On the use of Object-Role Modelling to Model Active Domains

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

Institute for Computing and Information Sciences,
Radboud University Nijmegen,
Toernooiveld 1, 6525 ED Nijmegen, the Netherlands, EU

{P.vanBommel, S.Hoppenbrouwers, E.Proper, Th.P.vanderWeide}@cs.ru.nl

1. Introduction and Background

Conceptual modelling methods such as ER (Chen, 1976), EER (Elmasri and Navathe, 1994; Gogolla, 1994), KISS (Kristen, 1994), NIAM (Nijssen and Halpin, 1989), OOSA (Embley, Kurtz, and Woodfield, 1992) and ORM (Halpin, 2001) have traditionally been developed with the aim of providing conceptual models of database structures. More recently, however, such modelling methods have shown their use for more generic modelling (of the ontology) of domains, leading to models capturing the concepts of a domain in general, as well as an associated language to express rules (such as business rules) governing the behaviour of the domain (Spyns, Meersman, and Jarrar, 2002; Proper, Bleeker, and Hoppenbrouwers, 2004; Spyns, 2005; Proper, Hoppenbrouwers, and van der Weide, 2005).

The above mentioned modelling methods typically take a natural language based perspective to the domain to be modelled. In this perspective, the resulting model is regarded as *domain grammar* describing the allowed communication about a domain (the *universe of discourse*). This way of thinking dates back to the ISO report on *Concepts and Terminology for the Conceptual Schema and the Information Base* (ISO, 1987), and is at the base of the modelling methods mentioned. A key advantage of such methods is that having a domain grammar at one's disposal enables validation of the model by domain experts, since the model can be validated in terms of statements that are close to the language used by these experts. Also, formal approaches to conceptual modelling imply adherence to a formal language; domain grammars (closely resembling signatures in formal logic) are an excellent basis for further formal modelling.

A basic domain grammar can be extended to cover rules (constraints) governing the structure and behaviour of/in the domain. When combined with a reasoning mechanism, such a rule language becomes a *domain calculus*. In the case of ORM, a domain calculus has been presented in the form of Lisa-D (ter Hofstede, Proper, and van der Weide, 1993), a formalisation of RIDL (Meersman, 1982). In (Proper, 1994b) and (Bloesch and Halpin, 1996), a more practical version was introduced (that is, from an implementation point of view), called ConQuer. What Lisa-D and ConQuer have in common is that they exploit the natural character of the domain grammar in the construction of rules (ter

Hofstede, Proper, and van der Weide, 1997). As a result, the formulation of rules, as well as chains of reasoning expressed in these rules, closely resemble natural language. As mentioned before, being able to do so supports validation of the models produced.

In the use of domain modelling methods, we observe *three important trends* which fuel our ongoing research activities. **Firstly**, more and more organisations strive for more mature levels of system development (Paulk, Curtis, Chrissis, and Weber, 1993). One of the steps towards maturity involves better *defining* of development processes in order to make them more *repeatable*. This also applies to modelling processes. Some organisations now indeed strive to make modelling processes more explicitly defined with the aim of achieving more repeatable results.

Modelling methods such as ORM (Halpin, 2001), NIAM (Nijssen and Halpin, 1989), OOSA (Embley et al., 1992), KISS (Kristen, 1994) and DEMO (van Reijswoud and Dietz, 1999) not only feature a *way of modelling*, but also have a fairly well-defined and explicit *way of working* based on natural language analysis. The *way of working* (Wijers and Heijes, 1990) of a method is concerned with processes, guidelines, heuristics, etc., which are to be used in the creation of models, as opposed to its *way of modelling* which refers to the syntax and semantics of the language in which the models are to be expressed. A well-defined and explicit way of working helps achieve a defined and more repeatable modelling process. Even though the ORM Conceptual Schema Design Procedure already provides clear guidelines for creating domain models in a repeatable fashion, more work in terms of sound theoretical underpinning and automated support of the (detailed steps of the) modelling process is still called for (Hoppenbrouwers, Proper, and van der Weide, 2005a), and is one of the main goals underlying our ongoing research. This is not our main focus here, but it is a partial explanation for our preference for ORM.

The **second** trend fuelling our research is the use of *controlled languages* as the basis for unambiguous communication concerning models and specifications (Farrington, 1996; Fuchs and Schwitter, 1996; AECMA, 2001; Hoppenbrouwers, 2003; Schwitter, 2004). The essential idea behind a controlled language is to define a subset of natural language which is rich enough to have a natural, intuitive feel to it, but still restrictive enough so as to avoid ambiguities. The use of a controlled language requires a realistic and nuanced approach. Too often it is simply assumed that any participant in the modelling process will be able to express herself freely and without problems in some controlled language. This is hardly ever a tenable assumption. A possible way of working around this is to use the controlled language strictly receptively (not productively) i.e. have a capable intermediary (a person, an automated system, or a combination thereof) rephrase in the controlled language the natural language statements made by the other participant(s) in the modelling process, and then have those participants confirm the validity of the controlled language statements (Hoppenbrouwers et al., 2005a).

We claim that in view of the goals of formal/unambiguous expression, a *domain grammar* and associated *domain calculus* provide a good starting point for engineering controlled languages for use in domain modelling. To some extent, a *domain calculus* already provides a controlled language, though not necessarily a “natural” one. Stepwise naturalization (using a number of levels of increasingly natural representations) is the approach we follow here; admittedly, we have not achieved fully acceptable naturalness

as of yet, but when we do it will still be based on fully formal underlying structures, thus linking focused and controlled natural representation of concepts with well-constructed formalism. Calculus-like controlled languages can be used to represent domain specific reasoning steps, providing an additional form of domain knowledge. In (Hoppenbrouwers, Proper, and van der Weide, 2005b) an initial study into the use of a domain calculus for such purposes has been reported.

The **third** trend we observe is the growing need for integrated models underlying a plethora of *viewpoints*, fuelled by the demands of MDA (OMG, 2003; Frankel, 2003) and enterprise architecture (Lankhorst, 2005). UML (OMG, 2003) as well as approaches for enterprise architecting feature a wide variety of viewpoint-specific diagramming techniques (viewpoints). A generic domain model can provide a *common underpinning* of these viewpoints, offering a *unified domain ontology*. An elaboration of this role of domain models has been presented in (Proper et al., 2005; Proper and van der Weide, 2005). However, as we adopt ORM as a generic technique for creating a unified domain ontology, work still needs to be done to bring together concepts for activity modelling and ORM. *This effort is the main focus of this chapter.* But let us elaborate some more on the reason why we use ORM for our purposes.

Importantly, it is not our intention to propose our ORM extension as a new technique for modelling active domains that is to be on par with, for example, UML Activity Diagrams. We applaud the use of such diagrams as popular *viewpoints*. Integration at a deeper level is our primary drive here, mostly for academic and educational purposes. Application in industry should eventually be possible, but is not one of our objectives at this point. For clarity's sake, the arguments for using ORM in our current investigation are summarised here:

- There is a growing need for integrated models underlying a plethora of *viewpoints*. Because of its generic conceptual nature, ORM seems a good candidate for use as a generic technique/method for integrating the many more specialised modelling techniques around.
- ORM provides a good existing basis for investigating, defining, and improving ways of working, in combination with formal rigour and soundness with respect to the way of modelling. Getting a better grip on the way of working serves the higher goal of making modelling processes more mature.
- As part of the way of working, a strongly natural language-oriented approach to elicitation and validation of models is part and parcel of the ORM/ORC approach (ter Hofstede et al., 1997; Hoppenbrouwers et al. 2005a). The Logbook principle and the use of controlled language are elements of this approach.

In this chapter, then, we show how the ORM/ORC approach (with the advantages it does, in our view, possess) could be applied to temporal modelling, *integrating it in the existing approach*. For this purpose, we need the following ingredients:

- A basic yet coherent set of temporal concepts to work with; this could perhaps have been another set. Please note that fundamental matters concerning the ideal set of temporal modelling concepts are not within the focus of this paper. For an elaboration on such issues, see for example (Allen and March, 2003; March and

Allen, 2003; Khatri, Ram, and Snodgrass, 2004). We make do with a small set of concepts fit for our current purposes, based on concepts from established workflow-related work (van der Aalst and ter Hofstede, 2005).

- Given our goal to apply and maintain ORM-style formal rigour to our way of modelling, we show that our ORM extension fits our existing ORM/ORC formalization. In order to do this, we need to explain some key aspects of that formalization. This does require both active aspects (activities, tasks, processes, etc.) and static aspects (results, documents, actors, tangible objects, etc.) to be expressed as *objects* playing *roles* in the domain.
- Given the elicitation/validation goals underlying the ORM/ORC approach, we include both a verbal and a graphical style of representation. While both are demonstrated, they are still somewhat experimental. They serve to illustrate the integration aspect of our exercise rather than as a proposition for new forms of representing active domain models in an industrial context.

The body of this chapter is structured as follows:

1. (Section 2) using the *logbook paradigm*. The activities taking place in an active domain can be reported in terms of (elementary) facts, which can consequently be used (in principle using ORM's standard approach) to derive a *domain grammar*;
2. (Section 3) any constraints, temporal dependencies, etc., governing the flow of activities in a domain can then be formulated using a *domain calculus* referred to as the *Object-Role Calculus*;
3. (Section 4) finally, special graphical conventions are introduced to provide more compact representations of specific aspects of the active domain, such as the *flow* of activities, or the *involvement* of actors.

2. The logbook paradigm

When focussing on active domains ORM needs to be refined in order to better cater to the active aspects of such domains. The underlying challenge is to extend ORM to be able to deal with such domains, while at the same time maintaining ORM's natural-language based modelling rigour. In doing so, we base ourselves on earlier (partial) results (Proper and van der Weide, 1994; Proper, 1994a; Frederiks, 1997; Frederiks and van der Weide, 2002).

Modelling an active domain does require a modelling language to deal with the notion of time. In the past, ORM has already been extended with the concept of time and evolution (Proper and van der Weide, 1994; Proper, 1994a). In this chapter we propose a formalization of temporal concepts in terms of a *logbook* (Frederiks, 1997; Frederiks and van der Weide, 2002), which is intended to trace/mirror the activities taking place in the domain. Such a logbook will consist of a series of events reporting on the lifecycle of facts in the domain. For example:

Trafic light 20 is green
`ceased being true at 11:03:20 on 22-05-2006`
Employee John works on the completion of order 50
`started being true at 09:30 on 19-05-2006`

In our view, a logbook approach is a natural extension of the earlier discussed natural language based perspective on modelling. To be more precise, we regard a *history* as an overview of the events that have taken place in the domain, while a *logbook* is a description of such a history. The facts contained in the descriptions of the events are assumed to be expressed in terms of some semi-natural language (controlled language) sentences as is normally the case in ORM's way of working. Using a traditional ORM approach, the set of facts used/allowed in a logbook can be generalised to a set of fact types, which together comprise the ORM model underlying the domain. As such, this ORM model then defines the domain grammar of the controlled language in which the facts are to be formulated.

Traditionally, ORM modelling is based on the modelling of facts in general. In the context of an active domain, these facts correspond to statements about what is the case and/or has happened in the domain at specific points in time. In ORM, the actual modelling process starts out from the verbalisation of such facts. These verbalisations are the starting point for the creation of the domain grammar. When considering an active domain, the set of facts that can be reported about this domain fall into two categories: (1) *acts* reporting on the performance of *actions* and (2) *effects* reporting the *results* of actions[1]. This dichotomy applies at the instances level (the facts) as well as the type level (the *fact types*), leading to *act types* and *effect types* respectively as sub-classes of fact types. In the case of *acts*, the objects involved (i.e. playing a role in the act) can be classified further into *actors* (objects responsible for performing the *act*) and *actands* (objects which are the effect of the *act*).

We assume that each event described in the logbook and the objects participating in the event can be uniquely identified in that logbook. We will call this the *Event Identification Principle*. This identification principle is used as the base for all other identification mechanisms. This principle does not inhibit different events to occur at the same time. In order to distinguish between accidentally and necessarily coupled events, we assume that events may also have a compound nature, in such a way that: (1) different events in a logbook are independent of each other, (2) events are cannot be split into multiple independent events.

We take the perspective that the state of an active domain is the result of the sequence of actions leading up to that state. These actions may either take place in the domain, or outside the domain (the latter possibility includes the very creation of the domain). As a result, we take the position that the *effects* can in principle be derivable from the set of reported *acts*. This is what we call the *Action Dominance Principle*. This principle leads

[1] Note: these two classes correspond to what Dietz (2005) respectively refers to as *acta* and *facta*.

to the theoretical question of how persistent properties, such as the speed of light, are to be treated in our logbook approach. This is covered by the *Property Origination Principle*, which states that each domain property pertains to: (1) either some act taken place in the domain, (2) or some effect of some act in the domain, (3) or some effect of the domain's creation (i.e. the result of a 'big bang' act). As a consequence, at each moment the state of the system is (in principle) the result of all the effects of the domain's creation and the acts that were reported since then.

An important consequence of the *Property Origination Principle* is that (for most objects in the domain), the property of *being alive* should be the result of some act (this notion is similar to that of "existence time" as discussed in (Khatri et al., 2004), among others). Therefore, objects that are not present in the initial state in principle require an explicit birth event. This is called the *Birth Principle*. Obviously, an object can not be responsible for its own birth, as it can not be active before coming into existence. The consequence is that some other object has to be responsible for causing this event, thus playing a dominant role in that event. If the existence of an object may terminate, then there should in principle be an explicit death action that enforces an object to have the property of being dead.

An immediate consequence of the Birth Principle and the Event Identification Principle is that objects may be identified by their birth event. If an event starts life for more objects, then we require that the individual objects in this case may be identified by this event and in addition the role they play in this event.

Note that the above principles could only hold absolutely under a closed world assumption, which is most practical cases is naïve. We therefore emphasise that more traditional means of object identification are not excluded from our approach. The principles as presented merely reflect our perspective on active domains. For example, if the birth event of an object is unknown, or even if it *is* known, identification of the object as such, by means of a simple key or label, would still be quite acceptable. Analytically, however, questions may be raised as to the origin of the object. Whether such questions are acutely relevant depends on the modelling context.

3. Object-Role Calculus

This section concerns a conceptual language in which rules can be expressed for describing the behaviour that may be observed in a logbook compatible with the domain being modelled. The language presented, referred to as *Object-Role Calculus* (ORC) is a variant of Lisa-D (ter Hofstede et al., 1993), a formalisation of RIDL (Meersman, 1982). Lisa-D was originally designed to describe all computable sets of facts that can be derived from the elementary facts defined in the underlying conceptual schema. The conceptual schema specifies all elementary sentences applicable in that domain. The semantics of Lisa-D have been described in terms of multi-sets. In this chapter we will provide a light-weight definition of the ORC variant of Lisa-D, which is intended to describe temporal and static aspects of the underlying domain.

3.1 Grounding in temporal logic

The semantics of ORC are grounded on Kripke structures (Chellas, 1980). In terms of Kripke structures, an application domain is seen as a Kripke structure $\langle S.R.s_0,\prod,L\rangle$, where:

1. S is a non-empty set of states,
2. $R \subseteq S \times S$ is a total transition function, i.e. $\forall_s \exists_t [(s,t)\ R]$,
3. s_0 is the initial state,
4. $\prod$ is a non-empty set of atomic propositions, and
5. L is a labelling function that maps each state on a subset of $\prod$.

Our main assumption is that the state of an application domain is in principle described by its history so far. As a consequence, a state corresponds uniquely to a logbook. Consequently, the transition function extends a logbook with a new event description, and the initial state is obtained as the empty logbook.

From the structure of the events in the logbook, the elementary object types can be derived. Their possible instantiations form the set of $\prod$ atomic propositions. The labelling function L then assigns the population of object types that is constructed by a logbook.

A linear-time temporal logic is syntactically described by the following BNF grammar (see for example (Lipeck and Saake, 1987)):

$$\phi \rightarrow true \mid false \mid \prod \mid \neg\phi \mid q \wedge \phi \mid \phi \vee \phi \mid \phi \Rightarrow \phi \mid \mathsf{X}\phi \mid \mathsf{F}\phi \mid \mathsf{G}\phi \mid \phi\,\mathsf{U}\,\psi$$

The expression $\mathsf{X}\phi$ states that ϕ will hold in the next state, $\mathsf{F}\phi$ that ϕ will eventually hold, $\mathsf{G}\phi$ that ϕ will globally hold and $\phi\,\mathsf{U}\,\psi$ states that at some point ψ will hold, while in all states before, ϕ is valid. Let M be a Kripke structure over logbook LB, and let σ be a history. We will further assume an environment E for evaluation, consisting of a partial assignment of values to a set V of variables. The standard semantic interpretation of the temporal operators is:

$$M, E, \sigma \models \mathsf{X}\phi \quad \equiv^{2} \quad M, E, \sigma^{1} \models \phi$$

$$M, E, \sigma \models \phi\,\mathsf{U}\,\psi \quad \equiv \quad \exists_n [\ \forall_{0<i<n} [\ M, E, \sigma^{i} \models \phi\ ] \wedge M, E, \sigma^{n} \models \psi\ ]$$

where σ^{i} denotes the i-th element of sequence σ and σ^{i} the subsequence of σ starting at position i. The other temporal operators are defined in terms of these base operators: $\mathsf{F}\phi$ is equivalent with $true\ \mathsf{U}\phi$, and $\mathsf{G}\phi$ is defined as $\neg\ \mathsf{F}\ \neg\ \phi$. The propositional operators are also interpreted in the standard way:

---

[2] For lack of a typographic alternative we use this symbol here for "is defined as"

$$M, E, \sigma \models \neg \phi \quad \equiv \quad \neg\, M, E, \sigma \models \phi$$
$$M, E, \sigma \models q \wedge \psi \quad \equiv \quad M, E, \sigma \models \phi \wedge M, E, \sigma \models \psi$$

The constant *false* is introduced as $p \wedge \neg p$ where p is any proposition from $\prod$, and *true* is derived by $\neg$*false*. The other logical operators ($\vee$ and $\Rightarrow$) are defined in the usual way. The conversion from a temporal proposition to a static expression requires the evaluation of the static expression for the population $L(\sigma(0))$ at the required point of time. This will be further elaborated in section 3.4.

3.2 Historical Information descriptors

History descriptors in ORC are meant to provide a language construct for reasoning in an historical setting about the application domain. For the purpose of this chapter, it will be sufficient to make more or less direct transcriptions of the basic temporal operators. For this the syntactical construct *history descriptor* is introduced. Let H be a history descriptor, then the semantics of H are denoted as [(H)]:

$$[(\text{always } H)] \quad \equiv \quad \mathsf{G}\,[(H)]$$
$$[(\mathsf{X}\, H)] \quad \equiv \quad \mathsf{X}\,[(H)]$$

In addition we introduce the following abbreviations:

$$\text{sometimes } H \quad \equiv \quad \neg \text{ always } \neg\, H$$
$$H_1 \text{ precedes } H_2 \quad \equiv \quad \text{always}((\mathsf{F}H_1)\ \mathsf{U}\ H_1$$
$$H_1 \text{ during } H_2 \quad \equiv \quad \text{always}(H_1 \Rightarrow H_2)$$
$$H_1 \text{ triggers } H_2 \quad \equiv \quad \text{always}(H_1 \wedge \neg H_2 \Rightarrow \mathsf{X}\,(\neg H_1 \wedge \neg H_2))$$

The first rule will be a target for the educational organisation. The latter rule states describes a trigger that, whenever the condition $H_1 \wedge \neg H_2$ is met, will respond by setting the condition $\neg H_1 \wedge \neg H_2$ at the next moment. Some example expression would be:

sometimes Lecturer lectures Course

Lecturer sets up Course precedes Lecturer lectures Course

This latter expression, however, is misleading as it does not bring about a connection between some specific lecturer and some specific course being set up and being lectured. In natural language, indicatives are used in most cases to make such references. We therefore introduce:

x [[D_1 PRECEDES D_2]] y ≡ (x [[D_1]] y) precedes $\exists_z$ [z [[D_2]] y]

x [[D_1 DURING D_2]] y ≡ (x [[D_1]] y) during $\exists_z$ [z [[D_2]] y]

The semantic and syntax of these constructions are further explained in section 3.4. Please note that we use the repeated bracket "[[" notation here, for typographical lack of a properly fused double square bracket. All immediately adjoining brackets in this chapter are double square brackets, never two single square brackets.

3.3 Indicative descriptors

The main idea behind ORC, as present in its early ancestor RIDL (Meersman, 1982) is a functional, variable-less description of domain-specific properties (and queries). RIDL does contain a linguistic reference mechanism (the indicative THAT). In ORC, variables have been introduced to handle more subtle referential relations that cannot be handled by indicatives. Variables are special names that are instantiated once they are evaluated in a context that generates values for this variable. The concept of *environment* is used to administrate the value of variables. In environment E, the variable v will evaluate to $E(v)$. Some examples of the use of variables:

Lecturer:x being hired precedes x sets up Course

Lecturer:x sets up c precedes x lectures Course:c

In this example, the expression Lecturer:x is a defining occurrence of variable x in which Lecturer has the role of value generator. The environment is used to administrate the variable-value assignment (see (ter Hofstede et al., 1993) for more details).

3.4 Information descriptors

The syntactic category to retrieve a collection of facts is called *information descriptor*. We will discuss the semantics of elementary information descriptors, and briefly summarise the construction of information descriptor (a diagram is provided in Figure 1; for more details, see (ter Hofstede et al., 1993)). Information descriptors are constructed from the names of object types and role type. The base construction for sentences is juxtaposition. By simply concatenating information descriptors, new information descriptors are constructed.

Information descriptors are interpreted as binary relationship; they provide a binary relation between instances of the population induced from the history. The semantics of information descriptor D are denoted as [[D]]; we will write x [[D]] y to denote the relationship between x and y. The statement M, E, σ |= x [[D]] y asserts that for Kripke structure M in environment E from history σ the relationship x [[D]] y can be derived.

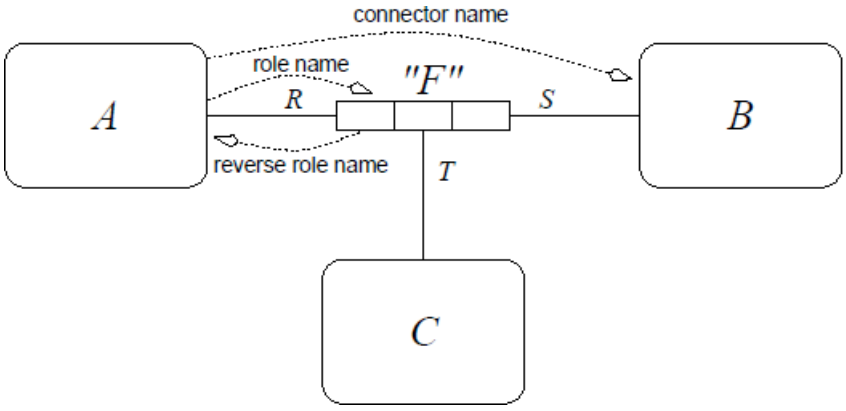

Figure 1: Role Names

A population assigns to each object type its set of instances. Let *n* be the name of object type *N* and *r* the name of a role type *R*, then *n* and *r* are information descriptors with semantics:

$$M, E, \sigma \models x\ [[n]]\ y \quad \equiv \quad x \in L(\sigma(N)) \wedge x = y$$
$$M, E, \sigma \models x\ [[r]]\ y \quad \equiv \quad (x, y) \in L(\sigma(R))$$

A single role may, in addition to its 'normal' name, also receive a *reverse role name*. Let *v* be the reverse role name of role R, then we have:

$$M, E, \sigma \models x\ [[v]]\ y \quad \equiv \quad (y, x) \in L(\sigma(R))$$

A combination of roles involved from a fact type may receive a *connector name*. The connector name allows us to 'traverse' a fact type from one of the participating object types to another one. If *c* is the connector name for a role pair $\langle R, S \rangle$, then the semantics of the information descriptor *c* are defined as:

$$M, E, \sigma \models x\ [[c]]\ z \quad \equiv \quad \exists_y\ [M, E, \sigma \models x\ [[R]]\ y \wedge M, E, \sigma \models y\ [[S]]\ z]$$

Elementary information descriptors can be composed into complex information descriptors using constructions such as *concatenation*, *conjunction*, *implication*, *disjunction,* and *complement*. These may refer to the fronts alone or both fronts and tails of descriptors. For more details, see (ter Hofstede et al., 1993). In this chapter we use:

$$x\ [[D_1\ D_2]]\ y \quad \equiv \quad \exists_z\ [\ x\ [[D_1]]\ z \wedge z\ [[D_2]]\ y\ ]$$
$$x\ [[D_1 \text{ AND ALSO } D_2]]\ y \quad \equiv \quad \exists_z\ [\ x\ [[D_1]]\ z\ ] \wedge \exists_z\ [x\ [[D_2]]\ z\ ] \wedge x = y$$

where D_1 and D_2 are information descriptors and x, y and z are variables. Some example expression would be:

Person working for Department 'I&KS'
Persons working for department 'I&KS'
Person (working for Department 'I&KS' AND ALSO owning Car of Brand 'Seat')
Persons working for department 'I&KS' who also own a car of brand Seat

Note that the natural language likeness of the ORC expressions used in this chapter can be improved considerably. In the above example we have added a naturalized version of the ORC expression in italics. We intend to develop a formal grammar for a naturalized version of ORC, that has a 1:1 correspondence to basic ("deep") ORC structures. However, such a grammar does not exist yet. We therefore provide ad hoc naturalized expressions, for clarification purposes.

3.5 Rules

ORC has a special way of using information descriptors to describe rules that should apply in a domain. These rules can be used to express constraints and/or business rules. We will use the generic term *rule* for any such expression. Rules consist of information descriptors that are interpreted in a Boolean way; i.e. if no tuple satisfies the relationship, the result is false, otherwise it is true. Some examples of such constructions are:

$$\begin{aligned} [[\text{SOME } D]] &\equiv \exists_{x,y} [\, x\, [[D]]\, y\,] \\ [[\text{NOT } R_1]] &\equiv \neg [[R_1]] \\ [[\text{NO } D]] &\equiv [[\text{NOT SOME } D]] \end{aligned}$$

where D is an information descriptor and R_1 a rule.

4. Graphical representation

Currently, we are experimenting with effective graphical representation of some key classes of temporal dependencies. In (Proper et al., 2005) we have provided some examples using notations inspired by the field of workflow modelling (van der Aalst and ter Hofstede, 2005).

An key modelling construct is the notion of a *life-cycle type*. An example of its use is provided in Figure 2, which contains two inter-linked life-cycle types: Course Offering and Course Attendance. Each of these life-cycle types comprises multiple *action types*.

> *In the example domain, courses are offered to students. In offering a course, a lecturer starts by setting up the course offering. This is followed by the actual lecturing. After lecturing the course, the lecturer sets an exam. This exam is given*

to the students attending the course, after which the lecturer marks the exam papers produced by the students. Students attend the course by enrolling. After their enrolment they attend the course. Once the course is finished, they prepare themselves for the exam, which is following by the actual exam, leading to an exam paper.

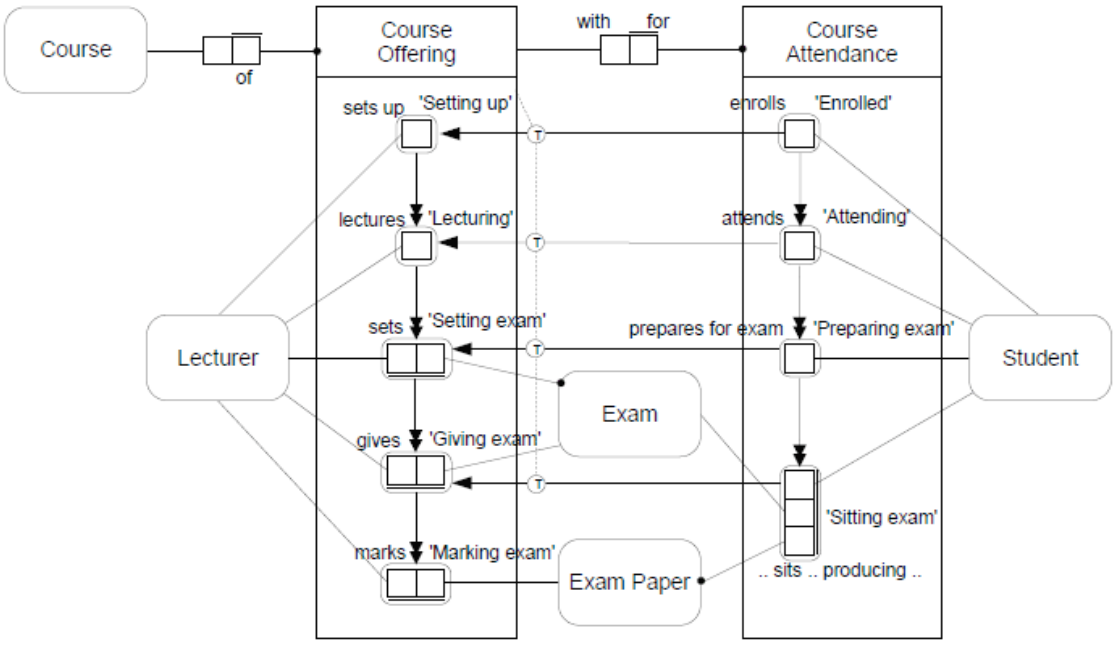

Figure 2: Lecturing example

In general, the *life-cycle type* typically involves multiple *action types*, and can best be regarded as an abbreviation as illustrated in Figure 3. The temporal dependency between *x* and *y* is defined as follows:

$x \longrightarrow\!\!> _S y$ ≡ *x* being act of *S* PRECEDES *y* being act *S*

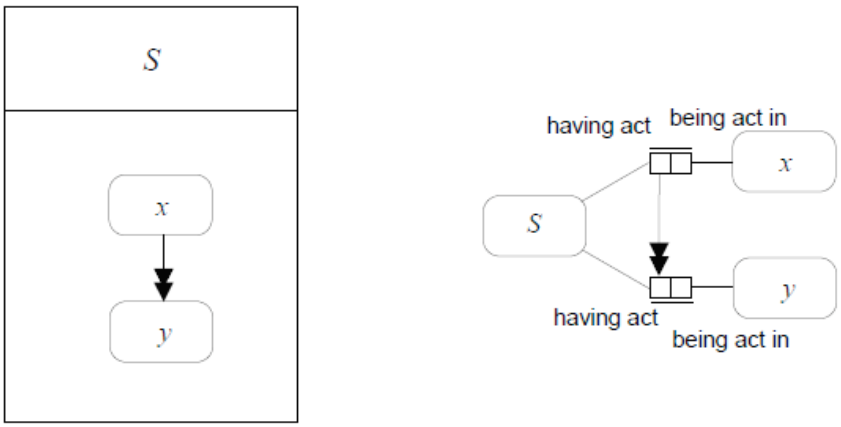

Figure 3: Life cycle types

The enrolment by students in a course should take place *during* the setup phase of a course. This is enforced by means of the temporal subset constraint from Enrolling action type to the Setting up action type. The connection between the temporal subset constraint and the Course Offering life-cycle type signifies that the temporal subset constraint should be evaluated via this object type. In general, the semantics are expressed as:

$$x \subseteq_T y \quad \equiv \quad x \text{ DURING } y$$

In the case of Figure 2, we have specified a join path, leading for example to:

Enrolling being act of Course Attendance for Course Offering
DURING
Setting up being act of Course Offering

Enrolling (which is an act of course attendance, in response to course offering)
takes place during
setting up (which is an act of course offering)

Finally, a model as presented in Figure 2 can be used as a base to derive specialised views such as depicted in Figure 4 focussing on the flow of activities performed by a lecturer.

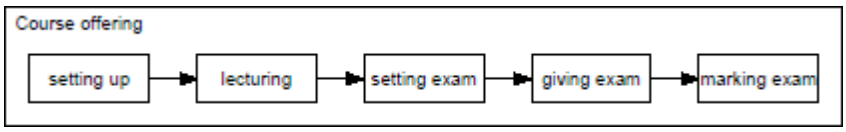

Figure 4: Lecture activities

Conclusion

The research reported in this chapter is part of our effort to find a suitable generalised domain modelling method to model active domains, in view of an ongoing attempt to achieve an integrated domain ontology underlying many viewpoints in conceptual modelling. In this chapter we have focussed on a strategy to apply ORM rigour, and the ORM approach to model elicitation and validation, in modelling active domains. We have introduced the *logbook paradigm* as a history-oriented extension of the traditional natural language orientation of ORM. To be able to define rules governing the behaviour of active domains, we have introduced Object-Role Calculus (ORC). The semantics of this rule language has been defined in terms of Kripke structures. Finally, we have shown how ORM can be extended with graphical constructs, in particular life-cycle types, focussing on temporal dependencies in a domain. This notation allows us to also derive specific views on a domain focussing solely on temporal behaviour, which has been demonstrated.

As made clear earlier, we do not put forward the verbal and graphical notations presented in this chapter as a competitor to existing and well-established techniques for modelling active domains. It is integration we strive for, and we do view ORM and ORC as good candidates for providing a foundation for the integration of many existing, dedicated models and views.

Consequently, validation of our representations in an industrial context seems not quite relevant, and has not been attempted. However, in academic education ORM, ORC, and recently the temporal extension presented in this chapter have been successfully used to teach MSc students in information science the fundamentals of formal conceptual modelling. We found it very helpful indeed to present students with an integrated set of models firmly grounded in a well-understood formalism, aiding them in coming to terms with the many complex issues (both formal and methodological) involved. In addition, our experience is that once the fundamentals have been acquired, students can easily apply them to other modelling techniques and methods, and learn and understand these better and more quickly than their colleagues did some years previous, when an integrated foundation was still lacking in the curriculum (other modelling techniques *are* in fact still taught). However, we admittedly these experiences have so far not been backed up by systematic research. Still, we consider the results good enough to continue

our approach and further develop integrated, ORM-style conceptual modelling as a core around which other modelling techniques and viewpoints are positioned.

As a next exercise, we intend to take some typical patterns from, for example, enterprise modelling and work-flow modelling, and study how to ground them in terms of an underlying ORM domain model with accompanying ORC rules. We expect this to provide further progress in our effort to find a suitable generalised domain modelling method to model active domains.

References

Aalst, W. van der & Hofstede, A. ter (2005). YAWL: yet another workflow language. *Information Systems*, 30(4):245-275.

AECMA - The European Association of Aerospace Industries (2001). *AECMA Simplified English - A guide for the preparation of aircraft maintenance documentation in the international maintenance language*. Issue 1, Revision 2 [Online]. Available: http://www.aecma.org.

Allen, G.N. & March, S.T. (2003). Modeling Temporal Dynamics for Business Systems. *Journal of Database Management* 14(3): 21-36 (2003)

Bloesch, A. & Halpin, T. (1996). ConQuer: A Conceptual Query Language. In Thalheim, B., editor, *Proceedings of the 15th International Conference on Conceptual Modeling (ER'96), Cottbus, Germany, EU*, volume 1157 of *Lecture Notes in Computer Science*, pages 121-133, Berlin, Germany, EU. Springer.

Chellas, B. (1980). *Modal logic: an introduction*. Cambridge University Press, Cambridge, United Kingdom, EU.

Chen, P. (1976). The Entity-Relationship Model: Towards a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9-36.

Dietz, J. L. (2005). A World Ontology Specification Language. In Meersman, R., Tari, Z., and Herrero, P., editors, *On the Move to Meaningful Internet Systems 2005: OTM Workshops - OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, GADA, MIOS+INTEROP, ORM, PhDS, SeBGIS, SWWS, and WOSE 2005, Agia Napa, Cyprus, EU*, volume 3762 of *Lecture Notes in Computer Science*, pages 688-699, Berlin, Germany. Springer-Verlag. ISBN 3540297391 doi:10.1007/11575863_88.

Elmasri, R. & Navathe, S. (1994). Advanced Data Models and Emerging Trends. In *Fundamentals of Database Systems*, chapter 21. Benjamin Cummings, Redwood City, California, USA. Second Edition.

Embley, D., Kurtz, B., & Woodfield, S. (1992). *Object-Oriented Systems Analysis - A model-driven approach*. Yourdon Press, New York, New York, USA. ASIN 0136299733.

Farrington, G. (1996). *An Overview of the International Aerospace Language*.

Frankel, D. (2003). *Model Driven Architecture: Applying MDA to Enterprise Computing*. Wiley, New York, New York, USA.

Frederiks, P. (1997). *Object-Oriented Modeling based on Information Grammars*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU. ISBN 9090103384.

Frederiks, P. & Weide, T. van der (2002). Deriving and paraphrasing information grammars using object-oriented analysis models. *Acta Informatica*, 38(7):437-88.

Fuchs, N. & Schwitter, R. (1996). Attempto Controlled English (ACE). In *Proceedings of the First International Workshop on Controlled Language Applications (CLAW96). Leuven, Belgium*, pages 124-136. Katholieke Universiteit Leuven Centre for Computational Linguistics.

Gogolla, M. (1994). An Extended Entity-Relationship Model: Fundamentals and Pragmatics, volume 767 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, EU.

Halpin, T. (2001). *Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design*. Morgan Kaufmann, San Mateo, California, USA. ISBN 1558606726.

Hofstede, A. ter, Proper, H.A., & Weide, T. van der (1993). Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489-523.

Hofstede, A. ter, Proper, H.A., & Weide, T. van der (1997). Exploiting Fact Verbalisation in Conceptual Information Modelling. *Information Systems*, 22(6/7):349-385.

Hoppenbrouwers, S. (2003). *Freezing Language; Conceptualisation processes in ICT supported organisations*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU. ISBN 9090173188.

Hoppenbrouwers, S., Proper, H.A., & Weide, T. van der (2005a). A Fundamental View on the Process of Conceptual Modeling. In Delcambre, L., Kop, C., Mayr, H., M ylopoulos, J., and Pastor, O., editors, *Conceptual Modeling - ER 2005 - 24 International Conference on Conceptual Modeling, Klagenfurt, Austria, EU*, volume 3716 of *Lecture Notes in Computer Science*, pages 128-143, Berlin, Germany. Springer-Verlag. ISBN 3540293892 doi:10.1007/11568322_9.

Hoppenbrouwers, S., Proper, H.A., & Weide, T. van der (2005b). Fact Calculus: Using ORM and Lisa-D to Reason About Domains. In Meersman, R., Tari, Z., and Herrero, P., editors, *On the Move to Meaningful Internet Systems 2005: OTM Workshops - OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, GADA, MIOS+INTEROP, ORM, PhDS, SeBGIS, SWWS, and WOSE 2005, Agia Napa, Cyprus, EU*, volume 3762 of *Lecture Notes in Computer Science*, pages 720-729, Berlin, Germany. Springer-Verlag. ISBN 3540297391 doi:10.1007/11575863_91.

ISO (1987). *Information processing systems - Concepts and Terminology for the Conceptual Schema and the Information Base*. ISO/TR 9007:1987 [Online]. Available: http://www.iso.org.

Khatri, V., Ram, S., & Snodgrass, R.T. (2004): Augmenting a Conceptual Model with Geospatiotemporal Annotations. *IEEE Trans. Knowl. Data Eng*. 16(11): 1324-1338.

Kristen, G. (1994). *Object Orientation - The KISS Method, From Information Architecture to Information System*. Addison Wesley, Reading, Massachusetts, USA. ISBN 0201422999.

Lankhorst, M. (Editor). (2005). *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Berlin, Germany, EU. ISBN 3540243712.

Lipeck, U. and Saake, G. (1987). Monitoring dynamic integrity constraints based on temporal logic. *Information Systems*, 12(3):255-269.

March, S.T. & Allen, G.N. (2003). On The Representation Of Temporal Dynamics. *Advanced Topics in Database Research*, Vol. 2 2003: 37-53;

Meersman, R. (1982). *The RIDL Conceptual Language*. Technical report, International Centre for Information Analysis Services, Control Data Belgium, Inc., Brussels, Belgium, EU.

Nijssen, G. & Halpin, T. (1989). *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Englewood Cliffs, New Jersey, USA. ASIN 0131672630.

OMG (2003). UML 2.0 *Superstructure Specification - Final Adopted Specification*. Technical Report ptc/03-08-02. http://www.omg.org.

Paulk, M., Curtis, B., Chrissis, M., & Weber, C. (1993). *Capability Maturity Model for Software*, Version 1.1. Technical Report SEI-93-TR-024, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

Proper, H.A. (1994a). *A Theory for Conceptual Modelling of Evolving Application Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU. ISBN 909006849X.

Proper, H.A. (1994b). *ConQuer-92 - The revised report on the conceptual query language LISA-D*. Technical report, Asymetrix Research Laboratory, University of Queensland, Brisbane, Queensland, Australia.

Proper, H.A., Bleeker, A., & Hoppenbrouwers, S. (2004). Object-Role Modelling as a Domain Modelling Approach. In Grundspenkis, J. and Kirikova, M., editors, *Proceedings of the Workshop on Evaluating Modeling Methods for Systems Analysis and Design (EMMSAD'04), held in conjunction with the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004),*, volume 3, pages 317-328, Riga, Latvia, EU. Faculty of Computer Science and Information Technology. ISBN 9984976718.

Proper, H.A., Hoppenbrouwers, S., & Weide, T. van der (2005). A Fact-Oriented Approach to Activity Modeling. In Meersman, R., Tari, Z., and Herrero, P., editors, *On the Move to Meaningful Internet Systems 2005: OTM Workshops - OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, GADA, MIOS+INTEROP, ORM, PhDS, SeBGIS, SWWS, and WOSE 2005, Agia Napa, Cyprus, EU*, volume 3762 of *Lecture Notes in Computer Science*, pages 666-675, Berlin, Germany. Springer-Verlag. ISBN 3540297391 doi:10.1007/11575863_86.

Proper, H.A. & Weide, T. van der (1994). EVORM - A Conceptual Modelling Technique for Evolving Application Domains. *Data & Knowledge Engineering*, 12:313-359.

Proper, H.A. & Weide, T. van der (2005). Modelling as Selection of Interpretation. In *Modellierung 2006*, volume P82 of *Lecture Notes in Informatics*, pages 223-232. ISBN 3885791765.

Reijswoud, V. van & Dietz, J. (1999). *DEMO Modelling Handbook*, volume 1. Delft University of Technology, Delft, The Netherlands, EU, 2nd edition.

Schwitter, R. (2004). *Controlled Natural Languages*. Centre for Language Technology, Macquary University, Sydney, New South Wales, Australia [Online]. Available: http://www.ics.mq.edu.au/rolfs/controlled- natural- languages/.

Spyns, P. (2005). Object Role Modelling for Ontology Engineering in the DOGMA Framework. In Meersman, R., Tari, Z., and Herrero, P., editors, *On the Move to Meaningful Internet Systems 2005: OTM Workshops - OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, GADA, MIOS+INTEROP, ORM, PhDS, SeBGIS, SWWS, and WOSE 2005, Agia Napa, Cyprus, EU*, volume 3762 of *Lecture Notes in Computer Science*, pages 710-719, Berlin, Germany. Springer-Verlag. ISBN 3540297391 doi:10.1007/11575863_90.

Spyns, P., Meersman, R., & Jarrar, M. (2002). Data modelling versus ontology engineering. *ACM SIGMOD Record*, 31(4):12-17.

Wijers, G. & Heijes, H. (1990). Automated Support of the Modelling Process: A view based on experiments with expert information engineers. In Steinholz, B., Sølvberg, A., and Bergman, L., editors, *Proceedings of the Second Nordic Conference CAiSE'90 on Advanced Information Systems Engineering, Stockholm, Sweden, EU*, volume 436 of Lecture Notes in Computer Science, pages 88-108, Berlin, Germany, EU. Springer. ISBN 3540526250.