Mapping BPMN Process Models to Data Models in ORM

Herman Balsters

University of Groningen, Faculty of Economics and Business
P.O. Box 800 9700 AV Groningen, The Netherlands
h.balsters@rug.nl

Abstract. Business processes define workflow dependencies inside an industry and/or organization. Business processes drive machines and people, and use business data to function properly. By systematically integrating data and processes, we can understand and assess complete business processes from beginning to end. Practice, however, often reveals that there is no systematic link between a business process and associated business data. The aim of this paper to tackle some of the problems encountered in deriving business data from process models. We will show how to systematically map basic business process models using Business Process Modeling Notation (BPMN) to data models specified in ORM. From the resulting ORM model, we can generate a complete (corporate) relational database, containing the business data that is tailor-made to support the business process.

Keywords: Process models, data models, mapping process models to data models, fact-based modeling.

Introduction

Object-Role Modeling (ORM) is a fact-oriented approach for modeling information in terms of the underlying facts, where facts and rules may be verbalized in a language easily understandable by non-technical domain experts. In contrast to Entity-Relationship (ER) modeling [2] and Unified Modeling Language (UML) class diagrams [15], ORM models are attribute-free, treating all facts as relationships (unary, binary, ternary etc.). ORM does, however, include procedures (e.g. the RMap [13]) for mapping to attribute-based structures, such as those of ER or UML. For a basic introduction to ORM see [10], for a thorough treatment see [13]. We will use the term Fact-based modelling (FBM [13]) as the general name of several fact-based conceptual data modelling dialects, such as ORM, Natural language Information Analysis Method (NIAM) [11], and Fully-Communication Oriented Information Modeling (FCO-IM) [1].

Business processes define workflow dependencies inside an industry and/or organization. Business Process Model and Notation (BPMN) is a standard for specifying business processes providing a graphical representation based on workflow diagramming. BPMN 1.0 was developed by the Business Process Management Initiative (BPMI) and then was further developed by the Object Management Group

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 503–512, 2013.

(OMG); as of 2011, the current version of BPMN is 2.0 [5]. Business *processes* define the whole work-flow dependency inside an industry and/or organization, whereas Business *data* is data inside some *database*. Properly designed databases can handle the collecting of data that can guarantee the business data quality. By systematically integrating data and processes, we can –in principle- understand and assess complete business processes from beginning to end. Practice, however, all too often indicates that there is no systematic link between a business process and associated business data. In the case of BPMN, there are hardly any academic papers on the relation between BPMN process models and corresponding data elements (called *artifacts* in BPMN terminology). We note that a *meta-model* description of BPMN (described in ORM) can be found in [14,16]; in [14] it is also discussed, be it not in terms of a systematic mapping, how activities in BPMN could relate to fact types in an information model.

In this paper, we will offer general principles of how to transform a basic BPMN process model to an ORM data model. In section 1, we will describe the elements of a basic BPMN process model, and offer an example of such a BPMN model. Section 2 shows how to derive, step by step, an ORM data model fragment from each element in a BPMN model. Eventually this procedure will result in a tailor-made data model supporting the complete BPMN process model. Section 3 describes a general mapping from a BPMN process model to an ORM data model. Section 4 offers conclusions. This paper uses basic ORM notations, cf. [10,13].

Section 1 Basic BPMN

This section describes some basics of the BPMN notation. We note that in our description of BPMN, we are not striving for completeness. On the contrary, *we wish to focus only on the very basic elements of BPMN notation, in order to explain the fundamentals of our approach to mapping Business Process Diagrams (BPD's) to ORM data models*. We refer the reader interested in more details of BPMN models to [5]. BPMN process models are composed of two basic categories of elements. The first category concerns activity nodes, denoting business events or items of work performed by humans or by software applications. The second category concerns control nodes capturing the flow of control between activities. Activity nodes and control nodes can be (almost arbitrarily) connected by means of a flow relation. Furthermore, activities within a process can be split into so-called pools and swimlanes. We will discuss some basic building blocks of BPD's below.

Events and Activities

An *Event* is represented by a circle and is something that "happens" during the course of a business process. Events affect the flow of the process and usually have a cause (trigger) or an impact (result). We will discern two types of events: the *start event* indicating the start of a flow, and the *end event* indicating the end of the flow. An *Activity* is represented by a rounded-corner rectangle and indicates a piece work or a task to be performed within the process. An Activity can be atomic or non- atomic

(compound). The types of Activities are: *Task* and *Sub-Process*. In this paper we will confine ourselves to simple tasks.

Control

A *Gateway* is represented by a diamond shape and is used to control the divergence and convergence of Sequence Flow. Thus, it will determine decisions, as well as the forking, merging, and joining of paths. Internal Markers indicate the type of behavior control.

Flow Relations

A *Sequence Flow* is represented by a solid line with a solid arrowhead and is used to show the order (the sequence) in which activities are performed in a Process.

An Example: The Thermostat

Our BPMN example concerns the modeling of a Thermostat: an example of a feedback-control based system used to steer a technical process. A thermostat is representative of many typical technical (but also business!) processes. For example, its behavioral characteristics lay the basis for airplane flight control (a Homeostat), or the basis for a quality management control system within an organization.

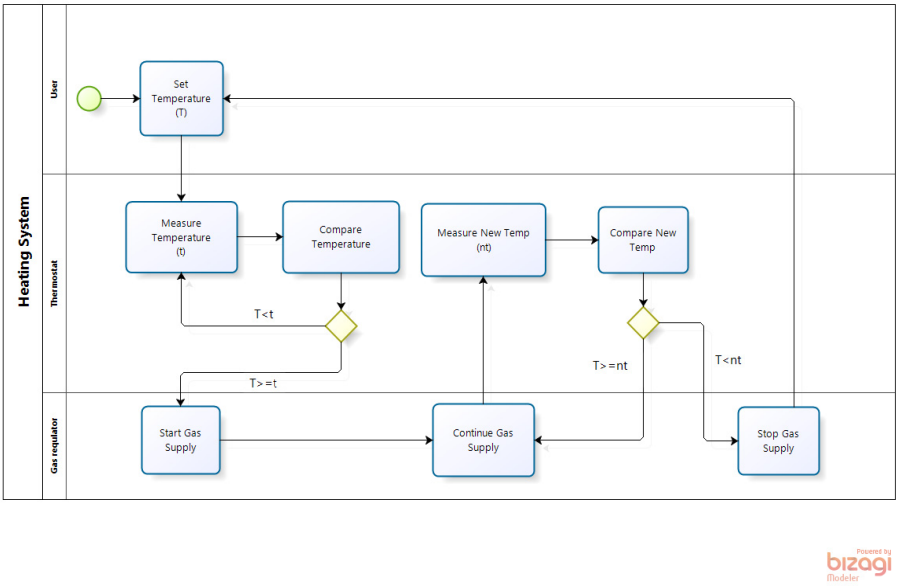


Fig. 1. Heating System with a Thermostat

Here we see a very simple (in some aspects over-simplified) Heating System described as a BPMN process model (designed using the BiZaGi BPMN process modeler tool [4]). The system distinguishes three swimlanes, associated to a user, the

thermostat, and the gas regulator, respectively. The system starts by having a user setting a temperature (T), followed by the thermostat measuring the environment temperature (t) and comparing t and T. Should it hold that T<t, then the thermostat measures (at some set time interval, say, e.g. each 30 seconds) the environment again, until the set temperature equals or exceeds that measured temperature. In the latter case, the gas regulator starts the gas supply, and will continue the gas supply as long a newly measured temperature (measured again at some set time interval, say) is indeed higher than the set temperature. In the latter case, the gas supply is stopped, and the system turns back to the start event waiting for a newly set temperature value. We note that this system has no end event: it is a typical example of an embedded system, constantly running and constantly returning to its start event.

Section 2 Transforming Basic BPMN Process Models into ORM Data Models

This section describes, using the example of our Thermostat in figure 1, a transformation of a basic BPMN process model into an ORM data model. Let's have a look at the first task in the process: an activity where a user sets some temperature. We could depict that task textually as follows:

BPMN-task1: <Set: Temperature>

where we employ a general format describing a BPMN task: <Verb-phrase present tense: Noun-phrase>

We could translate each occurrence (at a certain instant in time) of BPMN-task1 into an *event* as follows:

ORM-event1: [Temperature Setting: is logged]

where we employ a general format describing an event: [Noun-phrase Nominalized Verb-phrase: is logged]

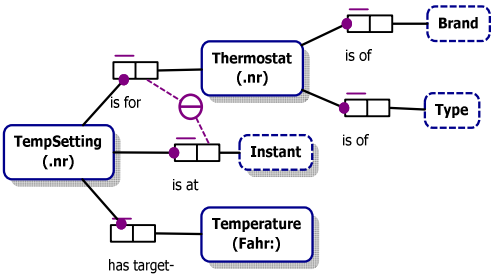


Fig. 2. Temperature setting (ORM-event1)

We will call the change of a verb into a corresponding noun phrase, the *nominalization* of that verb [13].We note that we use the word "event" as it is used in database terminology; i.e., in a CRUD sense. In the event [Temperature Setting: is logged], "is logged" refers to the time stamping of that event. A structure of an event is offered by a data model fragment capturing the occurrence of an activity at a certain instant in time. Using ORM, we could depict **ORM-event1** as follows:

We have arrived at this data model fragment (in this case pertaining to the event TemperatureSetting, written as TempSetting for short) by asking the following (not necessarily complete) set of general *event-identifying questions* against the task in question:

1. How do we *identify* the event?
2. At what *instant* (timestamp) does the event happen?
3. What do we have as *input* for the event?
4. What do we have as *output* of the event?

Of course, answering such a set of questions, will -in general- often need the support of domain expert knowledge. Using our framework of starting from a BPMN model, and posing such event-identifying questions, we can systematically arrive at a data model supporting the business process. Hence, the business process is seen as the *context* in which the business data is offered its place. In the case of our example, a temperature setting is identified by its own local number (a choice we have made), or by a particular thermostat together with an instant indicating the moment that the event occurs. Output of the event is a target temperature. This set of fact types as offered above, is *minimal* in the sense that it is tailor-made to offer exactly that data necessary to get the task of a temperature setting to run properly.

The second task that we investigate is

BPMN-task2 <Measure: Temperature>, resulting in the event

ORM-event2 [TempMeasurement: is logged]

The corresponding minimal ORM model is

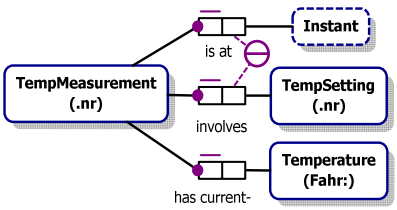


Fig. 3. Temperature measurement (ORM-event 2)

This model fragment was obtained by answering the same event-identifying question list as in the case of the previous TempSetting event. We note that we also need a rule (written here in OLE: ORM-Logic driven English [3]) indicating that a temperature measurement event is preceded (in time) by a temperature setting event

for each TempSetting, TempMeasurement, Instant1, **and** Instant2**:**
if that TempSetting is at **that** Instant1 **and** TempMeasurement is at **that** Instant2 **then that** Instant1<**that** Instant2.

Capturing dynamic aspects (such as TempSetting is followed by TempMeasurement) is taken care of by explicitly modeling time, and stating (static) rules about instants in time. This approach has been adopted from [2,12]. The subsequent BPMN-task is

BPMN-task3 <Compare: temperature>, and is followed by an exclusive-or fork, resulting in the event

ORM-event3: [TempComparison: is derived and logged]

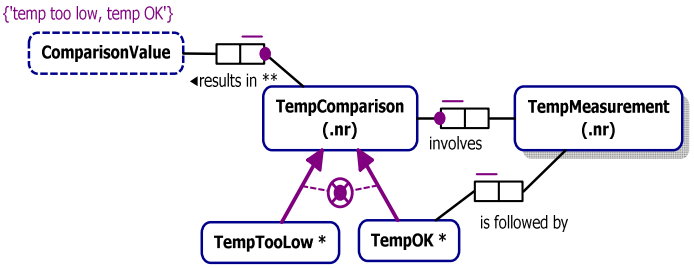


Fig. 4. Temperature comparison (ORM-event 3)

A temperature value will be called 'OK', exactly when the measured temperature is equal to or higher than the temperature value that has been set. Otherwise the temperature value is called 'too low'. In the case that the temperature value is 'OK', the system moves on to the next temperature measurement.
The derivation rule (written in OLE [3]) is offered by

for each TempComparison**: the** comparisonValue of **that** TempComparison **equals** `OK'**, exactly when the** temperature **of the** tempMeasurement **of that** TempComparison **is equal to or higher than the** temperature **of** (**the** tempSetting **of the** tempMeasurement **of that** TempComparison)

Our process of mapping the remainder of the BPMN-model to an ORM model moves along the same path as applied in the steps up till now. In summary, a BPMN-ORM procedure can be formulated as follows:

1. Create a BPMN-task
2. Transform that task into a desired ORM-event
3. Find a minimal model that realizes the desired ORM-event
4. Generate the corresponding relational view of a database
5. Create the next BPMN-task
6. Transform that task into a subsequent ORM-event
7. Find the minimal extension to the previous ORM model that realizes that subsequent ORM-event
8. Generate the next step in the evolution of the underlying database
9. Etc.

At the end of applying this procedure, you will have created a tailor-made corporate data model associated to a given BPMN process model. Subsequently, this model can be used to generate a relational database using NORMA [7,8]: the corporate database tailor-made to support the business process model.

Section 3 General Rules for Transforming BPMN Models to ORM Models

In this section, we offer some general rules for transforming a simple BPMN process model to an associated ORM data model. Bellow, in figure 6, you see a simple BPMN model containing the basic constructions for building a process: a start event, followed by two activities in sequence, followed by a gateway (in this case an exclusive-or fork) resulting in a divergence of two activities, after which the process stops. This very general type of model in BPMN transforms into an ORM data model.

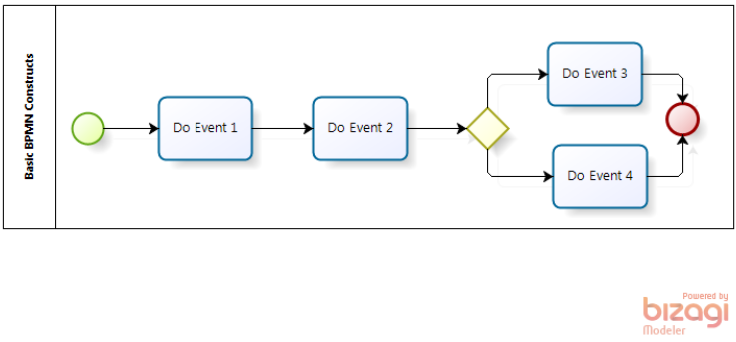
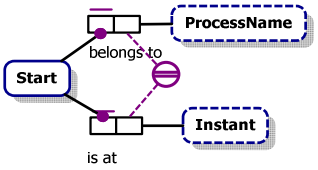


Fig. 5. Basic BPMN process model

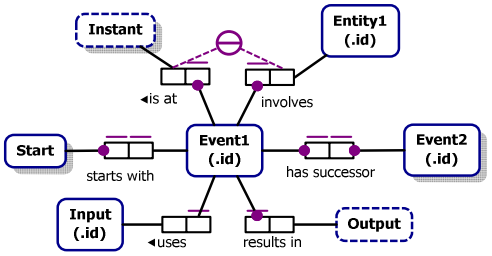
Activities will be denoted by employing a general format <Do EventName>, where an event here denotes the (time-stamped) occurrence of an activity in the CRUD sense. For example, the activity <Set: Temperature> (cf. the notation as introduced in

section 2), can be reformulated to <Do Temperature Setting> to fit such a format. We will now step by step transform figure 6 into an ORM data model

1. The Start event is transformed into



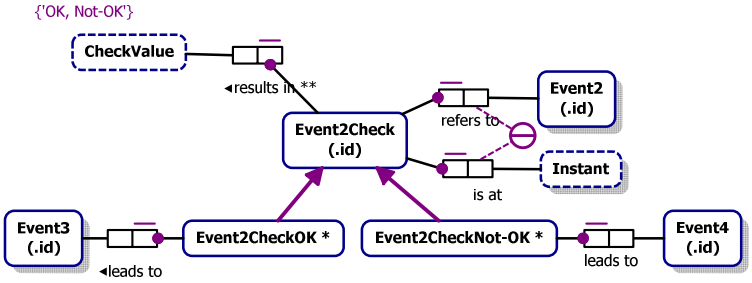
2. The sequence flow of activity <Do Event1> followed by <Do Event2> is transformed into



We note that we also need a rule to indicate that Event2 is preceded (in time) by Event1 (written here in OLE)

for each Event1, Event2, Instant1, **and** Instant2**:**
if that Event1 has successor **that** Event2 **and that** Event1 is at **that** Instant1 **and that** Event2 is at **that** Instant2
then that Instant1<**that** Instant2

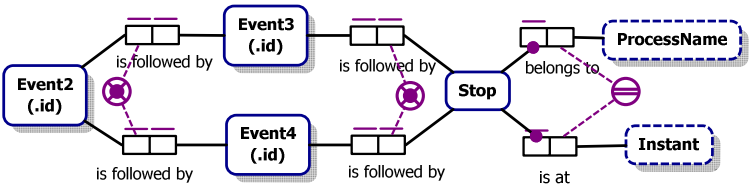
3. Activity <Do Event2> is followed by a gateway (in this case an exclusive-or fork) resulting in a divergence of two activities followed by an exclusive gateway, which is transformed into



where the two subtypes are defined (in OLE) by

- **an** Event2CheckOK **is an** Event2Check**, exactly when the** checkValue **of that** Event2Check **is** "OK"
- **an** Event2CheckNot-OK **is an** Event2Check**, exactly when the** checkValue **of that** Event2Check **is** "Not-OK"

4. The exclusive-or gateway results in a divergence of two activities <Do Event3> and <Do Event4>, after which the process stops; this results in the following transformation into an ORM model



We also need an extra constraint stating that if Event2 is followed by Event3, then Event3 is followed by some Stop event (and an analogous rule in the case that Event2 is followed by Event4). We can state this (in OLE) as follows

for each Event2 **and** Event3**:**
if that Event2 is followed by **that** Event3 **then that** Event3 is followed by **some** Stop

We note that an inclusive-or gateway in the process model, replacing the exclusive-or gateway, would result in replacing the exclusive-or constraint in the ORM model by a corresponding inclusive-or constraint. Parallel gateways can be treated analogously.

Section 4 Conclusions

This paper is aimed at tackling some of the problems encountered in deriving business data from process models. We have shown how to systematically map basic business process models using Business Process Modeling Notation (BPMN) to data models specified in ORM. From the resulting ORM model, we can generate a complete (corporate) relational database, containing the business data that is tailor-made to support the business process.

References

1. Bakema, G., Zwart, J., van der Lek, H.: Fully Communication Oriented Information Modelling. Ten Hagen Stam (2000)

2. Balsters, H., Halpin, T.: Formal Semantics of Dynamic Rules in ORM. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2008. LNCS, vol. 5333, pp. 699–708. Springer, Heidelberg (2008)
3. Balsters, H.: ORM Logic-Based English (OLE) and the ORM ReDesigner Tool: Fact-Based Reengineering and Migration of Relational Databases. In: Herrero, P., Panetto, H., Meersman, R., Dillon, T. (eds.) OTM-WS 2012. LNCS, vol. 7567, pp. 358–367. Springer, Heidelberg (2012)
4. BiZaGi Process Modeler, http://www.bizagi.com/
5. BPMN, http://www.omg.org/spec/BPMN/2.0/
6. Chen, P.P.: The entity-relationship model—towards a unified view of data. ACM Transactions on Database Systems 1(1), 9–36 (1976)
7. Curland, M., Halpin, T.: Model Driven Development with NORMA. In: Proc. 40th Int. Conf. on System Sciences (HICSS-40). IEEE Computer Society (January 2007)
8. Curland, M., Halpin, T.: The NORMA Software Tool for ORM 2. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 190–204. Springer, Heidelberg (2011)
9. FBM working group: Fact-based modeling exchange schema. Version 20111021c (2011), http://www.factbasedmodeling.org/
10. Halpin, T.: ORM 2. In: Meersman, R., Tari, Z. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 676–687. Springer, Heidelberg (2005)
11. Halpin, T.: ORM/NIAM Object-Role Modeling. In: Bernus, P., Mertins, K., Schmidt, G. (eds.) Handbook on Information Systems Architectures, 2nd edn., pp. 81–103. Springer, Heidelberg (2006)
12. Halpin, T.: Temporal Modeling and ORM. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2008. LNCS, vol. 5333, pp. 688–698. Springer, Heidelberg (2008)
13. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann, San Francisco (2008)
14. Morgan, T.: Business Process Modeling and ORM. In: Meersman, R., Tari, Z. (eds.) OTM-WS 2007, Part I. LNCS, vol. 4805, pp. 581–590. Springer, Heidelberg (2007)
15. OMG/UML: OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.3 (May 2010)
16. Russel, N., ter Hofstede, A.: Modern Business Process Automation, ch. 2. Springer (2010)