau_t} \varphi, N), \tau_k) \wedge \neg \text{DClipped}(\tau_s, \text{O}^{\tau_t} \varphi, \tau_k) \\
 & \wedge \text{DTerminates}(\text{trigger}(\text{O}^{\tau_t} \varphi, N), \tau_e) \\
 & \wedge (\tau_e > \tau_s) \wedge (\tau_s \leq \tau_k \leq \tau_e) \wedge N \geq 0
 \end{aligned}$$

A maintenance obligation will remain in force from τ_s and τ_e until no other relevant event occurs between the interval impacting the obligation.

$$\begin{aligned}
 \text{Happens}(\text{violation}(\text{O}^{\tau_s} \varphi, \tau_k), \tau_k) \leftarrow & \quad (A6) \\
 \text{D HoldsAt}(\text{O}^{\tau_s} \varphi, \tau_k) \wedge \neg \text{Happens}(\varphi, \tau_k) \\
 \wedge \neg \text{HoldsAt}(\varphi, \tau_k) \wedge (\tau_s \leq \tau_k) \wedge N \geq 0
 \end{aligned}$$

The violation is automatically triggered if the obligation contents is not achieved at any time instant in the time interval.

4. PROCESS ASPECT PREDICATES

As previously discussed, a business process is required to behave according to the conditions prescribed by the legal norms. Any divergence to this may lead to non-compliance where severe penalties can be imposed (Hashmi et al., 2016). In essence, these conditions can be relevant to one or more aspects of a business process, which can be broadly understood as the generic characteristics (or properties) that define the structure of the business process and constraints that are attached to it.

From a business process perspective, there are four distinct aspects that need to be considered, namely: control-flow, data, resources, and temporal (or time). The control flow aspect refers to the execution order, or interdependency, of tasks in a business process, and is affected by constructs such as choice, join, loops, synchronization between tasks (Kiepuszewski et al., 2003; Stroppi et al., 2015). Hence, in general, tasks in a business process can be executed in any order, including concurrent or parallel executions. For instance, a rule may prescribe that after concluding the custody contract, the customer legitimation and the account document needs to be sent to the market support simultaneously.

The data aspect describes what data (or information) is required for a task and looks to see how the business process can deliver, or generate, it from other task(s). In practice, the data can be manifested through different formats, such as different types of datasets, document corpus, images and tables, etc., and can be obtained from users directly, consolidated artefacts from datasets, or immediate results obtained from other tasks through message exchange.

Similar to the data aspect, the resources aspect defines the set of entities that are required by a task (Stroppi et al., 2015; Zur Muehlen, 2004). These entities can either be human resources (e.g., organization's employees (with or without special role), external contractors or collaborators, etc), computer hardware or other equipment, software applications or external computational services or environments, or a combination of both. As one can imagine, resources play a significant role in the performance of the whole business process, it is important that suitable resources should be assured and allocated to the tasks in advance.

The temporal (or time) aspect refers to the enforcement (or execution), enactment or retract of a task within a specific time frame. Accordingly, some tasks in a business process may have conditions that must be fulfilled in a determined time interval or by a given deadline, while other conditions may prescribe that some tasks are to be executed before or after the others. For example, a good

delivery contract may specify that an invoice must be sent within 30 days from the date it is issued, and that goods cannot be dispatched without a payment. In this case, it obligatory for the client to pay the invoice before the time of delivery. Hence, the temporal aspect prescribes the set of temporal constraints a process must comply with for the whole duration of its validity.

From the legal reasoning and automaton perspective, the formalism must be able to intuitively model legal norms and conditions attached to the specific rules. This typically includes the conditions attached to various aspects of a business process. The DEC can model legal norms but does not provide support for modeling the conditions related to control flow, data, time, and resources aspect of the business process. Hence, to extend the DEC support for modelling legal norms and associated conditions to process aspects, in this paper, we propose the use of $DHoldsAt$ predicate in DEC to capture the (deontic) fluents that are associated with different process aspects.

Before going into details, we first define the variable $\Delta[J] = \bigcup X \Delta[X]$ be the set of all process aspect related (deontic) fluents, and $\Delta[X]$ s.t. $X \in \langle C, D, R, T \rangle$ be the set of fluents which encapsulates the four aspects mentioned above. Hence, for the set $\Delta[J]$ to hold at a particular time instance τ , we have the following process aspect predicate³:

$$DHoldsAt(\Delta[J], \tau) \Leftrightarrow \langle DHoldsAt(\Delta[C], \tau) \cup DHoldsAt(\Delta[D], \tau) \\ \cup DHoldsAt(\Delta[R], \tau) \cup DHoldsAt(\Delta[T], \tau) \rangle$$

Notice however that, a compliance rule might prescribe constraints that are relevant to one or more aspects, meaning that some of the fluents above may be empty.

The next task is to associate the process aspects $\Delta[X]$ with the fluent for reasoning and verification of the normative constraints that include them.

Definition 1: $\Delta[X] = \langle \Delta, X, Idx \rangle$ is a triplet where Δ is the fluent attached to an obligation, and Idx maps $X \leftarrow Idx : \langle C, D, R, T \rangle$, associating the process aspects with the set of fluents X that encapsulate them.

5. FORMALIZING SECURITY PURCHASE PROCESS WITH DEC

Using the compliance rules as shown in Table 1, in this section, we are going to demonstrate how DEC can be used to model different process aspects of a business process.

As mentioned before, the rules R1 and R2 specify the control-flow conditions such that customer data must be collected before the start of customer's risk assessment and two brochures must be provided by the customer advisor to the customer, respectively. Accordingly, these conditions can be modelled using DEC as follows⁴:

$$\begin{aligned} DHoldsAt(O^{Ts} \text{ ConductRiskAssessment}_{[CA]} \tau_s) \leftarrow & \quad (A7) \\ \exists \tau_t, N : Happens(trigger(O^{Ts} \text{ ReceivedData}_{[CA, CD]} N), \tau_t) & \\ \wedge (\tau_s = \tau_t + N) \wedge N \geq 0 & \end{aligned}$$

Let's examine the details of above axioms. Suppose from R1 the event received data occurs for which from Axiom A1, we drive $(trigger(O^{Ts} \text{ ReceivedData}_{[CA, CD]} N), \tau_t)$ and from earlier defined process aspect predicate we obtain $DHoldsAt(O^{Ts} \text{ ConductRiskAssessment}_{[CA]} \tau_s)$, meaning that after receiving customer data (CD) it is obligatory for customer advisor (CA) to conduct the assessment. Notice that resource and data aspect are attached to fluent.

In contrast, R2 prescribes two fluents attached to the obligation, that is, providing basic security information (BSI) brochure and capital investment (CI). For simplicity reasons, we split the two conditions and model them as two separate rules as follow:

$$\text{D HoldsAt}(\text{O}^{\text{TS}} \text{ Provide}(\text{BSI Brochure})[\text{CA}, \text{D}], \tau_s) \leftarrow \quad (\text{A8})$$

$$\exists \tau_t, N : \text{Happens}(\text{trigger}(\text{O}^{\text{TS}} \text{ SecurityAdvise}[\text{CA}], \text{N}), \tau_t) \\ \wedge (\tau_s = \tau_t + N) \wedge N \geq 0$$

$$\text{D HoldsAt}(\text{O}^{\text{TS}} \text{ Provide}(\text{CI Brochure})[\text{CA}, \text{D}], \tau_s) \leftarrow \quad (\text{A9})$$

$$\exists \tau_t, N : \text{Happens}(\text{trigger}(\text{O}^{\text{TS}} \text{ SecurityAdvise}[\text{CA}], \text{N}), \tau_t) \\ \wedge (\tau_s = \tau_t + N) \wedge N \geq 0$$

For R2 suppose that the event security advise occurs at time τ_t for which from Axiom A1 we derive $(\text{trigger}(\text{O}^{\text{TS}} \text{ SecurityAdvise}_{[\text{CA}]}, \text{N}), \tau_t)$ and from the process aspect predicate we derive D HoldsAt predicate capturing the fluent provide brochures in the predicate that deontically holds at τ_t , and process aspect attached to the fluent.

$$\text{D HoldsAt}(\text{O}^{\text{TS}} \text{ ReceiveAcknowledgment}[\text{CA}, \text{D}], \tau_s) \leftarrow \quad (\text{A10})$$

$$\exists \tau_t, N : \text{Happens}(\text{trigger}(\text{O}^{\text{T}} \text{ SecurityAdvise}[\text{CA}], \text{N}), \tau_t) \\ \wedge \text{D HoldsAt}(\text{Provide}(\text{BSI Brochure})[\text{CA}, \text{D}], \tau) \\ \wedge \text{D HoldsAt}(\text{Provide}(\text{CI Brochure})[\text{CA}, \text{D}], \tau) \\ \wedge (\tau_s = \tau_t + N) \wedge N \geq 0.$$

$$\text{D HoldsAt}(\text{O}^{\text{TS}} \text{ SendMarketSupport}(\text{AD})[\text{CA}, \text{D}], \tau_s) \leftarrow \quad (\text{A11})$$

$$\exists \tau_t, N : \text{Happens}(\text{trigger}(\text{O}^{\text{T}} \text{ ConcludeCustodyAccount}[\text{CA}], \text{N}), \tau_t) \\ \wedge \text{D HoldsAt}(\text{ConcludeCustodyAccount}[\text{CA}], \tau) \\ \wedge (\tau_s = \tau_t + N) \wedge N \geq 0$$

$$\text{D HoldsAt}(\text{O}^{\text{TS}} \text{ SendMarketSupport}(\text{CL})[\text{CA}, \text{D}], \tau_s) \leftarrow \quad (\text{A12})$$

$$\exists \tau_t, N : \text{Happens}(\text{trigger}(\text{O}^{\text{T}} \text{ ConcludeCustodyAccount}[\text{CA}], \text{N}), \tau_t) \\ \wedge \text{D HoldsAt}(\text{ConcludeCustodyAccount}[\text{CA}], \tau) \\ \wedge (\tau_s = \tau_t + N) \wedge N \geq 0$$

The details of Axioms A10 to A12 are, respectively, similar to the details of axioms for R2.

The rule R5, on the other hand, prescribes a condition that only consultants with securities competence level C or above are permitted to provide investment advice to a customer. Here, it should be noted that handling permissions is one of the fundamental requirements for normative modelling and has been argued that any formal language not able to properly model permissions is doomed to capture real-life compliance requirements (Governatori, 2015). In this regard, Governatori and Hashmi (2015) has extended DEC to cater this need by considering the dual relationship⁵ between obligation and permission, that is, $\text{O}\phi = \sim \text{P} \sim \phi$. Accordingly, the rule R5 can be modelled as follow:

$$\begin{aligned} & \text{D HoldsAt}(\text{P}^{\text{TS}} \text{CompetenceCorAbove}[\text{CA}], \tau_s) \leftarrow \\ & \exists \tau_t, N : \text{Happens}(\text{trigger}(\text{P}^{\text{TS}} \text{SecurityAdvise}[\text{CA}], N), \tau_t) \\ & \wedge \text{D HoldsAt}(\text{InvestmentAdvice}[\text{CA}], \tau_s) \wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned} \quad (\text{A13})$$

Next, the rule R6 prescribes an obligation that the customer identification and customer legitimization must be handled by customer advisor under normal circumstances. However, in case of suspected money laundering, it should be handled by anti-money laundering officer. This separation of duties (SoD) allows companies to streamline their employees according to their capabilities and experiences. Since rule R6 prescribe two separate provisions, we model the rule into two sub-rules as follows:

$$\begin{aligned} & \text{D HoldsAt}(\text{O}^{\text{TS}} \text{CL}[\text{CA}, \text{D}], \tau_s) \leftarrow \\ & \exists \tau_t, N : \text{Happens}(\text{trigger}(\text{O}^{\text{TS}} \text{SecurityAdvise}[\text{CA}], N), \tau_t) \\ & \wedge \text{D HoldsAt}(\text{InvestmentAdvice}[\text{CA}], \tau) \wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned} \quad (\text{A14})$$

$$\begin{aligned} & \text{D HoldsAt}(\text{O}^{\text{TS}} \text{Check}(\text{CId})[\text{CA}, \text{D}], \tau_s) \leftarrow \\ & \exists \tau_t, N : \text{Happens}(\text{trigger}(\text{O}^{\text{TS}} \text{SecurityAdvise}[\text{CA}], N), \tau_t) \\ & \wedge \text{D HoldsAt}(\text{InvestmentAdvice}[\text{CA}], \tau) \wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned} \quad (\text{A15})$$

$$\begin{aligned} & \text{D HoldsAt}(\text{O}^{\text{TS}} \text{CheckMLC}[\text{MLO}], \tau_s) \leftarrow \\ & \exists \tau_t, N : \text{Happens}(\text{trigger}(\text{O}^{\text{TS}} \text{SuspectMLC}[\text{CA}], N), \tau_t) \\ & \wedge \text{D HoldsAt}(\text{SecurityAdvice}[\text{CA}], \tau) \wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned} \quad (\text{A16})$$

Notice that from a business process execution perspective, R6 is a structural rule prescribing segregation of duty (SoD) constraint and implements a prohibition by assigning the activities to the relevant/appropriate personnel. Axiom A16 implements the prohibition as an obligation based on the duality relation between obligation and prohibition written formally as $O \sim \phi = F\phi$, meaning that if it is obligatory not to do ϕ , then it is prohibited to do ϕ (Sartor, 2005).

R7, on the other hand, expresses an exception in which customers advisor can conclude a custody account if a resolution of an unresolved money-laundering case is achieved. Essentially, it is important that the language can represent exceptions which in turn is required to correctly understand the distinction between strong and weak permissions⁶. In the legal domain, strong permissions express exceptions to obligations, i.e., derogating obligation with a permission provides an exception to prohibition (Governatori et al., 2013; Stolpe, 2010). In this case, R7 can be expressed as either a permission or a maintenance obligation where the permission represents the exception to the obligation as follows:

$$\begin{aligned} & \text{D HoldsAt}(\text{P}^{\text{TS}} \text{ConcludeCustodyAccountContract}[\text{CA}], \tau_s) \leftarrow \\ & \exists \tau_t, N : \text{Happens}(\text{trigger}(\text{P}^{\text{T}} \text{ResolvedMLC}[\text{MLO}], N), \tau_t) \\ & \wedge \text{D HoldsAt}(\text{SuspectedMLC}[\text{CA}], \tau) \wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned} \quad (\text{A17})$$

$$\begin{aligned} & \text{D HoldsAt}(\text{O}^{\text{Ts}} \text{ ConcludeCustodyAccountContract}[\text{CA}], \tau_s) \leftarrow \\ & \exists \tau_t, N : \text{Happens}(\text{trigger}(\text{O}^{\text{T}} \text{ ResolvedMLC}[\text{MLO}], N), \tau_t) \\ & \wedge \text{D HoldsAt}(\text{SuspectedMLC}[\text{CA}], \tau) \wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned} \quad (\text{A18})$$

In contrast, modelling the rule as a maintenance obligation of not concluding a custody account until a money-laundering case is solved represents a prohibition. It has been argued (Hashmi, 2015; Hashmi & Governatori, 2017; Hashmi et al., 2014, 2016) that maintenance obligations are suitable for representing prohibitions since maintenance obligations must be complied with for all the instants of the interval in which the obligation is in force.

R8 also prescribes a maintenance obligation which is followed as:

$$\begin{aligned} & \text{D HoldsAt}(\text{O}^{\text{Ts}} \text{ UpdateCustomerInformation}[\text{D}], \tau_s) \leftarrow \\ & \exists \tau_t, N : \text{Happens}(\text{trigger}(\text{O}^{\text{T}} \text{ FutureCustomerContact}[\text{CU}, \text{D}], N), \tau_t) \\ & \wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned} \quad (\text{A19})$$

The meaning of Axiom A19 is similar to the meaning of axioms for R7. Rule R9, on the other hand, presents interesting case of data and temporal and permission constraints which we model as below:

$$\begin{aligned} & \text{D HoldsAt}(\text{P}^{\text{Ts}} \text{ ChargeAccountFee}[\text{CA}, 5 \text{ Euro}, 1 \text{ Year}], \tau_s) \leftarrow \\ & \exists \tau_t, N : \text{Happens}(\text{trigger}(\text{P}^{\text{Ts}} \text{ StocklessCustodyAccount}[\text{CU}, \text{CAD}], N), \tau_t) \\ & \wedge \text{D HoldsAt}(\text{InvestmentAdviseCase}[\text{CA}], \tau) \wedge (\tau_s = \tau_t + N) \wedge N \geq 0 \end{aligned} \quad (\text{A20})$$

The permission to ChargeAccountFee is represented as fluent by the permission fluent ($\text{P}^{\text{Ts}} \text{ ChargeAccountFee}$) which deontically enters into force at time τ_s . The event trigger($\text{P}^{\text{Ts}} \text{ StocklessCustodyAccount}_{[\text{CU}, \text{CAD}], N}$), τ_t whose meaning is to initiate the permission to charge the account fee for stockless account, and when the fluent InvestmentAdvise deontically holds when event occurs at τ . The temporal and data process aspects conditions are attached to the fluent.

6. COMPLEXITY EVALUATION

In this section, we present the complexity evaluation of the ease in terms of efforts and time required to modelling the compliance rules using newly introduced process aspect predicates. For this purpose, we use Halstead Complexity Metrics (HCM) (Halstead, 1977), a well-known software performance and quality metric. We selected the Halstead metrics due to its flexible nature as the metrics can be used to measure the efforts needed to model compliance rules without modifying any existing parameters, which is not possible with other metric tools such as Cyclomatic Complexity and Reachability Matrix (McCabe, 1976) and information flow (Henry & Kafura, 1981). The HCM consists of unique operators and operands to measure the complexity of efforts to write compliance rules which are generally determined before the function can be computed. The operators in HCM cover the commands and program structuring elements such as parenthesis, brackets, commas, etc., whereas operands enumerate variables and consonants (Mendling, 2007; Zasada, 2019). The metrics evaluates the complexity of effort by computing the predicate length, volume, difficulty and effort and time required to model the rules through the formulas shown in Table 3.

Table 3. Halstead's operators and operands definitions on DEC axioms

ID	n_1	n_2
R1	Parentheses(), Bracket[], Comma(), Operator(\leftarrow), Operator(\exists), Colon(:), Operator(\wedge), Operator(+), Operator(=), Operator(\geq)	Pattern, ModalityOperator, TimeStamp, Fluent, Event, Delay, ResourcesAspect, DataAspect, ZERO
R2	Parentheses(), Bracket[], Comma(), Operator(\leftarrow), Operator(\exists), Colon(:), Operator(\wedge), Operator(+), Operator(=), Operator(\geq)	Pattern, ModalityOperator, TimeStamp, Fluent, ProcessAspect(resource), ProcessAspect(data), Delay, Event, ZERO
R3	Parentheses(), Comma(), Bracket[], Operator(\leftarrow), Operator(\exists), Colon(:), Operator(\wedge), Operator(+), Operator(=), Operator(\geq)	Pattern, ModalityOperator, TimeStamp, Fluent, ProcessAspect(resource), ProcessAspect(date), Delay, Event, ZERO
R4	Parentheses(), Comma(), Bracket[], Operator(\leftarrow), Operator(\exists), Colon(:), Operator(\wedge), Operator(+), Operator(=), Operator(\geq)	Pattern, ModalityOperator, timestamp, Fluent, ProcessAspect(resource) ProcessAspect(data), Delay, Event, ZERO
R5	Parentheses(), Comma(), Bracket[], Operator(\leftarrow), Operator(\exists), Colon(:), Operator(\wedge), Operator(+), Operator(=), Operator(\geq)	Pattern, ModalityOperator, TimeStamp, Fluent, ProcessAspect(resource) ProcessAspect(data), Delay, ZERO
R6	Parentheses(), Comma(), Bracket[], Operator(\leftarrow), Operator(\exists), Colon(:), Operator(\wedge), Operator(+), Operator(=), Operator(\geq)	Pattern, ModalityOperator, TimeStamp, Fluent, ProcessAspect(resource) Delay, Event, ZERO
R7	Parentheses(), Comma(), Bracket[], Operator(\leftarrow), Operator(\exists), Colon(:), Operator(\wedge), Operator(+), Operator(=), Operator(\geq)	Pattern, ModalityOperator, TimeStamp, Fluent, ProcessAspect(resource), Delay, Event, ZERO
R8	Parentheses(), Comma(), Bracket[], Operator(\leftarrow), Operator(\exists), Colon(:), Operator(\wedge), Operator(+), Operator(=), Operator(\geq)	Pattern, ModalityOperator, Fluent, ProcessAspect(data), TimeStamp, ProcessAspect(resource) Delay, Event, ZERO
R9	Parentheses(), Comma(), Bracket[], Operator(\leftarrow), Operator(\exists), Colon(:), Operator(\wedge), Operator(+), Operator(=), Operator(\geq)	Pattern, ModalityOperator, TimeStamp, Fluent, ProcessAspect(resource), ProcessAspect(data), Delay, Event, ZERO

$$\text{Vocabulary (size), } n = n_1 + n_2 \quad (1)$$

$$\text{Predicate Length (size), } N = N_1 + N_2 \quad (2)$$

$$\text{Predicate Volume, } V = N \times \log_2 n \quad (3)$$

$$\text{Predicate Difficulty, } D = \frac{n_1}{2} \times \frac{N_1}{2} \quad (4)$$

$$\text{Predicate Effort, } E = D \times V \quad (5)$$

$$\text{Time (in sec), } T = \frac{E}{18} \quad (6)$$

where n_1 and n_2 are the number of unique operators and operands (variables or constants), and N_1 and N_2 , respectively, are the sum of operators and operands occurrences in the program. They are defined as the parameters of complexity measurement before the start of computations.

In principle, the measure of complexity, i.e., the effort of modelling the compliance rules, is directly proportional to the size of operators and operands and the frequency with which they appear in the program. Besides, the metrics can also be extended to measure the correctness of a program with regards to the size, the level of abstraction of the program, and the number of errors delivered (Ferrer et al., 2013).

Table 3 illustrate the operators and operands representing various elements of the compliance rules derived as per the following criteria:

- Patterns, deontic modalities (i.e., norm types, modality operator etc.), tasks, triggers, events, gateways are enumerated for all flow objects.
- Logical operators, sequences, message flows for connectives.
- Process aspects such as time, data and resources, etc., are counted for process aspects.
- No brackets included, and parenthesis enumerated as pairs.
- Deontic modality and pattern enumerated per concrete rule.

Notice that HCM is not computed per language but per concrete program covering all characteristics of the language. Hence, the enumeration criteria have been derived based on the characteristics of the language as well as concrete compliance rules.

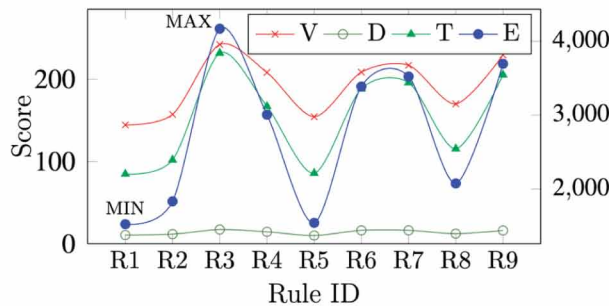
Figure 2 show the HCM values computed for the DEC axioms modelled in Section 4. Given the stable levels of difficulty (D) and sizes (V) of predicate per concrete rule, the time (T) and efforts (E) of modelling the compliance rules relevant to the process aspects with DEC predicates fluctuates across the rules as shown in Figure 2(b). The average time and effort required to model a rule is $E = 2,751.23$ and $T = 152.85$ sec, which is relatively low. This clearly demonstrates that the complexity of modelling the compliance rules with DEC predicates is significantly lower even when the complexity of the standard EC is exponential assuming the relative time and partial order nature, and the precise semantics of the EC (Cervesato et al., 1994).

Accordingly, on the rule level, from Figure 2(a) it can be seen that the minimum effort $E = 1,525.18$ and time $T = 84.73$ sec, and the maximum $E = 4,169.48$ and $T = 231.64$ sec, are computed for R1 and R3, respectively. For R1, it is lower because the rule prescribes simple flow related constraint and has not complex structure, thus it can be modelled with simple flow control constructs that is available in almost every modelling language (such as Event-Driven Process Chains (EPC) (Mendling, 2008), BPMN⁷, etc.). In contrast for other rules, for example, R3 and R6 and R9, respectively, the time and effort to write the rule gradually increase. This is because these rules have complex structure and prescribe multiple process aspects conditions attached to the fluent increasing thus the difficulty of modelling the rules. To simplify the modelling process, we split complex rules into distinct DEC axioms. This leads to some axioms containing repetitive elements which increased the relative size (V) of rule. However, as discussed earlier, the size of efforts is directly proportional to the number of operators and operands, and the frequency with which they appear in the axiom. Hence, due to the increase in size of the rules, the time and efforts required to write the rules has been slightly increased, yet such efforts remain relatively small.

Figure 2. (a) and (b) Halstead metric computations for DEC axioms

Rule ID	n_1	n_2	n	N_1	N_2	N	V	D	E	T	Remark
R1	10	9	19	15	19	34	144.43	10.56	1525.18	84.73	MIN
R2	10	9	19	16	21	37	157.17	11.67	1834.17	101.90	
R3	10	9	19	26	31	57	242.13	17.22	4169.48	231.64	MAX
R4	10	9	19	23	26	49	208.15	14.44	3005.69	166.98	
R5	10	8	18	21	16	37	154.29	10.00	1542.9	85.72	
R6	10	8	18	24	26	50	208.5	16.25	3388.13	188.23	
R7	10	8	18	26	26	52	216.84	16.25	3523.65	195.76	
R8	10	9	19	18	22	40	169.92	12.22	2076.42	115.36	
R9	10	9	19	25	29	54	229.39	16.11	3695.47	205.30	
Mean	10	8.67	18.67	21.56	24	45.56	192.31	13.86	2751.23	152.85	

(a)



(b)

7. CONCLUSION

In this paper, we have extended DEC predicates to intuitively represent normative requirements related to the aspects which is one basic requirement to validate the compliant behaviour of business processes. The extended predicates are expressive and present low complexity in modelling the compliance requirements. At this point we have evaluated the textual complexity of the extended predicates in terms of efforts, difficulty and time required for modelling and the norms. We plan to implement this work to evaluate the efficiency in order to validate the computational complexity of the proposed predicates as the complexity analysis at this stage is relatively small. Moreover, in the context of this paper, we have used the notion of permission in a boarder sense and do make any distinction of the concepts. In literature, however, various classes of permissions exist. We plan to continue this work to address the issue of representing various cases of permissions and exceptions involving the process aspects with DEC. More specifically, we plan to create a catalogue of DEC pattern predicates for modeling types of permissions. On the same note, the scope of used process aspects such as data, resources, time, etc., and the compliance requirements are generic. However, these aspects are granular and may involve more complex process interaction patterns such as iteration, termination, force completion, triggering patterns, etc. We plan to investigate these aspects and develop a comprehensive catalogue of DEC predicate patterns covering compliance requirements about more complex process aspects. The workflow patterns proposed by Nick Russell et al. (2016) can be the starting point toward this direction.

Another line of interest is to develop transformations of the proposed predicates such that (legal) norms pertaining business process aspects can be represented using LegalRuleML, and into its equivalent defeasible logic transformations. This can facilitate representing business contracts and performing reasoning about them in a more intuitive and declarative way. Besides, developing such transformations and applying them into some smart contract-based systems e.g., ethereum can extend the language of such systems in a way that users can make their smart contracts explicit and define them declaratively. The methodology proposed by Lam and Hashmi (2019) can be benefited for developing such transformations.

ADDITIONAL FUNDING INFORMATION

The publisher has waived the Open Access Processing fee for this article.

ACKNOWLEDGMENT

I sincerely thank Dr Ho-Pun Lam for his sincere and fruitful discussions significantly improving the quality of this manuscript.

REFERENCES

- Abouzaid, F., & Mullins, J. (2008). A Calculus for Generation, Verification and Refinement of BPEL Specifications. *Electronic Notes in Theoretical Computer Science*, 200(3), 43-65. 10.1016/j.entcs.2008.04.092
- Alexiev, V. (1995). *The Event Calculus as a Linear Logic Program*. <https://era.library.ualberta.ca/items/0339b9fb-8715-47b1-ad1f-879653397016/view/28b9a3ad-72c1-4368-925e-407f0581b1cb/TR95-24.pdf>
- Andrade-Lotero, E. J. (2006). *Meaning and Form in Event Calculus*. Institute for Logic, Language and Computation, University of Amsterdam. <https://eprints.illc.uva.nl/764/>
- Antoniou, G., Maher, M. J., & Billington, D. (2000). Defeasible logic versus Logic Programming without Negation as Failure. *The Journal of Logic Programming*, 42(1), 47-57. 10.1016/S0743-1066(99)00060-6
- Berndtsson, M., & Mellin, J. (2009). ECA Rules. *Springer US*. doi:10.1007/978-0-387-39940-9_504
- Cervesato, I., Chittaro, L., & Montanari, A. (1994). What the Event Calculus actually does, and how to do it efficiently. GULP-PRODE.
- Cervesato, I., Franceschet, M., & Montanari, A. (1997). Modal Event Calculi with Preconditions. *Proceedings of the 4th International Workshop on Temporal Representation and Reasoning*.
- Farrell, A. D. H., Sergot, M. J., Salle, M., Bartolini, C., Trastour, D., & Christodoulou, A. (2004). Performance monitoring of service-level agreements for utility computing using the event calculus. *Proceedings of the 1st IEEE International Workshop on Electronic Contracting*.
- Ferrer, J., Chicano, F., & Alba, E. (2013). Estimating software testing complexity. *Information and Software Technology*, 55(12), 2125–2139.
- Governatori, G., Olivieri, F., Rotolo, A., & Scannapieco, S. (2015). Thou Shalt is Not You Will. *Proceedings of ICAIL'15*. doi:10.1007/s10992-013-9295-1
- Governatori, G., & Rotolo, A. (2006). Logic of Violations: A Gentzen System for Reasoning with Contrary-To-Duty Obligations. *The Australasian Journal of Logic*, 4.
- Governatori, G., & Hashmi, M. (2015). Permissions in Deontic Event-Calculus. *Proceedings of the Jurix 2015*. doi:10.1007/s10992-013-9295-1
- Governatori, G., Olivieri, F., Rotolo, A., & Scannapieco, S. (2013). Computing Strong and Weak Permissions in Defeasible Logic. *Journal of Philosophical Logic*, 42(6), 799–829. doi:10.1007/s10992-013-9295-1
- Halle, S., & Villemare, R. (2008). Runtime Monitoring of Message-Based Workflows with Data. *2008 12th International IEEE Enterprise Distributed Object Computing Conference*.
- Halstead, M. H. (1977). *Elements of Software Science*. Elsevier Science Inc.
- Hashmi, M. (2015). *Evaluating Business Process Compliance Management Frameworks*. Queensland University of Technology.
- Hashmi, M., & Governatori, G. (2017). Norms modeling constructs of business process compliance management frameworks: a conceptual evaluation. *Artificial Intelligence and Law*, 26(3), 251–305.
- Hashmi, M., Governatori, G., & Wynn, M. T. (2014). Modeling Obligations with Event-Calculus. *Proceedings of RuleML'2014*.
- Hashmi, M., Governatori, G., & Wynn, M. T. (2016). Normative requirements for regulatory compliance: An abstract formal framework. *Information Systems Frontiers*, 18(3), 429–455.
- Henry, S., & Kafura, D. (1981). Software Structure Metrics Based on Information Flow. *IEEE Transactions on Software Engineering*, SE-7(5), 510–518. doi:10.1109/TSE.1981.231113
- Joost, K., & Hans, W. (2020). Formalising Commitments Using the Event-Calculus. Value Modelling and Business Ontologies (VMBO 2020), Brussels, Belgium.

- Kiepuszewski, B., ter Hofstede, A. H. M., & van der Aalst, W. M. P. (2003). Fundamentals of control flow in workflows. *Acta Informatica*, 39(3), 143–209.
- Knuplesch, D., Reichert, M., Ly, L. T., Kumar, A., & Rinderle-Ma, S. (2013). Visual Modeling of Business Process Compliance Rules with the Support of Multiple Perspectives. *Proceedings of ER 2013*. doi:10.1007/978-3-642-41924-9_10
- Kowalski, R., & Sergot, M. (1986). A Logic-based Calculus of Events. *New Generation Computing*, 4(1), 67–95. doi:10.1007/978-3-642-41924-9_10
- Kumar, A., & Barton, R. R. (2017). Controlled violation of temporal process constraints – Models, algorithms and results. *Information Systems*, 64, 410–424. doi:10.1016/j.is.2016.06.003
- Lam, H.-P., & Hashmi, M. (2019). Enabling reasoning with LegalRuleML. *Theory and Practice of Logic Programming*, 19(1), 21–26.
- Lam, H.-P., Hashmi, M., & Kumar, A. (2020). Towards a Formal Framework for Partial Compliance of Business Processes. *AI Approaches to the Complexity of Legal Systems (AICOL 2020)*.
- Lohrmann, M., & Reichert, M. (2010). *Basic Considerations on Business Process Quality*. http://dbis.eprints.uni-ulm.de/667/1/LoRe10_TR.pdf
- Makinson, D., & van der Torre, L. (2003). Permission from an Input/Output Perspective. *Journal of Philosophical Logic*, 32(4), 391–416. doi:10.1023/A:1024806529939
- McCabe, T. J. (1976). A Complexity Measure. *IEEE Transactions on Software Engineering*, SE-2(4), 308–320.
- Mendling, J. (2007). *Detection and Prediction of Errors in EPC Business Process Models*. Vienna University of Economics and Business Administration.
- Mendling, J. (2008). *Event-Driven Process Chains (EPC)*. Springer Berlin Heidelberg. doi:10.1007/978-3-540-89224-3_2
- Miller, R., & Shanahan, M. (1999). The Event Calculus in Classical Logic - Alternative Axiomatisations. *The Electronic Transactions on Artificial Intelligence*, 3(A), 77–105.
- Miller, R., & Shanahan, M. (2002). Some Alternative Formulations of the Event Calculus. In A. C. Kakas & F. Sadri (Eds.), *Computational Logic: Logic Programming and Beyond: Essays in Honour of Robert A. Kowalski Part II* (pp. 452–490). Springer Berlin Heidelberg. doi:10.1007/3-540-45632-5_17
- Montali, M., Chesani, F., Mello, P., & Maggi, F. M. (2013). Towards Data-aware Constraints in Declare. *Proceedings of ACM/SIGAPP SAC '13*. doi:10.1145/2480362.2480624
- Montali, M., Maggi, F. M., Chesani, F., Mello, P., & Aalst, W. M. P. v. d. (2014). Monitoring Business Constraints with the Event Calculus. *ACM Transactions on Intelligent Systems and Technology*, 5(1), 17:11–17:30. doi:10.1145/2480362.2480624
- Pereira, J. L., & Varajão, J. (2017). The Temporal Dimension of Business Processes - Dealing with Time Constraints. *Procedia Computer Science*, 121, 1034–1038.
- Pnueli, A. (1977). The Temporal Logic of Programs. *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*. doi:10.1109/SFCS.1977.32
- Russell, van der Aalst, & Hofstede. (2016). *Workflow Patterns: The Definite Guide*. MIT Press. <https://mitpress.mit.edu/books/workflow-patterns>
- Sadri, F., & Kowalski, R. (1995). Variants of the Event Calculus. *Proceedings of the 12th International Conference on Logic Programming*. doi:10.1109/SFCS.1977.32
- Sartor, G. (2005). *Legal Reasoning: A Cognitive Approach to the Law* (Vol. 5). Springer. doi:10.1109/SFCS.1977.32
- Shanahan, M. (1999). The Event Calculus Explained. In M. J. Wooldridge & M. Veloso (Eds.), *Artificial Intelligence Today: Recent Trends and Development* (pp. 409–430). Springer Berlin Heidelberg. doi:10.1007/3-540-48317-9_17

Stolpe, A. (2010). Deontic Logic in Computer Science. In *10th International Conference, DEON 2010, Fiesole, Italy, July 7-9, 2010. Proceedings* (pp. 98–115). Springer Berlin Heidelberg.

Stroppi, L. J. R., Chiotti, O., & Villarreal, P. D. (2015). Defining the resource perspective in the development of processes-aware information systems. *Journal of Information and Software Technology*, 59(C), 86–108.

Turetken, O., Elgammal, A., van den Heuvel, W.-J., & Papazoglou, M. P. (2012). Capturing Compliance Requirements: A Pattern-Based Approach. *IEEE Software*, 29(3), 28–36. doi:10.1109/MS.2012.45

Vardi, M. Y. (2001). Branching vs. Linear Time: Final Showdown. *Tools and Algorithms for the Construction and Analysis of Systems*.

von Wright, G. H. (1951). Deontic Logic. *Mind*, 60(237), 1–15. <https://www.jstor.org/stable/2251395>

Zasada, A. (2019). *How Complex Does Compliance Get?* Information Systems Engineering in Responsible Information Systems.

Zasada, A., & Bui, T. (2018). More Than Meets the Eye: A Case Study on the Role of IT Affordances in Supporting Compliance. *Proceedings of AMCIS'2018*. doi:10.1007/978-3-030-21297-1_22

Zur Muehlen, M. (2004). Organizational Management in Workflow Applications – Issues and Perspectives. *Information Technology and Management*, 5(3), 271–291. doi:10.1007/978-3-030-21297-1_22

ENDNOTES

- ¹ DEC provides various predicates, events, and boolean switches for handling cases and effects of the obligations; however, due to space limit we only list predicates and events that are relevant to the discussion. Interested readers please see Hashmi et al. (2014) for a full list.
- ² There may be other cases of termination of an achievement obligation, please refer Hashmi et al. (2016) for details.
- ³ Here we abuse the notation a little bit by putting a fluents set into the predicate.
- ⁴ To simplify the notations, the following abbreviations will be used in the modelled axioms: Customer=CU, Customer Advisor=CA, Customer Data=CD, Customer Account Data=CAD, Money Laundry Case=MLC, Money Laundry Officer=MLO, Stockless Custody Account=SCA, Basic Security Information=BSI, Capital Investment=CI, Customer Legitimation=CL, Account Details=AD, Customer Identification=CId, Customer Account=CustAcct, Future Customer Contact=FCC.
- ⁵ Notice that the idea of permission is largely elusive and has not been sufficiently researched in literature. Generally, the relationship between permission and obligation is viewed as inadequate because in some situations, it hardly allows to model the meaning of some cases of norms (Governatori, G., Olivieri, F., Rotolo, A., & Scannapieco, S. (2013). Computing Strong and Weak Permissions in Defeasible Logic. *Journal of Philosophical Logic*, 42(6), 799-829. Also, given one deals with unconditional obligations, the idea of weak permission can be seen as a dual of obligation Makinson, D., & van der Torre, L. (2003). Permission from an Input/Output Perspective. *Ibid.*, 32(4), 391-416. We subscribe to this position and assume the dual relation between obligation and permission without making any distinction between various types of permission.
- ⁶ In literature permissions and prohibitions have been extensively studied. Several types of permissions such as strong, weak (Governatori, G., Olivieri, F., Rotolo, A., & Scannapieco, S. (2013). Computing Strong and Weak Permissions in Defeasible Logic. *Ibid.*, 42(6), 799-829.), positive and negative permissions (Makinson, D., & van der Torre, L. (2003). Permission from an Input/Output Perspective. *Ibid.*, 32(4), 391-416.) have been proposed. In this paper, we do not make any distinction and use permission in the generic sense.
- ⁷ Business Process Modelling Notation (BPMN): <https://www.omg.org/spec/BPMN/2.0/>

Mustafa Hashmi is a Research Fellow with Center for Law, La Trobe University, Australia, and Lecturer at Federation University (Brisbane Campus), Australia. Besides, Mustafa is a Visiting Research Scientist at Data 61, CSIRO Australia where he earlier served as a Research Engineer and Graduate Researcher, respectively. He is engaged in several (inter-) national strategic projects including D2D - CRC project, CSIRO Legal Informatics, and EU SPIRIT projects. He is the author of DEC—a normative variant of Event-Calculus, and (with colleagues) the recipient of Best Paper Award at RuleML 2016 held in the United States for his co-authored work on Enabling Reasoning with LegalRuleML. Before joining Data61, CSIRO in Australia, Mustafa has been working in Germany in various capacities primarily in the ICT domain. Mustafa has extensively published in various premium journals and conferences on Computer Science and Legal Reasoning and co-authored an edited book on Service Research and Innovation. His research interests include, but not limited to, in the areas of automation and analysis of business processes, regulatory compliance management, non-monotonic reasoning, defeasible and model logics, and their applications to solve complex problems in large scale enterprises. Mustafa is Senior member of the IEEE Computer Society.