Business Process Activity Relationships: is there anything beyond arrows?

Greta Adamo[1,4], Stefano Borgo[2], Chiara Di Francescomarino[1], Chiara Ghidini[1], Nicola Guarino[2], and Emilio M. Sanfilippo[3]

[1] FBK-IRST, Trento, Italy
[2] ISTC-CNR Laboratory for Applied Ontology, Trento, Italy
[3] Ecole Centrale de Nantes – Laboratoire des Sciences du Numérique – LS2N, Nantes, France
[4] University of Genova, DIBRIS, Genova, Italy
{adamo,dfmchiara,ghidini}@fbk.eu
{stefano.borgo,nicola.guarino}@cnr.it
{emilio.sanfilippo@ls2n.fr}

Abstract. Business process modelling languages enable the depiction of the processes of an organisation by exploiting graphical symbols to denote the key elements to be represented. Despite the variety of approaches, graphical symbols, and (in)formal interpretations associated to the different languages, a fundamental component of every business process modelling language is the representation of the way activities are related by means of control arcs and gateways. While *multiple* kinds of relationships may hold among such activities, mainstream business process modelling languages seem actually only interested in modelling a *single* (very important) kind of relationship, namely the *activity execution order* within the control flow. In this paper we investigate the role of another kind of fundamental relationship between activities, namely *ontological dependence*, in the context of business process modelling. In particular, we introduce three forms of generic ontological dependence, namely *historical* dependence, *causal* dependence, and *goal-based co-occurrence*. We illustrate different forms in which they can occur, we introduce a language to express them and we discuss their usefulness in two concrete use cases.

1 Introduction and Motivations

Business process modelling languages enable the depiction of the processes of an organisation by exploiting graphical symbols to denote the key elements to be represented. Examples are the sequence of activities to be executed (the so-called control flow), the actors involved, the data objects required/manipulated by the activities, message exchanges, and so on.

Despite the variety of approaches, graphical symbols, and their (in)formal interpretations, a fundamental component of every business process modelling language is the representation of the way activities (and events) are related by means of control arcs and connectors (gateways). However, while mainstream business process modelling languages seem actually only interested in modelling a *single* (very important) kind of relationship, namely the *activity execution order* within the control flow, *multiple* kinds of other relationships may hold among such activities.

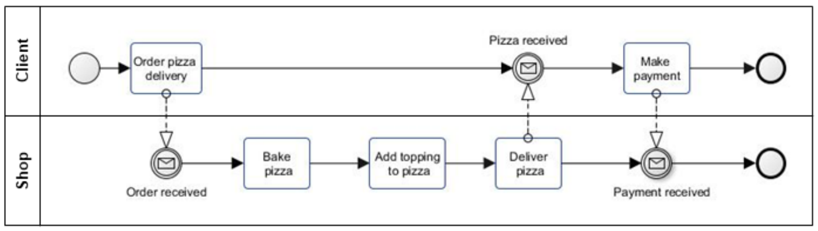


Fig. 1. A simple pizza delivery process model.

Consider, for instance, the simple BPMN diagram of Fig. 1. Its control arcs specify that the execution of a pizza delivery process starts with the order, continues with the baking of the pizza, the addition of toppings, the delivery, and the payment. In addition to the relation between activities captured by the control arcs, most human beings would easily identify further relationships in this process. As a first example, the (indirect) relationship between `Bake pizza` and `Deliver pizza` presupposes an intrinsic execution order that is independent on this particular process model. Indeed, delivering a pizza requires having (made) it first. This relation does not depend upon the way the organisation decides to structure the control flow. On the contrary, it holds in virtue of the very nature of such activities in the *real world*, and this influences the way the real business processes are organised (and thus represented in the model).

As a second example, one may notice that `Deliver pizza` and `Make payment` exhibit a different kind of mutual relationship. Indeed, an organisation may freely organise its own processes asking for payment before or after a delivery. We can nonetheless assume that the commercial nature of the pizza shop and its business goal of making money suggests that delivering a pizza must be (sooner or later) associated to a payment in order to have a meaningful process. These two simple examples show different real world relations between activities that can hold in the real world. Nonetheless, they are represented in the same way in the process model of Fig. 1. This happens because the model only represents the execution order of activities within the control flow.

The inability to account for aspects coming from *real world* constraints makes the standard business process modelling notations less informative from an explanatory perspective, and less robust against possible changes that violate fundamental domain constraints. Indeed, while the (intentionally very simple) pizza shop example reflects characteristics of the real world that most of us know, intrinsic aspects of more complex domains may be more difficult to understand, and could be missed by people (or algorithms) who lack the background knowledge required to understand them.

This may result in crucial modelling mistakes, especially during model redesign. For example, while common sense would prevent refactoring the pizza process by imposing to deliver a pizza before baking it, no information in the model actually forbids that. Similarly, removing the payment activity from the process would dramatically change its meaning, so as to question whether it should still be considered the “same” process. On the contrary, the removal of the `Add topping to pizza` activity would intuitively be considered just a process refactoring.

Characterising relationships between business process activities beyond the control flow perspective is not trivial, due to the multitude of aspects and features that may be considered. For example, activity relationships may be distinguished according to their temporal features, co-occurrence constraints, the nature of actors or participants involved, and the goals of the business process. Some of these aspects reflect *normative choices* (or *business rules*) concerning the expected process structure, while others are bound to genuine *ontological constraints* intrinsic in the activities themselves. In the remaining of the paper we provide an analysis of some of such constraints, focusing in particular on temporal co-occurrence, and we apply them to distinguish among different kinds of activity relationships within business process models. In particular, we provide:

- an analysis of activity co-occurrences in terms of ontological dependences which allows to select and introduce three forms of generic ontological dependence among activities, namely *historical* dependence, *causal* dependence, and *goal-based co-occurrence* (Section 2);
- a first investigation of different forms of historical dependence, causal dependence, and goal-based co-occurrence, depending on their genuine *ontological* aspects, *goal*-related aspects, and *norm*-related aspects (Section 3);
- a proposal on how to incorporate historical dependence, causal dependence and goal-based co-occurrence in business process models by following a hybrid modelling approach (Section 4);
- an illustration of the usefulness of historical dependence, causal dependence, and goal-based co-occurrence in two concrete use cases concerning business process documentation and business process redesign (Section 5).

2 Activity Co-occurrence as Ontological Dependence

The goal of this paper is to make explicit the nature of the links holding amongst activities that pertain a business process. We rely here on Weske's definition [30], according to which a business process is *"a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations."*

In particular we based our analysis on *ontological dependences* resulting in co-occurrence constraints involving activities that occur during the same process execution. Such constraints hold by necessity in a particular domain, independently of the way a business process is designed. For example, delivering a pizza necessarily presupposes that the pizza has been baked. Similarly, no receive event can occur without a corresponding send event.

In formal ontology, ontological dependence is a fundamental relationship (or set of relationships) which can take many forms [5, 10]. In general, an entity is dependent upon another when it is not ontologically *self-sufficient*, in the sense that it cannot exist alone. A basic form of dependence is so-called *specific* (or *rigid*) *existential dependence*, which holds among two objects when the existence of one necessarily implies the existence of the other. For instance, we may say that a person is specifically existentially dependent on her brain. A weaker form is the so-called *generic existential dependence*, which holds when the existence of an object requires the existence of another *of a given kind*. For instance, a human being is generically dependent on a heart (under the assumption that the heart may be substituted). An even weaker form of dependence may hold between kinds, when the existence of an instance of one kind requires the existence of an instance of the other kind. This seems to be enough in our case, since in most business process models key elements (such as activities in a BPMN model or transitions in a Petri Net) are indeed understood as *kinds*, and we are interested in the relationships among them. However, since the instances of such kinds are temporal entities, we should speak of *occurrence* instead of *existence*, so that instead of *existential dependence relationships* we have to talk of *co-occurrence dependence relationships*. In the following, we shall introduce three forms of ontological relations that characterize the nature of such co-occurrence dependence relationships. The reason why we have chosen these specific forms of ontological dependences between activities is twofold: on the one hand they are grounded on important generic ontological dependences investigated in literature; on the other hand they seem to play a fundamental role in all the business processes (models) that have been examined for this work.

A first type of co-occurrence dependence relationship is *historical dependence*. This captures the situation where a certain activity occurrence presupposes that another activity occurred *in the past*. For example, an instance of `Deliver pizza` may occur only if an instance of `Bake pizza` occurred beforehand. We shall define historical dependence as follows:

> Let P_1 and P_2 be business process activities (that is, *kinds* of actions that may occur in a business process). We shall say that P_1 is *historically dependent* on P_2 iff, necessarily, whenever an instance *x* of P_1 occurs at time *t*, there exists an instance *y* of P_2 that has occurred at a time $t' < t$.

Note that historical dependence is a relation holding *necessarily*, and has therefore an *ontological* nature. On the contrary, a mere *temporal precedence* relation simply resulting from the fact that two activities precede one another in a *particular* business process model may have just a *prescriptive* nature, if no historical dependence holds among the same activities. For example, a certain model may say that an activity `Check contract` should always precede the activity `Sign contract`. Although these activities may be done in any order (since none of them causes or implies the existence of the other), there is a clear reason to have them in a specific temporal order, but this reason reflects a *business rule* and not an ontological constraint.

A stronger type of occurrence dependence relationship is *causal dependence*. Causality is notoriously challenging to define [11], and its complete characterisation is behind the purposes of this work. For our purposes, we assume the following definition, which characterizes causality in terms of *contribution to explanation*:

> A process activity P_1 is *causally dependent* on P_2 iff, necessarily, whenever an instance *x* of P_1 occurs, there exist an instance *y* of P_2 that occurs before *x*, whose occurrence *contributes to explain why x occurred*.

This definition is admittedly naive, but it seems to be enough for practical cases. For example, an event of message receiving occurs because an event of message delivering occurred. Analogously, a pizza delivering activity occurs because an ordering event occurred in the past, and not because a particular pizza was baked. So, the relation between `Deliver pizza` and `Bake pizza` is a historical dependence, while that between `Deliver pizza` and `Order pizza delivery` is a causal dependence. Of course, a causal dependence implies a historical dependence.

Finally, a third kind of occurrence dependence relationship is what we shall call *goal-based co-occurrence*[5]:

> Let G be a goal, typically associated to a certain business process. The process activities P_1 and P_2 are *goal-based co-occurrence* iff the occurrence of both P_1 and P_2 is necessary for the satisfaction of G.

Consider that no temporal constraint is imposed on P_1 and P_2, which may occur in whatever order. In other terms, we only say that, for the satisfaction of G, instances of P_1 cannot occur if instances of P_2 do not occur, and vice versa. Consider, for example, the activity `Deliver pizza` in Fig.1. Given the nature of our process' goal, which may be stated as "Selling pizza", both `Deliver pizza` and `Make payment` (for the pizza) are necessary for the satisfaction of such goal, and they are therefore co-occurrent with respect to such goal. Assuming that no historical dependence holds necessarily between the two activities, a process re-factoring is possible, where the delivery occurs before the payment. What is necessary, however, is that the payment occurs sooner or later. Note that goal-based co-occurrence is symmetric, differently from the previous two relations.

Note that, for the sake of simplicity, we are considering here only relationships between pairs of activities. Nonetheless the dependences introduced in this section could be generalised to multiple activities or to process patterns / sub-processes.

3 Forms of Occurrence Dependence

As already stated in Section 1, dependence relationships between business process activities can be motivated by different aspects of the world a real process is embedded in. In this section we exemplify, by means of examples, the role that (i) genuine *ontological* constraints (hereafter 'laws of nature'), (ii) the *goal* of the process, and (iii) *norms* can play in determining historical dependence, causal dependence and goal-based co-occurrence. While the categories considered here are not meant to be exhaustive, they are of fundamental importance for the representation of business processes. Genuine ontological dependence exists because of the way the real world is structured and cannot be circumvented by business processes. Dependences related to the goal often refer, in our opinion, to the very nature of the process. They may be circumvented, but their violations may have dramatical effects on the meaningfulness of the process. Finally, laws and regulations often define a social world as important as the physical one for business processes. Also in this case, dependences may be violated but their violations have strong effects on the compliance of the process w.r.t. the normative world that regulates them (see e.g., [13]).

[5] While co-occurrences may, in principle, be based on different elements, goals seem to play a fundamental role in co-occurrences in all the business processes (models) we have examined for this work. We leave the investigation of other forms of co-occurrences for future work.

3.1 Historical dependence

Historical dependence seems to play an important role in business process models and may come in different forms. A first example is provided by pairs of activities that pertain the "switch" between two complementary states such as turning on and off, entering and exiting and so on. A paradigmatic example in business process models is constituted by the activities `Login` and `Logout` from a web page in a session. While it is possible the login occurs without a logout, the opposite can not occur. If a logout does occur, then the login must have occurred. This is a particular case of historical dependence and is due to a 'law of nature' that can be generalised, as we said, to all changes between complementary and mutually exclusive binary states. Different examples still due to 'laws of nature' are the ones of `Bake pizza` and `Deliver pizza` discussed in previous sections, or the one of an administrative procedure of applying for a PhD position in which an applicant submits the PhD request (application form) to the PhD office, which is then checked for compliance to the submission rules. `Submit PhD application` and `Check PhD application` are connected together by a historical dependence as the PhD office can not check something that has not been submitted. By generalisation, the two forms of historical dependence mentioned here depend upon a 'law of nature' that determines that one can perform an activity on an artefact only if this artefact exists and is available.

An example of historical dependence related to the goal of the process is the one involving two `Make diagnosis` and `Propose treatment` activities in a healthcare process. While a diagnosis is not a genuine ontological constraint for the proposal of a treatment, the goal of the process of providing an effective (if not the best) cure to a patient triggers this historical dependence in a meaningful process.

A further example of historical dependence may be due to normative laws. For instance, in an on-line shopping purchase a `Login` activity may be a normative necessary pre-requisite for the execution of a `Purchase goods` activity, in order to certify the identity of the customer. Similarly to the example above, while a login is not ontologically needed for a customer in order to buy something, the social world determined by the norm imposes that a customer identification via `Login` is strictly necessary in order to accomplish an e-buy activity.

3.2 Causal dependence

A first form of causal dependence, due to a sort of 'law of nature', is the one that holds between `Send` and `Receive` activities (events, in certain notations). Indeed the activity `Send message` not only is an existential requirement for `Receive message` to exist but it also causes the receipt of the message itself.

Further examples of causal dependence can be found if we focus on the goal of a business process. Consider again the pizza example. In this example `Order pizza delivery` causes several further activities in the process, and in particular `Deliver pizza`. Note that this is not due to a 'law of nature' but to the goal of the pizza shop, which is the one of making money by selling pizzas to customers and fulfilling their (customers) expectations. While causal dependence is also historical dependence the opposite does not hold as `Bake pizza` does not cause its delivery. Indeed a pizza (or any good) is not sold just because it is made but because someone asked for it.

Normative regulations can also refer to activities that are involved in a causal dependence. Consider for instance the activity `First use of software` and `Evaluate terms and conditions`. In this example, the first usage of a just installed software triggers the evaluation of terms and conditions and also motivates/explains why this activity occurs in a software installation process. Similarly to the above this is not due to a 'law of nature' but to normative requirements regulating the usage of artefacts (the software, in our case).

3.3 Goal-based co-occurrence

When it comes to the goal of the process, a typical example of goal-based co-occurrence is the one involving the activities `Deliver good` and `Pay for good` in the context of an economically motivated selling-oriented business process, of which `Deliver pizza` and `Make payment` (for pizza) in Fig. 1 is a specific example already illustrated in Section 4. As a further example, consider the annual evaluation process of an employer in a given organisation. Whenever the goal is to ensure a transparent and fair evaluation, a goal-based co-occurrence may involve two activities `Send evaluation to Human Resources` and `Send evaluation to employer` executed by the employer's boss. Indeed the provision of the evaluation to Human Resources is required to make the evaluation adopted by the organisation, while with the provision of the evaluation to the employer provides a possibility to highlight unfair treatments, and they are jointly required to achieve the overall goal.

4 Modelling Dependence relationships in Business Processes

In Sections 2 and 3 we have introduced the historical dependence, the causal dependence and the goal-based co-occurrence, and illustrated, by means of examples, their occurrence in typical business process scenarios. Here we introduce a simple language for expressing these dependences, investigate their meaning in terms of temporal properties, and make a proposal on how to include them in (hybrid) business process models.

First of all we define the syntax of dependence expressions. Let $T = \{T_1, \ldots, T_n\}$ be an alphabet of business process activities. A *dependence expression* is an expression of the form $\textsc{Cooc}(T_i, T_j)$, $\textsc{Hist}(T_i, T_j)$, and $\textsc{Cause}(T_i, T_j)$, where $T_i, T_j \in T, i \neq j$.[6] Next, we need to understand what is the meaning of these expressions and what does it mean to enforce them upon a business process model.

A first question we need to clarify is whether dependence expressions concern a business process diagram (only) or execution paths. From the description of dependences provided in the previous sections, it is clear that they refer to process execution paths. Indeed when we state, e.g, that activities `Deliver pizza` and `Make payment` (for pizza) co-occur in a process model we do not simply intend that they both should appear

[6] We follow previous work in the area of BPM and focus on process models with no repeating activities, in the spirit of [1]. The investigation of dependences between repeated activities occurring in loops is left for future work.

in a diagram in whatsoever position of the control flow (perhaps as mutually exclusive choices) or none should, but also the more stringent constraint that each actual pizza production process execution must contain both or none. A similar reading holds for a historical or a causal dependence.

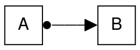
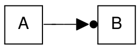
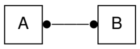
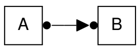
Since dependence expressions have effects on finite execution traces, a way to characterise (some of) their effects on process executions is to describe them using Linear-time Temporal Logic (LTL$_f$) with $f$inite execution semantics [6]. $\textsc{Cooc}(T_i, T_j)$ states that either T_i and T_j co-occur in a process execution or they both do not appear. This corresponds, in LTL$_f$ to the formula $\Diamond T_i \leftrightarrow \Diamond T_j$. $\textsc{Hist}(T_i, T_j)$ states that the execution of T_j necessarily requires a previous execution of T_i. An occurrence of T_i, nonetheless does not depend upon T_j. In particular, when T_j is not present in the trace, T_i can either occur or not. This corresponds, in LTL$_f$ to the formula $\neg T_j \mathcal{W} T_i$. $\textsc{Cause}(T_i, T_j)$ states that the execution of T_j necessarily requires a previous execution of T_i and the previous execution of T_i is necessary to explain the execution of T_j. Thus both T_i and T_j must occur in the execution in this order (or none of them does). This corresponds, in LTL$_f$ to the formula $\neg T_j \mathcal{W} T_i \wedge \Box(T_i \rightarrow \Diamond T_j)$. Given this interpretation of dependence expressions, we can note that a causal dependence enforces also a goal-based co-occurrence and a historical dependence.

Note that the characterisation of dependence expressions provided above only concerns some necessary temporal properties that these expressions should enforce upon a process execution. A formal characterisation of historical dependence, causal dependence and goal-based co-occurrence, that takes into account also their ontological nature is left for future work.

Incorporating Dependence Expressions in (Hybrid) Process Models. Dependence expressions are not meant to be used on their own. Instead, they are thought of as expressions that complement a business process model and provide the ability to capture aspects from the real world (including the social world and goal oriented aspects) that otherwise would be lost. In particular, in case of procedural process models, such as BPMN models or WF-nets, we envisage a model of a real process P as composed of two separate (but related) parts: a procedural model (diagram) and a set of dependence expressions. This proposal is in line with several recent work in the BPM field (see e.g., [18, 7]) where so-called *hybrid models* are introduced as a way to combine a procedural component that describes all the allowed control flows in an imperative manner and a declarative component that describes only what should not be violated. The two parts are kept separated so as not to hamper the perceptual discriminability of the various model elements [20].

Given the characterisation of $\textsc{Cooc}(T_i, T_j)$, $\textsc{Hist}(T_i, T_j)$, and $\textsc{Cause}(T_i, T_j)$ in terms of LTL$_f$ one may consider the idea of exploiting the declarative language DECLARE [23] to represent dependence expressions. Indeed, it is easy to note that the interpretation of the three expressions provided here creates a correspondence between $\textsc{Cooc}(T_i, T_j)$, $\textsc{Hist}(T_i, T_j)$, and $\textsc{Cause}(T_i, T_j)$ and the DECLARE patterns co-$existence(T_i, T_j)$, $precedence(T_i, T_j)$, and $succession(T_i, T_j)$, respectively (see Table 1, where the graphical notation and the formalisation in terms of LTL$_f$ of relevant DECLARE patterns is proposed). The exploitation of DECLARE would leverage an existing modelling language, thus avoiding the burden of a new notation. Moreover, the investiga-

Table 1. Graphical notation and LTL formalisation of some Declare templates.

TEMPLATE	FORMALIZATION	NOTATION	DESCRIPTION
response(A,B)	$\Box(A \rightarrow \Diamond B)$		If A occurs, B must eventually follow
precedence(A,B)	$\neg B \, \mathcal{W} \, A$		B can occur only if A has occurred before
co-existence(A,B)	$\Diamond A \leftrightarrow \Diamond B$		If B occurs, then A occurs, and viceversa
succession(A,B)	response (A,B) $\wedge$ precedence(A,B)		A occurs if and only if it is followed by B

tion proposed here could be seen as a sort of ontological grounding of specific DECLARE patterns. Nonetheless, we prefer not to commit to this proposal in this paper. In fact, flattening e.g., a causal relation onto a succession pattern would have three undesirable consequences: first, it would overload the meaning of DECLARE patterns with notions that are outside DECLARE (the notion of causality in this case); second, it would reduce ontological dependence to mere temporal patterns; third, it would 'transfer' to ontological dependences entailments that are only valid for temporal patterns. As an example, while $co\text{-}existence(T_i, T_j)$ and $precedence(T_i, T_j)$ entail $succession(T_i, T_j)$, it would be incorrect to state that a goal-based co-occurrence and a historical dependence between two activities also force the validity of a causal dependence among them.

Nevertheless, the formalisation of dependence expressions in terms of LTL$_f$ enables us to leverage existing techniques and tools (e.g., [16, 7]) for the automated check and repair of a procedural model with respect to dependence expressions, at least for what concerns their temporal characterisation.

5 Application Scenarios

In this section we describe two application scenarios which could benefit of the analysis carried out in the previous sections: business process documentation and business process redesign.

5.1 Business Process Documentation

Business process models are often used by organizations as a means for documenting the procedures carried out. However, the information contained in the model sometimes is not enough in order to make clear the reasons why some parts of the process model have been designed in a certain way.

Let us consider a realistic scenario of an *Intake process for elderly patients with mental problems*, inspired by the procedure reported in [9] that describes the process carried out in a healthcare institution of the Eindhoven region. The *Intake* process starts when the institute receives a notice by the family doctor of the person who needs the treatment. The notice is answered, recorded and printed. The patient's folder is retrieved, if it already exists, or it is created, if the patient has never been registered in the healthcare information system, and the notice added to the patient's folder. Two intakers (a social-medical worker and a physician) are then assigned to the patient and the assignments stored in the system. Two cards containing information about the patient, one per intaker, are printed and handed out. Meanwhile, if needed, the medical file of the patient is requested to the patient's doctor and, whenever it is received, the document is added to the patient's folder. Once the medical file is available for the appointment, the patient can meet the intakers and is asked to pay the ticket. At the end of each of the two meetings, the patient's folder is enriched with the new information acquired by the intakers. When the documentation by each of the two intakers has been collected, it is evaluated and a treatment for the patient decided.

Fig. 2 reports the *Intake* process described in BPMN and annotated with some hypothetical activity cycle time (including both processing and waiting time) as well as with the probability distribution of the alternative branches.

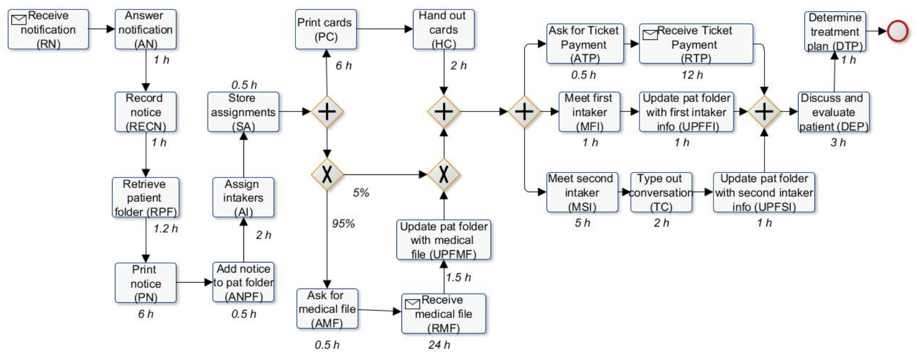

Receive notification (RN)
Answer notification (AN)
1 h
Record notice (RECN)
1 h
Retrieve patient folder (RPF)
1.2 h
Print notice (PN)
6 h
Add notice to pat folder (ANPF)
0.5 h
Assign intakers (AI)
2 h
Store assignments (SA)
0.5 h
Print cards (PC)
6 h
Hand out cards (HC)
2 h
5%
95%
Ask for medical file (AMF)
0.5 h
Receive medical file (RMF)
24 h
Update pat folder with medical file (UPFMF)
1.5 h
Ask for Ticket Payment (ATP)
0.5 h
Receive Ticket Payment (RTP)
12 h
Meet first intaker (MFI)
1 h
Update pat folder with first intaker info (UPFFI)
1 h
Meet second intaker (MSI)
5 h
Type out conversation (TC)
2 h
Update pat folder with second intaker info (UPFSI)
1 h
Discuss and evaluate patient (DEP)
3 h
Determine treatment plan (DTP)
1 h

Fig. 2. *Intake* process of a healthcare institute

Let us assume that a new director has been appointed, and she has been provided with the institute business process models in order to get familiar with the procedures carried out in the institute. When looking at the *Intake* process model in Fig. 2 (in which data objects are not reported to ease the readability, and activity labels, as often happens, are not extremely informative), she is only able to grasp the execution ordering of the activities currently carried on in the institute, while missing other types of dependences among them. This lack of information could result in possible misunderstandings of the process model as well as of what it represents. By only looking at the model, she may ask the reason why in the model the activity `Assign intakers` occurs before the activity `Update pat` (ient) `file with first intaker information`.

Table 2 reports the dependence expressions identified among the activities of the *Intake* process. Some of the dependences are real-world ones, i.e., they depend on *laws of nature*, others relate to the *business goal* of the process, while others pertain to *norms*. The dependence expressions are grouped accordingly in Table 2.

Among the *law-of-nature* dependences, a historical dependence can be identified between the activities `Record notice` and `Print notice`. Intuitively, printing a notice demands for a state of the world in which the notice is in an electronic format, i.e., it requires that it has been (electronically) recorded. Similarly, a historical dependence exists between the `Retrieve patient folder` and all the activities that demand for the existence of the folder in order to be executed (i.e., `Add notice to patient folder`, `Update pat. folder with medical file`, `Update pat. folder with first intaker info`, `Update pat. folder with second intaker info`). A historical dependence also exists between the activities `Print cards` and `Hand out cards`, as handing out card demands for a state of the world in which the cards have been printed out. Few causal dependences can also be identified, as for instance between the activities `Receive notice` and `Answer notice` (the notice answer is caused or explained by the notice receipt), between the activities `Ask for medical file` and `Receive medical file` (the receipt of the medical file is caused by the request of the file to the doctor) and between the activities `Ask for ticket payment` and `Receive ticket payment` (the payment reception is caused by the payment request).

Among the *business goal* dependences, a goal-based co-occurrence can be identified between the activities `Receive ticket payment` and `Determine treatment plan`. Indeed, due to the business nature of the *Intake* process, in order to get the process accomplished, both determining the treatment plan for the patient and getting the ticket paid for the service are necessary activities. Removing the occurrence of one of the two activities would change the process into a different one. However, the two activities are not bound by any temporal constraint. Similarly, for the goal-based co-occurrence between the activities `Receive ticket payment` and `Discuss and evaluate patient info`. Moreover, a historical dependence can be identified between the activities `Discuss and evaluate patient` and `Determine treatment plan`. Indeed, in an *Intake* process, a decision on the treatment plan of a patient cannot be taken, unless the patient's information has been carefully evaluated. Last but not least, a causal dependence relationship holds between the activity `Receive notice` and the activity `Discuss and evaluate patient`. The discussion and evaluation of the patient is indeed triggered (in an *Intake* process) by the request to start an intake procedure. Similarly for the causal dependence between the activities `Receive notice` and the activity `Determine treatment plan`.

Finally, among the norm-based dependence expressions, two historical dependences can be identified (between the pair `Assign intakers` and `Update pat. folder with first intaker information` and between the pair `Assign intakers` and `Update pat. folder with second intaker information`). Indeed, an intaker is allowed to report information in the patient folder only if she has been appointed to do it, i.e., a historical dependence relationship holds between the two activities (and, hence, the latter cannot occur before the former).

The additional information that the dependence expressions are able to provide, makes it clear to the new director that a dependence relationship holds between the activities `Assign intakers` and `Update pat. folder with first intaker information`, as well as the reason why they have to occur in that specific order. Hence, making explicit these dependences helps the new director to understand why the procedure has been designed as it is.

Ontological dependences			
Law-of-nature	Hist(RECN,PN)	Hist(RPF,ANPF)	Hist(RPF,UPFMF)
	Hist(RPF,UPFFI)	Hist(RPF,UPFSI)	Hist(PC,HC)
	Cause(RN,AN)	Cause(AMF,RMF)	Cause(ATP,RTP)
Business Goal	Cause(RN,DEP)	Cause(RN,DTP)	Hist(DEP,DTP)
	Cooc(RTP,DTP)	Cooc(RTP,DEP)	
Norm	Hist(AI,UPFFI)	Hist(AI,UPFSI)	

Table 2. Dependence expressions characterizing the *Intake* process.

5.2 Business Process Redesign

It is often the case that business process models need to be redesigned. This can be due to different reasons e.g., because the world, the organization or the procedure they describe changes, or for optimization reasons. Several approaches and techniques have been investigated in the BPM community in order to support business analysts in business process redesign (see e.g., [9, 24]).

Let us assume, that the new director of the healthcare institute, in order to better understand the efficiency of her institute, has appointed a business analyst to analyze the processes carried out in the institute. By analyzing the process under the perspective of evaluating its cycle time, the business analyst notices that the process presents some bottlenecks. Indeed, the activities `Print notice`, `Receive Payment Ticket` and `Receive medical file` have a high average duration time (6, 12 and 24 hours, respectively). In the first case, the high duration time is due to the fact that only one printer is available in the institute, while in the second and in the third case this is due to the response time required by patients and medical doctors to pay the ticket and to provide the medical file, respectively. Moreover, although in the last case the request of the file from the doctor is optional, it is needed in 95% of the cases. This causes a high average process cycle time[7] ($= 53.4$h). In order to solve the issue, the institute director, at the suggestion of the business analyst, decides to redesign the process.

[7] The computation of the average process cycle time is based on flow analysis [9] and depends on the structure of the process. In this case, the average time required for a process execution is given by the average time required by: (i) the sum of the time required by the activities in sequence before the first split AND gateway, which is, in turn, given by the sum of the average times of the activities in sequence ($(1 + 1 + 1.2 + 6 + 0.5 + 2 + 0.5)h=$ 12.2h); (ii) the sum of the times required by the most costly branches of the two AND blocks, i.e., the one dealing with the optional request to the doctor of the medical file and the one related to the ticket payment receipt. The former is computed as the weighted (with the corresponding probabilities) average of the two alternative branches between the XOR split and the XOR join, (i.e., $((0.95 * (0.5 + 24 + 1.5)) + (0 * 0.05))h=$ 24.7h), while the second is the sum of the average cycle time of the activities `Ask for ticket payment` and `Receive ticket payment`, (i.e., $(0.5 + 12)$h=12.5h), respectively; and (iii) the time required by the last two activities (i.e., $(3+1)$h=4h). The average cycle time is hence $(12.2+24.7+12.5+4)$h$=$ 53.4h.

In order to reduce the overall cycle time of the procedure, the business analyst suggests to apply two business process behaviour heuristics: *parallelism* and *resequencing* [9]. While the first heuristic consists of evaluating what “can be executed in parallel”, the second one consists of “moving the activities to more convenient places” [9]. According to the process re-design heuristics, the business analyst suggests to (i) parallelize the printing of the notice and the enrichment of the patient file up to the storing of the intaker assignments; (ii) anticipate the request of the payment to the patients and the request of the medical file to the doctor. Fig. 3 shows the redesigned model. Such a redesign allows the healthcare institute to save about 16.5 hours of average cycle time by reducing the cycle time from 53.4 to 36.9 hours - as most of the flow related to the notice management and to the intaker assignment is actually in parallel with the costly time required for waiting for the medical file.

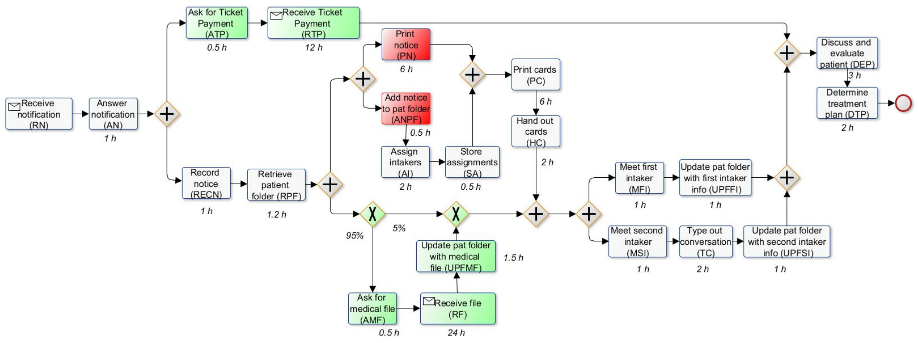

Receive notification (RN)
Answer notification (AN)
1 h
Ask for Ticket Payment (ATP)
0.5 h
Receive Ticket Payment (RTP)
12 h
Print notice (PN)
6 h
Add notice to pat folder (ANPF)
0.5 h
Assign intakers (AI)
2 h
Store assignments (SA)
0.5 h
Print cards (PC)
6 h
Hand out cards (HC)
2 h
Record notice (RECN)
1 h
Retrieve patient folder (RPF)
1.2 h
95%
5%
Update pat folder with medical file (UPFMF)
1.5 h
Ask for medical file (AMF)
0.5 h
Receive file (RF)
24 h
Meet first intaker (MFI)
1 h
Update pat folder with first intaker info (UPFFI)
1 h
Meet second intaker (MSI)
1 h
Type out conversation (TC)
2 h
Update pat folder with second intaker info (UPFSI)
1 h
Discuss and evaluate patient (DEP)
3 h
Determine treatment plan (DTP)
2 h

Fig. 3. *Intake* process redesigned according to the analyst’s suggestions

However, by looking at the dependence expressions reported in Table 2, the business analyst can easily notice that, while anticipating the request of the medical file to the doctor and the ticket payment to the patient (depicted in green in the diagram) does not violate any of the identified dependences, this is not the case for the parallelization of the printing notice and the enrichment of the patient folder (marked in red). Indeed a historical dependence relationship holds between the activity `Add notice to the patient folder` and the activity `Print notice`, so that swapping them would result in an incorrect model.

Automated Check of Dependence Expression Enforcement. As reported in Section 4, the formalisation of dependence expressions in terms of LTL_f, allows us to take advantage of existing works (e.g., [16, 7]) for the automated check of the enforcement of declarative properties or rules on procedural models, explanation of possible violations and repair actions. For instance, in the scenario described above, these techniques can be leveraged by the analyst to detect the inconsistency between the ontological dependence expressions and the redesigned process model, thus enabling the application of the only redesign heuristics that do not violate any dependence expression.

6 Related work

We can roughly classify the literature related to this paper into three main groups: (i) works dealing with the analysis of business process model notations and its elements; (ii) works leveraging ontological analysis of business process modelling notations and its elements; and, finally, (iii) works combining declarative and procedural models.

Several papers in the literature focus on the analysis of the *elements* involved in business processes and business process modelling languages. Many of these works provide a comparison of different modelling notations [15, 29] or develop metamodels of business process models across notations [14, 19]. Other works, instead, take an ontological perspective to achieve the same goal. Indeed, some of them use ontologies for guiding the development of conceptual models and domain ontologies [4] or for semantically enriching business process models [25, 12, 21], while others provide upper-level ontologies for business processes [22].

The second category of works leverages ontological analysis to deal with business process notations and business process model elements. Within this category, we can find works using the ontological analysis of business process elements (e.g., participants) across notations, such as [2]. In [26] the authors offer an ontological analysis of BPMN 2.0 elements and choreography diagram elements, respectively, with particular emphasis on the ontological characterization of BPMN events and activities. In [3] an effort towards a semantic foundation of the notion of *role* in the enterprise is provided. However, none of these works deals with the analysis of dependences among activities.

Indeed, although many efforts have been carried on so far in order to characterize ordering relationships between business process activities, an ontological analysis of these dependences has not been proposed yet. The analysis presented in this paper has been stimulated by philosophical and ontological papers like [5], [10] and [17] which are strongly focused on defining and classifying ontological dependences, and where distinctions like weak vs rigid, ontological vs existential dependence are presented. Dependence as a schema is further discussed in [27] where an initial list of qualifications is also attempted (financial, practical, physiological, functional, ontological, logical and so on). Investigations on ordering relationships between activities are present also in the BPM community. An example is [8], where a definition of causal relation has been proposed as a sequence of events that can not be ordered in the opposite direction. Nevertheless, none of these works explicitly deals with ontological dependences in the context of business processes.

Several works combining declarative and procedural models have been recently investigated in the literature, some of which also provide automated support to deal with such a combination. Examples of the latter are works dealing with the automated discovery of hybrid process models [28, 18], the automated check of a declarative formula on a procedural model [16], as well as the automated enforcement of the declarative component on the procedural one [7].

7 Conclusions

Existing business process modelling notations mainly focus on the representation of a specific kind of relationship between activities, that is their execution ordering within the control flow. However, the relationships between their activities of real-world processes are much richer and go beyond such a privileged relationship, covering relational constraints of different nature (e.g., ontological ones). In this paper we provided a characterisation of three ontological relationships (a.k.a. dependences) between business process activities: historical dependence, causal dependence, and goal-based co-occurrence. We introduced a language (for expressing them), made a proposal on how to incorporate them in business process models by adopting a hybrid approach, and showed their importance by discussing two application scenarios.

In the future, on the one hand, we would like to further investigate the ontological dependences between business process activities, by analysing the role and the ontological implications that business process participants (e.g., data objects, actors) have on the characterization of these ontological dependences; on the other hand, we are interested to extend our exploration of ontological relationships also to the relationships between activities and other types of business process participants.

Acknowledgments This research has been partially carried out within the Euregio IPN12 KAOS, which is funded by the "European Region Tyrol-South Tyrol-Trentino" (EGTC) under the first call for basic research projects.

References

1. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. 16(9), 1128–1142 (2004)
2. Adamo, G., Borgo, S., Di Francescomarino, C., Ghidini, C., Guarino, N., Sanfilippo, E.M.: Business processes and their participants: An ontological perspective. In: Proc. of the 16th Int. Conf. of the Italian Association for Artificial Intelligence (AI*IA 2017). LNCS, vol. 10640, pp. 215–228. Springer (2017)
3. Almeida, J.P.A., Guizzardi, G., Santos Jr, P.S.: Applying and extending a semantic foundation for role-related concepts in enterprise modelling. Enterprise Information Systems 3(3), 253–277 (2009)
4. Benevides, A.B., Guizzardi, G.: A model-based tool for conceptual modeling and domain ontology engineering in ontouml. In: International Conference on Enterprise Information Systems. pp. 528–538. Springer (2009)
5. Correia, F.: Ontological dependence. Philosophy Compass 3(5), 1013–1032 (2008)
6. De Giacomo, G., De Masellis, R., Montali, M.: Reasoning on LTL on finite traces: Insensitivity to infiniteness. In: Proc. of the 28th AAAI Conference on Artificial Intelligence. pp. 1027–1033. AAAI Press (2014)
7. De Masellis, R., Francescomarino, C.D., Ghidini, C., Laponin, A., Maggi, F.M.: Rule propagation: Adapting procedural process models to declarative business rules. In: 21st IEEE International Enterprise Distributed Object Computing Conference, (EDOC 2017). pp. 165–174. IEEE Computer Society (2017)
8. Desel, J.: Validation of process models by construction of process nets. In: Business Process Management, Models, Techniques, and Empirical Studies. LNCS, vol. 1806, pp. 110–128. Springer (2000)
9. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer Publishing Company, Incorporated (2013)
10. Fine, K.: Ontological dependence. Proc. of the Aristotelian Society 95(n/a), 269–290 (1994)

11. Galton, A.: States, processes and events, and the ontology of causal relations. In: Proceedings of the 7th Int. Conf. on Ontology in Information Systems (FOIS 2012). Frontiers in Artificial Intelligence and Applications, vol. 239, pp. 279–292. IOS Press (2012)
12. Ghidini, C., Di Francescomarino, C., Rospocher, M., Tonella, P., Serafini, L.: Semantics-based aspect-oriented management of exceptional flows in business processes. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42(1), 25–37 (2012)
13. Governatori, G., Rotolo, A.: Norm compliance in business process modeling. In: Semantic Web Rules. pp. 194–209. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
14. Heidari, F., Loucopoulos, P., Brazier, F., Barjis, J.: A meta-meta-model for seven business process modeling languages. In: Business Informatics (CBI), 2013 IEEE 15th Conference on. pp. 216–221. IEEE (2013)
15. List, B., Korherr, B.: An evaluation of conceptual business process modelling languages. In: Proc. of the 2006 ACM symposium on Applied computing. pp. 1532–1539. ACM (2006)
16. Lohmann, N., Fahland, D.: Where did I go wrong? - explaining errors in business process models. In: Proc. of the 12th Int. Conf. on Business Process Management (BPM 2014). LNCS, vol. 8659, pp. 283–300. Springer (2014)
17. Lowe, E.J.: The Possibility of Metaphysics: Substance, Identity, and Time. Clarendon Press (1998)
18. Maggi, F.M., Slaats, T., Reijers, H.A.: The automated discovery of hybrid processes. In: Proc. of the 12th International Conference on Business Process Management (BPM 2014). LNCS, vol. 8659, pp. 392–399. Springer (2014)
19. Mili, H., Tremblay, G., Jaoude, G.B., Lefebvre, É., Elabed, L., Boussaidi, G.E.: Business process modeling languages: Sorting through the alphabet soup. ACM Computing Surveys (CSUR) 43(1), 4 (2010)
20. Moody, D.L.: The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering 35(6), 756–779 (November/December 2009)
21. Natschläger, C.: Towards a BPMN 2.0 ontology. In: Proc. of the 3rd Int. Workshop on Business Process Model and Notation (BPMN 2011). LNBIP, vol. 95, pp. 1–15. Springer (2011)
22. Nicola, A.D., Lezoche, M., Missikoff, M.: An ontological approach to business process modeling. In: Proc. of 3rd Indian Int. Conf. on Artificial Intelligence. pp. 1794–1813 (2007)
23. Pesic, M., Schonenberg, H., van der Aalst, W.: DECLARE: Full Support for Loosely-Structured Processes. In: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007). pp. 287–300. IEEE Computer Society (2007)
24. Reijers, H.A., Liman Mansar, S.: Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. Omega 33(4), 283–306 (2005)
25. Rospocher, M., Ghidini, C., Serafini, L.: An ontology for the Business Process Modelling Notation. In: Proc. of 8th Int. Conf. on Formal Ontology in Information Systems (FOIS 2014). Frontiers in Artificial Intelligence and Applications, vol. 267, pp. 133 – 146. IOS Press (2014)
26. Sanfilippo, E.M., Borgo, S., Masolo, C.: Events and activities: Is there an ontology behind bpmn? In: Proc. of 8th Int. Conf. on Formal Ontology in Information Systems (FOIS 2014). Frontiers in Artificial Intelligence and Applications, vol. 267, pp. 147–156. IOS Press (2014)
27. Simons, P.: Parts: a Study in Ontology. Clarendon Press, Oxford, Oxford (1987)
28. Smedt, J.D., Weerdt, J.D., Vanthienen, J.: Fusion miner: Process discovery for mixed-paradigm models. Decision Support Systems 77, 123–136 (2015)
29. Söderström, E., Andersson, B., Johannesson, P., Perjons, E., Wangler, B.: Towards a framework for comparing process modelling languages. In: Proc. of 14th Int. Conf. on Advanced Information Systems Engineering (CAiSE). LNCS, vol. 2348, pp. 600–611. Springer (2002)
30. Weske, M.: Business Process Management. Concepts, Languages, Architectures. Springer (2012)