A Fact-Oriented Approach to Activity Modeling

H.A. (Erik) Proper, S.J.B.A. Hoppenbrouwers, and Th.P. van der Weide

Institute for Computing and Information Sciences, Radboud University Nijmegen,
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, EU
{E.Proper, S.Hoppenbrouwers, Th.P.vanderWeide}@cs.ru.nl

Abstract. In this paper we investigate the idea of using an ORM model as a starting point to derive an activity model, essentially providing an activity view on the original ORM model. When producing an ORM model of an inherently active domain, the resulting ORM model can provide an appropriate base to start out from. We will illustrate this basic idea by means of a running example. Much work remains to be done, but the results so-far look promissing.

1 Introduction

As argued before [7], ORM is not just suitable for the conceptual design of databases, but for the analysis of domains in general. Even if automation of some functionality is not a goal, ORM can be used to formalize and understand domains.

In this paper we propose to use ORM's fact oriented approach in the context of activity modeling. Addmittedly, the work is still at a preleminary stage. Much further work remains to be done. The basic idea is to provide a smooth transition from an ORM model of an *active* domain to an activity model of that domain. The approach we propose is to indeed start out from a 'plain' ORM model, and then to add 'temporal dependencies' between the fact types in the model. These temporal dependencies are preludes towards the actual triggering relationships between activity types and are therefore based on triggering mechanisms found in workflow modeling approaches. We use YAWL (Yet Another Workflow Language) [1] as a reference point. YAWL is a workflow language which aims to cover a range of workflow patterns that is as wide as possible [1].

Note that the resulting activity model really corresponds to a specific view of the underlying ORM model. Not all details present in an ORM model will appear in an activity model 'view' of the ORM model. Activity models are particularly suited to display the intended *flow* of activities, while an ORM model provides a much more generic perspective on the same domain. Most notably, the rich variety of constraints will not re-appear in the activity model 'view' of an ORM model.

Once the temporal dependencies have been determined, the fact types that are present in the ORM model of an (active) domain can be examined more closely. In this step the aim is to identify the *actors* that perform activities, the *activities* they perform and the *actands* which these activities are performed on.

R. Meersman et al. (Eds.): OTM Workshops 2005, LNCS 3762, pp. 666–675, 2005.

This leads to an ORM model with roles that are explicitly classified in terms of their *kind* of involvement in activities.

From this attributed ORM model, an activity model can be derived rather mechanically. In follow up research we are looking into ways of further mechanising the derivation of activity models from an attributed ORM model and also to maintain this link during the modeling process. We envisage modelers being able to manipulate the activity models while the underlying ORM domain model is kept up-to-date as well.

Note that the approach suggested does not particularly favor an event-driven, a process-driven or a data-driven modeling approach. ORM is used as a language to model domains in general, starting out from a fact-oriented perspective on the domain. In other words, the domain that is modeled (be it processes, data, or events) is viewed as being represented as a set of *facts*. These facts may deal with processes that occur in a domain or may deal with the kind of data/information that is being manipulated in the domain.

2 Temporal Ordering of Facts

As a running example, consider the following verbalizations (at the type level) of a domain dealing with patients visiting doctors:

A Person fills in a Form
A Doctor produces a Diagnose
A Person is examined by a Doctor
A Doctor writes a Prescription

This leads to the situation as depicted in Figure 1. Suppose now that in this domain we have the case that:

Before a Person can be examined by a Doctor, s/he should have filled in a Form.
Before a Doctor produces a Diagnose, a Person should have been Examined.
Before a Doctor writes a Prescription, a Person should have been Diagnosed.

These rules, however, still provide an incomplete picture. The examination, the diagnosing, and the writing of a prescription should, for a given doctor's visit,

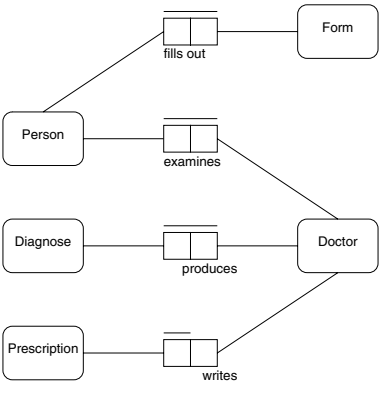

Fig. 1. Basic model of a visit to a Doctor

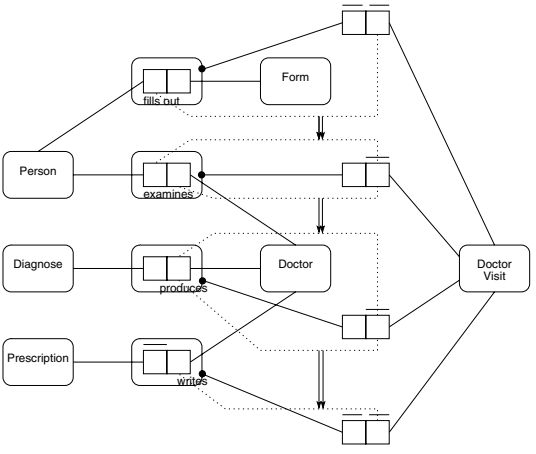

Fig. 2. Doctor visits

all be performed by the same doctor. Even more, as a person may visit a doctor twice for two different reasons, the diagnose and prescription really pertain to a specific doctor visit. This leads to the situation as depicted in Figure 2.

In defining the semantics of the depicted temporal dependencies a time axis is needed. When observing the universe of discourse at point in time t_1 we may observe: Person 'John' fills out Form 'A20.9012', at some later point in time t_2 John may have finished filling out the form. This means that at t_2 we cannot observe Person 'John' fills out Form 'A20.9012'. We could, for the sake of the example, imagine that at t_2 Doctor 'Smith' examines Person 'John' while this was not (yet) the case at t_1.

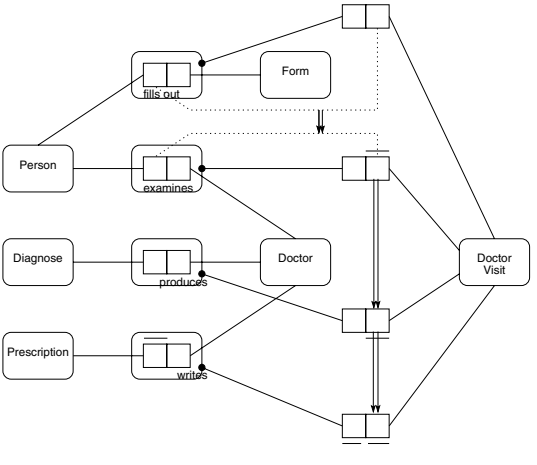

Fig. 3. Doctor visits with alternative semantics

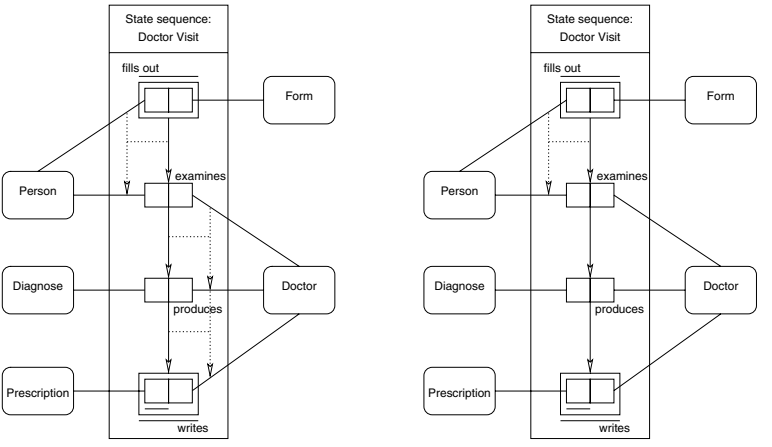

Fig. 4. Compact versions of Doctor visit

Informally, the semantics of the double arrow can now be defined as follows. All role combinations at the source of the arrow should cease to exist (in time) just before role combinations at the destination of the arrow come into existance. A proper formalisation of the semantics of such temporal dependencies is beyond the scope of this paper. It can, however, be defined in terms of the temporal semantics associated to ORM as provided in e.g. [8].

Note that work on *object life cycles* as reported in e.g. [2] focuses on the temporal depencies of roles (in facts) that are played by (the instances of) *one* object type only. However, to be able to arrive at activity models describing the concerted behavior of several objects within a domain, a notation as shown-in / suggested-by Figure 2 is needed.

Now consider the situation as depicted in Figure 3. This time, the doctor who examined the patient does not have to be the doctor producing the diagnose.

Since the diagrams of Figure 2 and 3 are rather complex, we introduce (in line with the notation proposed in [3]) the graphical notation as shown in Figure 4.

Based on the notation used in YAWL [1], the temporal dependencies may be combined to form complex synchronisation patterns. This is illustrated in Figure 5.

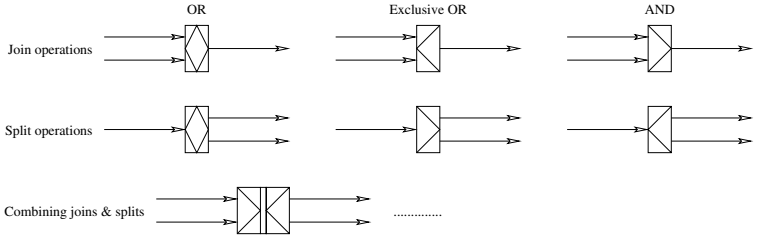

Fig. 5. Complex synchronisation

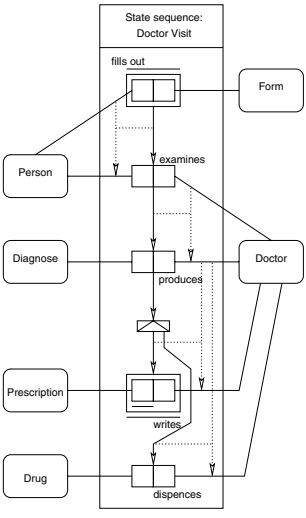

Fig. 6. Choice of medication

An example of the use of a complex synchronisation pattern is provided in Figure 6. In this model, a doctor can either dispense a drug directly, (exclusive) or provide a prescription.

3 Classification of Roles

In activity modeling we are concerned with modeling of domains that are *active*. So there must be *objects* in the domain playing an active *role*. These activities should be reported in the facts that can be verbalized when capturing the universe of discourse. The aim of this section is to take a closer look at these facts in order to find the *activities*, the *actors* who perform them and the *actands* they are performed on. *When* the activities occur is captured by the temporal dependencies as discussed in the previous section.

With these new modeling concepts, we can typically take ORM domain models and "annotate" them in terms of the refined concepts. We will base this process of annotation on linguistic foundations in line with the ORM tradition. Based on these annotated ORM models, we expect to be able to mechanically derive models in a notation that is better suited for the modeling of activities.

Let us now, as an example, consider the following domain:

> A person with name James is writing a letter to his loved one, at the desk in a romantically lit room, on a mid-summer's day, using a pencil, while the cat is watching.

with elementary facts:

> A person is writing a letter
> This person has the name James
> This letter has a romantic nature
> This letter has intended recipient James's loved one
> The writing of this letter by James, occurs on a mid-summer's day

The writing of this letter by James, is done using a pencil
The writing of this letter by James, is done while the cat is watching
The writing of this letter by James, is taking place at a desk
This desk is located in a room
This room is romantically lit

Within these elementary facts, several *players* can be identified. In the above example, we can isolate the players and facts as follows:

[A person] is writing [a letter]
[This person] has [the name James]
[This letter] has a [romantic nature]
[This letter] has intended recipient [James's loved one]
[The writing of this letter by James], occurs on a [mid-summer's day]
[The writing of this letter by James], is done using [a pencil]
[The writing of this letter by James], is done while [the cat] is watching
[The writing of this letter by James], is taking place at [a desk]
[This desk] is located in [a room]
[This room] is lit in [a romantic] way

The writing of the letter is the central fact in the above domain. All players in the facts describing the above domain are players in this domain. What are the activities, actors and actands? Several degrees of activeness exist with regards to the role which a player plays in a fact/domain. Numerous linguistics-oriented frameworks exist to classify the roles that objects play in sentences/facts (for example [5]). In our case we are primarily interested in distinguishing between actors and actands. With this aim in mind, we propose the following classification scheme for roles. The secondary classification level is mainly intended to better and further clarify the top level classification. On a scale of decreasing activity:

Actor role – A role where the player is regarded as carrying out an activity. Linguists may also use the term *agentive*.
In the example domain: The person.
Two sub-classes may be identified:
Initiating role – An agentive role, where the player is regarded as being the initiator of the activity.
Reactive role – A non-initiating agentive role.

Actand role – A role where the player is regarded as experiencing/undergoing an activity.
In the example domain: a letter, a loved one and the cat.
Three sub-classes may be identified:
Patient role – An experiencing role, where the player is regarded as undergoing changes (including its very creation) as intended by the actor.
Beneficiary role – An experiencing role, where the player is regarded as the beneficiary and/or recipient of the results of the activity.

Contextual role – A role where the player is regarded as being a part of the context in which the activity takes place. This role typically corresponds to the "adjuncts" in natural language.
Four sub-classes may be identified:
Instrumental role – A role where the player is regarded as being an instrument in an activity.
In the example domain: a desk and a pencil.
Spatial-locative role – A role, where the player is regarded as being the *physical* location of an activity.
In the example domain: the desk and the room.
Temporal-locative role – A role, where the player is regarded as being a temporal orientation of the activity.
In the example domain: mid-summer's day
Catalysing role – A role, where the presence of the player is regarded as being beneficial (either in a positive or a negative way) to an activity.
In the example domain: the room lit in a romantic way.
Observative role – An experiencing role, where the player is regarded as observing/witnessing the activity
In the example domain: the cat.

Predicative role – A role where the player is regarded as being a predicate on some other player.
In the example domain: the name James.

Each role in an elementary fact must fit within one of these classes. The choice between the different classes is subjective. It depends on the viewer. An elementary fact is an *activity* if-and-only-if it contains at least one actand or actor role, otherwise it is a *predication*. The objects playing the four main classes of roles are regarded as the *actors*, *actands*, *context elements*, and *predicators* respectively.

For the example given above, we would have:

Activity: [Actor: A person] is writing [Actand: a letter]
Predication: [This person] has [the name James]
Predication: [This letter] has a [romantic nature]
Predication: [This letter] has intended recipient [James's loved one]
Predication: [The writing of this letter by James], occurs on a [mid-summer's day]
Predication: [The writing of this letter by James], is done using [a pencil]
Predication: [The writing of this letter by James], is done while [the cat] is watching
Predication: [The writing of this letter by James], is taking place at [a desk]
Predication: [This desk] is located in [a room]
Predication: [This room] is lit in [a romantic] way

In the example domain, the writing of the letter by the person is regarded as the key activity in the domain. In other words, writing is an activity, while the person is the actand and the letter is the actand. We may regard the cat and the loved one as an actand as well. What about the pen, the name James, the desk, etc? They are really players in *predications* over the other players. The fact that a pen is used by the person to write the letter is a *predication* of the writing *activity*.

If we were to zoom in on a sub-domain of the above sketched domain, we could actually find that what is an actand in the super-domain is an actor in the sub-domain. Consider, for example, the sub-domain:

[The writing of this letter by James], is done while [the cat] is watching

When considered in isolation, one may quite easily argue that the primary activity here is the watching, which is something that is being done by the cat. This really makes the cat into an actor rather than an actand, while the thing that is being watched (the writing) becomes the actand. This means that our notions of actor, actand, activity and predication are really to be taken *relative* to the domain under consideration.

As mentioned before, the work reported is still in its preleminary stages. One of the work that remains to be done is the provision of a sound theoretical base by which modelers can determine wether an object is a actor, actand or activity. In doing so, we plan on integrating the work on DEMO [9], which has a sound theoretical base in speech-act theory [4].

When taking our example domain of doctor visits we have:

Activity: [Actor: Person] fills out [Actand: Form]
Activity: [Actor: Doctor] examines [Actor: Person]
Activity: [Actor: Doctor] produces [Actand: Diagnose]
Activity: [Actor: Doctor] writes [Actand: Prescription]

Note that we have chosen to regard the person, who is being examined, as an actor as well. It is an example of a *collaborative* activity. Graphically, we now obtain the situation as depicted in Figure 7. Note that the classification is associated to the *roles*. Even though the example does not show it, an object

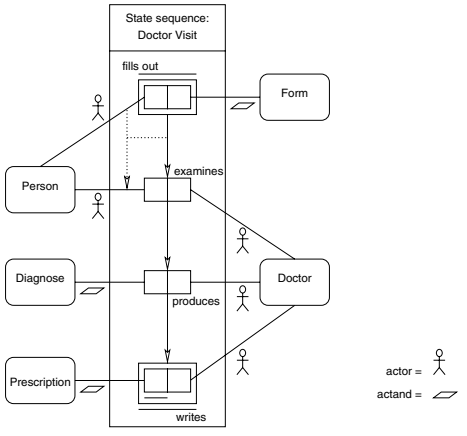

Fig. 7. Attributed ORM model of a Doctor visit

may (for obvious reasons) be an actor in one activity, while it is an actand in another one.

Finally, Figure 8 depicts the two variations of the doctor visit example domain as an activity model using the ArchiMate [6] notation. The notation used in this diagram is (arguably) better suited to represent activity models than the

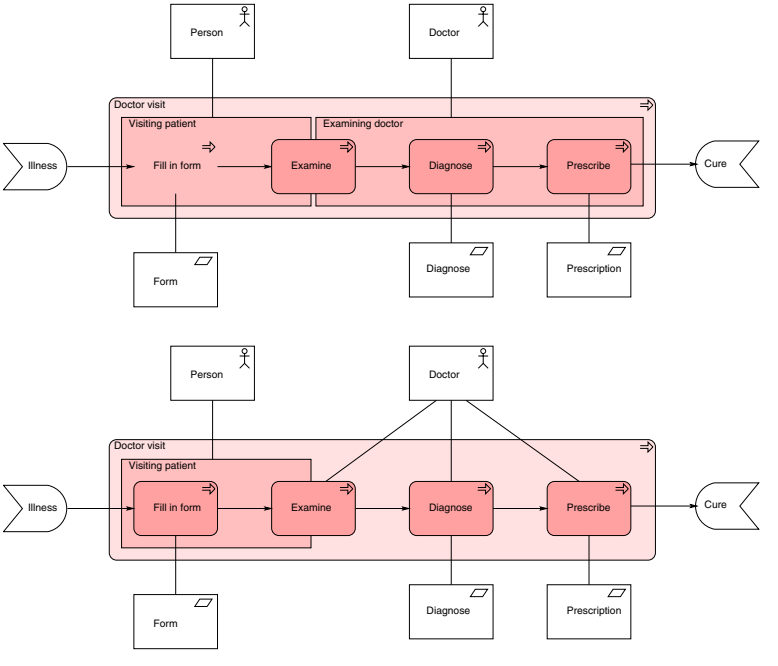

Fig. 8. Doctor visit as an activity model

ORM notation. However, we have now established a clear (and formalizable) relationship between activity models as presented in Figure 8 and ORM models as shown in Figure 4. Note the one-on-one correspondence between the actors, actands and activities.

4 Conclusions

In this short paper we have presented the idea of using ORM's fact oriented approach in the context of activity modeling. The basic idea as presented, is to provide a smooth transition from an ORM model of an *active* domain to an activity model of that domain. We did so by adding temporal dependency constraints to the ORM model, and attributing the model with a classification of roles. The resulting model was mapped onto a graphical notation used for activity modeling in a mechanical manner.

As mentioned before, the resulting activity model really corresponds to a specific view of the underlying ORM model. Not all details present in an ORM model will appear in an activity model 'view' of the ORM model. Activity models are particularly suited to display the intended *flow* of activities, while an ORM model provides a much more generic perspective on the same domain.

In future research, we will be looking at ways to provide modelers with more guidelines on classifying the roles. Furthermore, the relationship between the workflow patterns from YAWL [1] and the temporal dependencies that may be used in ORM models needs further investigation. We also intend to look at strategies to further mechanise the derivation of activity models from an attributed ORM model and also to maintain this link during the modeling process. We envisage modelers being able to manipulate the activity models while the underlying ORM domain model is maintained as well.

Finally, using ORM as a generalized domain modeling approach, and then 'specializing' this model towards an activity model (as illustrated in this paper), is not an idea that is limited to activity modeling alone. The general underlying idea of using a fact-oriented approach to produce an ORM model for a given domain, and then to specialize this model by re-interpreting the fact types in terms of a specialized classification, is an approach that is expected to be suitable for a wide range of specialized modeling languages including business modeling and architecture modeling. We will therefore also investigate these workings in more detail.

References

1. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245–275, 2005.
2. P. van Bommel, P.J.M. Frederiks, and Th.P. van der Weide. Object-Oriented Modeling based on Logbooks. *The Computer Journal*, 39(9):793–799, 1996.
3. P.N. Creasy and H.A. (Erik) Proper. A Generic Model for 3-Dimensional Conceptual Modelling. *Data & Knowledge Engineering*, 20(2):119–162, 1996.

4. J. Habermas. *The Theory for Communicative Action: Reason and Rationalization of Society*, volume 1. Boston Beacon Press, Boston, Massachusetts, 1984.
5. J.J.A.C. Hoppenbrouwers, B. van der Vos, and S.J.B.A. Hoppenbrouwers. Nl structures and conceptual modelling: Grammalizing for KISS. *Data & Knowledge Engineering*, 23(1):79–92, 1997.
6. M.M. Lankhorst and others. *Enterprise Architecture at Work : Modelling, Communication and Analysis*. Springer, Berlin, Germany, EU, 2005.
7. H.A. (Erik) Proper, A.I. Bleeker, and S.J.B.A. Hoppenbrouwers. Object-role modelling as a domain modelling approach. In J. Grundspenkis and M. Kirikova, editors, *Proceedings of the Workshop on Evaluating Modeling Methods for Systems Analysis and Design (EMMSAD'04), held in conjunctiun with the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004)*, volume 3, pages 317–328, Riga, Latvia, EU, June 2004. Faculty of Computer Science and Information Technology, Riga Technical University, Riga, Latvia, EU.
8. H.A. (Erik) Proper and Th.P. van der Weide. EVORM - A Conceptual Modelling Technique for Evolving Application Domains. *Data & Knowledge Engineering*, 12:313–359, 1994.
9. V.E. van Reijswoud, J.B.F Mulder, and J.L.G. Dietz. Commucation Action Based Business Process and Information Modelling with DEMO. *The Information Systems Journal*, 9(2):117–138, 1999.