Seamless (and Temporal) Conceptual Modeling of Business Process Information

Carlo Combi and Sara Degani

Department of Computer Science - University of Verona
Strada le Grazie 15, 37134 Verona, Italy
carlo.combi@univr.it,sara.degani@gmail.com

Abstract. There are a number of aspects to be considered when modeling a business process, and research has mainly focused on the control flow perspective. The conceptual modeling of process-related information has been mainly left to traditional data models, like the ER model or UML class and object diagrams; these, however, are generic data models, and they do not offer means to express information features specific for business processes; as a result, the process structure and information are modeled as separate aspects. In this paper we define a connection between business process control flow and information perspectives at the conceptual level. We introduce the possibility to specify what is the key information of a business process, to associate tasks to information entities and relationships that are visible to the task itself, and to capture the effective task usage of process information in each single process case.

Key words: business process, temporal conceptual modelling, data perspective

1 Introduction

There are a number of aspects to be considered when modelling a business process; the most common are the control flow, the resource, and the information perspectives [6]. Research has mainly focused on control flow, and a wide range of process languages have been developed to better represent business process functional behaviour in various application domains. Resource and data modelling has received less attention. Conceptual modelling of process data, in particular, has not been much considered, and it has been mainly left to traditional data models like the widespread ER model or the UML class and object diagrams [4,5]. These, however, are generic data models that can be adopted in various domains; therefore, they allow the designer to model information entities and their relationships, but they do not offer means to express information features specific for business processes, as which data a task has access to. The result is that the process structure and process data are modelled as two separate aspects at the conceptual level, and the effective usage of data in a business process is only expressed at the implementation level, usually by means of process variables.

When modelling business process information, moreover, special attention should be given to temporal aspects [3]. Two relevant temporal dimensions in particular should be considered, namely valid time and transaction time: valid time allows the management of possible changes on process information, while transaction time captures the history of process data. The traditional ER model or UML class diagrams do not offer explicit means to represent these temporal dimensions; a temporal extension of the models would allow a more complete conceptual modelling of process information.

In this paper we define a connection between business process control flow and information perspectives, allowing the designer to capture information features specific for business processes, and to conceptually model process data jointly to their associations with the conceptual process schema. In particular, we will propose a way of supporting the joint conceptual design of business processes and (temporal) data through some notations that allow one to graphically highlight and characterize data relevant for a given task. Moreover, it is possible to specify how (and which) data have to be dealt with when executing either a single task or the overall business process. In this direction, our proposal may be considered as complementary and more design-oriented with respect to the distinction in the WfMC reference model between workflow-relevant data and workflow-application data, where the first ones are data the WfMS has to deal with in order to manage the workflow execution, while the second ones are related to the application domain tasks have to consider [7].

In this paper, we adopt the well-known YAWL notation [1] to model the control flow perspective; moreover, we make use of a temporal extension of the ER model called TimeER [2] to model business process schema of data. The connection between the two perspectives is achieved by introducing the notions of *core entity* and *case instance*, that allow one to identify the key information entities of a business process on a TimeER diagram, and the corresponding instances that are managed by a specific process case. Then, *view associations* are introduced between each single process task and TimeER entities and relationships that are visible to the task itself. Finally, constraints are defined over view associations, that characterize the entity (relationship) instances the task has access to, the type of access and the view cardinality.

The paper is structured as follows. Section 2, describes a business process modelled with the YAWL notation and the associated TimeER diagram as a motivating scenario; Section 3, introduces the main contribution of the paper, i.e., the notions of *core entity*, *case instance*, *task view* and task view constraints; finally, Section 4 illustrates concluding remarks.

2 A Motivating Scenario

In this section we introduce a case study taken from the the clinical domain of radiological reporting [8]. We first describe a clinical process that represents the execution of radiological exams on patients; therefore, we analyze the clinical information this process needs to consult and record in order to execute all the tasks it is composed of.

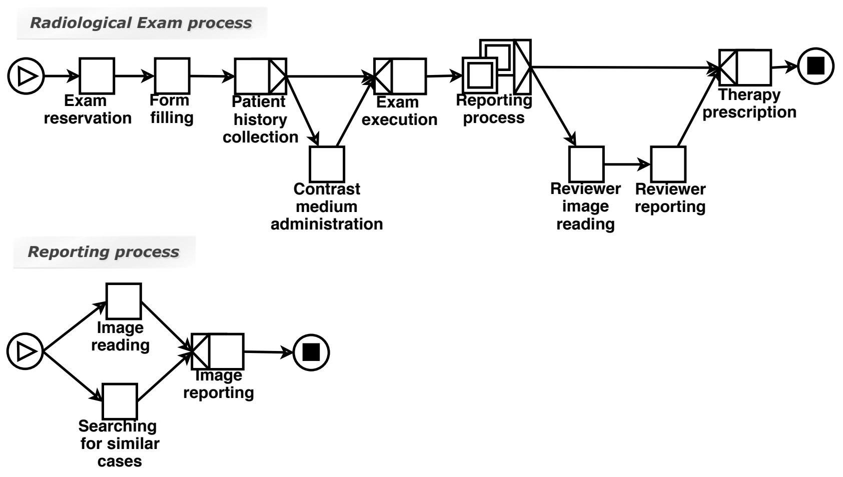

Fig. 1. The workflow schema *Radiological Exam process*

The clinical process *Radiological Exam process* in Figure 1 represents the execution of radiological visits on patients. The process starts with the collection of exam reservations from patients (Exam reservation task): date availability for the exam is checked and an appointment is fixed; if not already recorded, the patient personal data are stored in the clinical database; therefore, exam data (reason, urgency, type of exam, date) are recorded.

On the day fixed for the exam, the patient fills a form with personal data and information about previous interventions, possible pathologies, drug allergies and the drug itself if not already stored in the database (Form filling task).

The patient is then visited by the physician, who collects information about the patient clinical history, including possible symptoms (Patient history collection task). A contrast medium is administered to the patient if required for the visit (Contrast medium administration task); afterwards, the visit is performed, radiological images are produced and information about the visit are recorded (Exam execution task).

Then, a double reporting process starts (Reporting process task). Two different physicians read the radiological images produced during the radiological visit, consult similar cases and formulate a diagnosis; finally they create two different reports. If physicians have discordant opinions, a reviewer is consulted that reads the radiological images (Reviewer image reading task) and creates a third report (Reviewer reporting task). Finally, a therapy is prescribed for the patient on the basis of the visit reports and images, considering patient's drug allergies, and consulting patient's past prescriptions and (possibly) other patients' therapy prescriptions (Therapy prescription task).

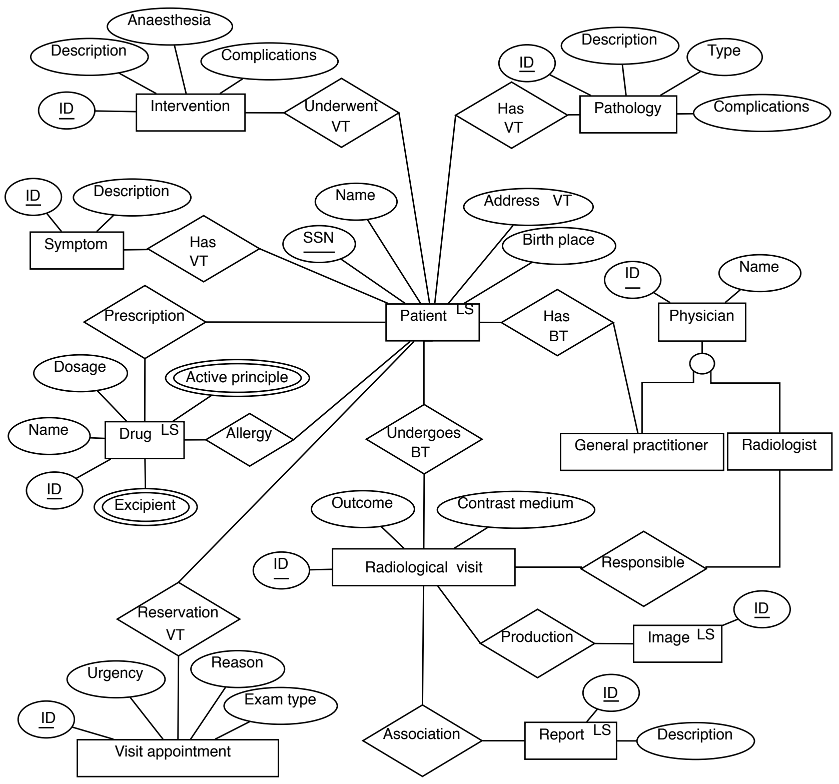

Fig. 2. The TimeER diagram modelling information for process in Figure 1. The usual ER notation (i.e., box for entity diamond for relationship, oval for attribute, double oval for attribute with multiple values, underlined attribute names for attributes participating to a key) is extended with the specification of the supported temporal dimension (BT for bitemporal, VT for valid time, and LS for lifespan).

Figure 2 shows the TimeER diagram describing the information related to the radiological exam process. The process manages information about patients, medical visits, physicians, radiological images and medical reports.

A patient is identified by an SSN (Social Security Number) and is characterized by name, address and birth place. Possible changes of a patient's address are recorded too. The process needs to store information about the patient's clinical history, like interventions the patient underwent, pathologies, and possible symptoms. For each intervention, a description, possible complications, the type of anaesthesia and the date are necessary; for each pathology, a description, the pathology type and possible complications are recorded; for each patient symptom, a brief description is given. Drug prescriptions for a patient are stored; for each drug, the name, the dosage, the active principle, excipients and the drug validity. The process needs also to record information about patient allergies to drugs.

A patient has a general practitioner, that could vary over time, for which the name and the email are recorded, together with the time when this information is current in the database. The process needs information about visit reservations (reason, urgency, type of exam, date); moreover, it needs information about the performed radiological visit (contrast medium, outcome, responsible radiologist), and also about the time of execution and the time during which this information is current in the database. Finally, the process stores the images produced during the visit and physician reports, together with the time during which images and reports have been created.

With respect to the described scenario, some requirements arise at the conceptual design level: (i) to link, even graphically, each workflow schema to the information required to correctly execute it: for example, we would be able to specify that the any execution of the workflow schema *Radiological Exam process* should produce a new instance of the entity *Radiological visit*; (ii) to characterize how single entities/relationships are involved in the execution of tasks: for example, we would specify that past and current occurrences of entity *Drug* and of relationship *Prescription* related to other patients could be read during the workflow related to a given patient; (iii) to specify, possibly in a graphical way, those data that have to be inserted into the database when a workflow/task is executed: for example, we would specify that the execution of task *Therapy prescription* has to produce some new information about drug prescriptions for the considered patient.

In the following, we will introduce a seamless approach to the conceptual design of data-centric business processes, that fulfils the requirements previously sketched, allowing the designer to specify several details and features related to the multifaceted "connections" between processes, tasks, and the related, possibly complex, data.

3 Conceptual Modelling of Business Process Information

A TimeER diagram allows the designer to represent the information used by a business process in terms of entities and relationships; in other words, it captures how the process information is structured, and the corresponding temporal aspects. Such a model however is not expressive enough to show the relevance of the different pieces of information with respect to business process goals, and how this information is associated with the process tasks. In this section we introduce a connection between the control flow model of the process, represented with the YAWL notation, and the informational perspective of the process itself, represented by a TimeER diagram. More specifically, we introduce the notions of *core entity* and *case instance*, that allow the designer to specify what is the key information of a business process. Therefore, we define the notion of *task view*, that indicates what is the information a task has access to, in order to be properly performed; finally, we introduce a set of constraints expressing how the key information guides the whole process.

3.1 Core Entities and Case Instances

A business process describes a set of activities that are performed in an organization to achieve a specific business goal. Process activities need to manage information to be performed. Not every information entity managed by a process has the same relevance: key information entities can be identified, that keep track of the business operations and capture process goals. In an order management process, for example, the key information entity is the customer order; indeed, each activity in the process performs actions that are directed to manage a specific customer order. Considering our motivating example, we observe that each case of the process manages a single radiological exam, and each task manages information related to that specific radiological exam. For example, the *exam reservation* task collects the personal data of the patient that undergoes the exam; the *Therapy prescription* task stores prescriptions for the patient that undergoes the exam. We can therefore state that the key information entities of this process are those that describe the radiological exam.

Key information entities of a business process can be identified thinking about: entities that are significant to the whole process, not only to single activities; entities that are managed by most activities; entities that are necessary for the process to start (as in the case of a customer order); entities that represent the service or product provided by the process (as in our motivating example); entities that have to be defined for the process to end correctly.

Considering our modelling framework, key information entities can be identified on the TimeER diagram describing the informational perspective of a process. In the TimeER diagram of our motivating scenario, radiological exams are represented by the entity *Radiological visit*. We call an entity that represents the key information managed by a business process a process *core entity*.

A process case manages one or more instances of the core entity. We call the core entity instances that are managed by a case *case instances*. As an example, a case of the radiological exam process manages a single exam for a specific patient; therefore, it creates only one instance of *Radiological visit*, that is its case instance for the current case. More in general, a process may have one or more core entities, and one or more case instances for each core entity.

Figure 3 shows an example of how the core entities of a business process, and the corresponding number of case instances, can be identified on a TimeER diagram: a box with the label *CE(n)*, where n is the number of case instances of the entity, is drawn below the rectangle representing the core entity itself. In the example, the entity *Radiological visit* is marked with the label *CE(1)*; this means that the entity is a core entity with a single case instance.

Information related to a core entity is represented by other entities that are directly involved in a relationship with the core entity, or that are indirectly related to it through a path of relationships. In our example, the personal data of a patient are represented by the entity *Patient*, that is directly involved in the relationship *Undergoes* with *Radiological visit*; the entity *Drug* records information about drugs, and it is indirectly related to *Radiological visit* through the path composed of the relationships *Prescription* and *Undergoes*. Information related

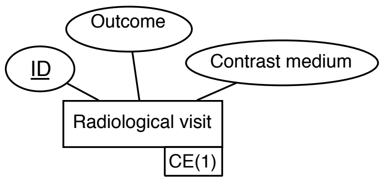

Fig. 3. Core entity graphical representation

to a core entity can be represented also by relationships that directly involve the core entity, or that are indirectly connected to it through a path of other relationships. For example, drug allergies of the patient that undergoes the exam are represented by the relationship *Allergy*, that is connected to *Radiological visit* through the path composed of the relationship *Undergoes*.

Core entities and case instances allow one to characterize the instances of other entities and relationships, and to identify those instances that represent the information managed by a specific process case. A simple entity (i.e., an entity that is not core) may include instances that are involved in a relationship with a case instance, that we call *case-instance-related*, and instances that are not involved in a relationship with a case instance, called *not-case-instance-related*. The relationship may be direct, if a TimeER relationship between the core entity and the simple entity exists, or indirect, if a path of TimeER relationships between them exists. For example, the case-instance-related instances of *Drug* are those that describe the drugs prescribed to the patient that undergoes the radiological exam represented by the current case; all the other instances are not-case-instance-related.

As regards relationships, we can distinguish between instances that involve a case instance (if the relationship directly involve the core entity) or that are indirectly connected to a case instance through a path of relationships, called *case-instance-involving*, and instances that do not involve a case instance, termed *not-case-instance-involving*. For example, the instance of *Undergoes* that connects a patient to the radiological visit performed in the current case is case-instance-involving; all the other instances are not-case-instance-involving.

3.2 Task View and View Constraints

Each task of a business process may be associated with a part of the TimeER diagram that describes the information that is necessary for the task to perform its job. The set of entities and relationships that are included in this part represents the informational view of the process task, that we call *task view*. The set of entities and relationships that are not included in the task view cannot be managed by the task itself.

We say that a process task is in a *view association* with each of the entities and relationships that are included in the task view. For each view association, it is possible to specify a set of *view constraints* that indicate: (i) the type of access to the information the entity (relationship) represents; (ii) the view cardinality; (iii) the group of instances the task has access to.

The type of access to an entity (relationship) can be *read* or *write*. If a task has read access to an entity (relationship), then it can read the values of a group of attributes of the instances the task has access to; if the task has write access to an entity (relationship), it can create an instance of the entity (relationship) or it can modify the value of a set of attributes of the entity (relationship).

The cardinality values that can be specified for a view association are the following: $(0,1)$ - the task has access to at most one instance of the considered entity (relationship); $(1,1)$ - the task has to access to one and only one instance of the considered entity (relationship); $(0,N)$ - the task has access to at most N instance of the considered entity (relationship); (M,N) - the task has access to at least M and at most N instance of the considered entity (relationship).

For each entity (relationship) that is involved in a view relation with a process task, it is then possible to restrict the group of instances the task has access to, by means of the instances characterization described in the previous section: if the considered TimeER element is a core entity, it is possible to specify that the task has access to the case instances only, or to all the instances but the case ones; if it is a simple entity, it is possible to specify that the task has access to the case-instance-related instances only, or to all the instances but the case-instance-related ones; finally, if the TimeER element is a relationship, it is possible to specify that the task has access to the case-instance-involving instances only, or to all the instances but the case-instance-involving ones. We observe that if none of these restrictions is specified, then the task has access to all the instances of the entity (relationship).

A view constraint is therefore expressed in the form: $< access\_type >$ $[(< min\_card >, < max\_card >)][< instance\_group >][< path >]$ where: $access\_type \in \{read, write\}$; $min\_card \in \{0, \ldots, N\}$, $N \in \mathbb{N}$; $max\_card \in \{1, \ldots, N\}$, $N \in \mathbb{N}$; $instance\_group \in \{CI, NCI, CIR, NCIR, CII, NCII\}$ (the labels stand for *case instance*, *not case instance*, *case instance related*, *not case instance related*, *case instance involving*, and *not case instance involving* respectively); *path* is a string composed of relationship names separated by a semicolon and represents the path of relationships that is intended to be followed to reach the core entity, starting from the entity (relationship) involved in the constraint.

As an example, we define view associations and view constraints for the task *Therapy prescription* of our motivating example. The task reads the personal data of the patient that is examined during the current process case; this is expressed defining the view constraint *read (1,1) CIR* on a view association between the task and the entity *Patient*; the constraint indicates that the task reads the (only) instance of the entity *Patient* that is related to the case instance of *Radiological visit*. Then, the task can read drug prescriptions, but only those of patients that are not the patient visited in the current visit; this is expressed by the constraint *read (0,N) NCII Undergoes* on a view association between the task and the relationship *Prescription*. On the same view association, the constraint *write (1,N) CII Undergoes* indicates that the task has to write at least one instance of *Prescription*, and this instance has to involve the patient related to the case instance; this means that the task has to create at least one prescription for the patient that underwent the radiological exam executed in the current process case. Moreover, the task may read any drug and drug allergies related to the current patient; this is expressed creating a view association between the task and the entity *Drug*, and defining the constraints *read (0,N)* and *read (0,N) CII Undergoes* on the view associations involving the entities *Drug* and *Allergy* respectively. The constraints indicate that the task can read any instance of *Drug*, and the instances of the relationship *Undergoes* that involve the patient that is related to the case instance. Finally, the task consults reports and radiological images produced during the execution of the radiological exam of the current process case; this is expressed by the constraints *read (1,1) CIR* on a view association between the task and the entities *Report* and *Image* respectively.

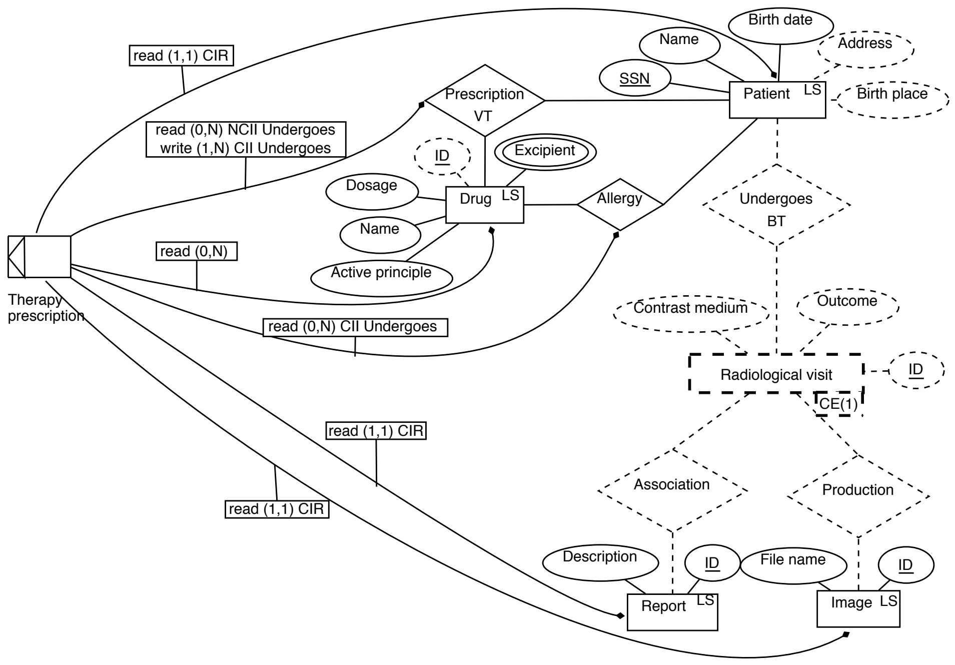

Fig. 4. Task view graphical representation

Figure 4 shows an example of how the view associations for a specific task and the corresponding view constraints can be graphically specified. The example represents the view for the task *Therapy prescription.* As we can see in the figure, each single view association between a process task and an entity (relationship) is shown graphically connecting the task to the entity (relationship) with a line, ending with a filled diamond. A box is connected to this line, containing the corresponding view constraints. Entity (relationship) attributes that are not visible to the task are modelled with a dashed oval; in a similar way, entities and relationship that are not visible to the task are represented with dashed shapes. We observe that the core entity is represented with a dashed rectangle; the reason is that the task does not manage core entity instances. It is however assumed that each task of the process knows the identity of the case instances. In the example shown in Figure 4 it is evident what is the core entity each constraint refers to. For example, the constraint *read (1,1) CIR* on the view association between *Therapy prescription* and *Report* refers clearly to the core entity *Radiological visit*. In some cases however, this could result unclear, if, for example, there are two different core entities, and a constraint is specified on a view association between a task and an entity involved in two different relationships with the two core entities. In this case it is necessary to specify what is the core entity the constraint refers to; this is achieved using a path, that specifies the name of the relationship that connects the entity involved in the view association to the desired core entity.

4 Discussion and Conclusions

In this paper we addressed the problem of how to create a connection between business process control flow and informational perspectives at the conceptual modelling level. Some notions have been introduced to characterize process information: process key informational entities are called core entities; core entity instances managed by a specific process case are called case instances; moreover, a task view relationships identify entities and relationships that can be managed by a specific process task.

References

1. W.M.P. van der Aalst and A.H.M. ter Hofstede: YAWL: Yet Another Workflow Language. Information Systems, vol. 30(4), pp. 245–275 (2005)
2. C. Combi, S. Degani, and C. S. Jensen: Capturing Temporal Constraints in Temporal ER Models. In: ER 2008. LNCS, vol. 5231, pp. 397–411. Springer (2008)
3. C. Combi, G. Pozzi: Temporal Conceptual Modelling of Workflows. In: ER 2003. LNCS, vol. 2813, pp. 59–76. Springer, (2003)
4. R. Elmasri, S.B. Navathe: Fundamentals of Database Systems, 2nd Edition, Benjamin/Cummings (1994)
5. G. Engels, A. Föorster, R. Heckel and S. Thöne: Process Modeling Using UML. In: Process-Aware Information Systems. John Wiley & Sons, Inc. (2005)
6. M. Weske: Business Process Management: Concepts, Languages, Architectures. Springer (2007)
7. Workflow Management Coalition, Hollingsworth, D.: The workflow reference model. http://www.wfmc.org/standards/framework.htm (1995)
8. J. Zhang, X. Lu, H. Nie, Z. Huang, W.M.P van der Aalst: Radiology information system: a workflow-based approach. International Journal of Computer Assisted Radiology and Surgery, vol. 4(5), pp. 509–516 (2009)