Concepts and Strategies for Quality of Modeling

P. (Patrick) van Bommel[1], S.J.B.A. (Stijn) Hoppenbrouwers[1],
H.A. (Erik) Proper[2], and J. (Jeroen) Roelofs[1]

[1] Institute for Computing and Information Sciences,
Radboud University Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands
[2] Capgemini, Papendorpseweg 100, 3500 GN Utrecht, The Netherlands
{p.vanbommel, s.hoppenbrouwers}@cs.ru.nl;
e.proper@acm.org; jeroenroelofs@chello.nl

Abstract A process-oriented framework (QoMo) is presented that aims to further the study of analysis and support of processes for modelling. The framework is strongly goal-oriented, and expressed largely by means of formal rules. The concepts in the framework are partly derived from the SEQUAL framework for quality of modelling. A number of modelling goal categories is discussed in view of SEQUAL/QoMo, as well as a formal approach to the description of strategies to help achieve those goals. Finally, a prototype implementation of the framework is presented as an illustration and proof of concept.

1 Introduction

This chapter aims to contribute to the area of conceptual modeling quality assessment and improvement, in particular by providing some fundamental concepts concerning the quality of the *process* of modeling, and for structured description of *ways of achieving* quality models. Though operationalization of the concepts and strategies is still limited in this version of the framework, an initial application has been realized and is discussed.

There is a clear link between the work presented and the field of Situational Method Engineering. In particular, the basic idea of combining (patterns of) language related aspects of methods with process related aspects is commonplace in method engineering (see for example Mirbel and Ralyté, 2006; Ralyté et al., 2007). We believe the specific contribution of the current chapter lies in its formal, rule-based nature, and a strong emphasis on combinations of rather specific modeling goals. Also, we focus only on modeling, whereas method engineering in general also covers other activities in systems engineering. Finally, we choose a relatively fine-grained view on the activity of modeling, whereas method engineering generally deals with process aspects only at the level of clearly distinguishable *phases* (i.e. has a more course-grained view, which is not to say that such a view is not a very useful one in its own right).

We first present a process-oriented 'Quality of Modeling' framework (QoMo), which for a large part is derived from the established SEQUAL framework for quality of models. QoMo is based on knowledge state transitions, the cost of the activities bringing such transitions about, and a goal structure for activities-for-modeling. Such goals are directly linked to concepts of SEQUAL.

We then proceed in two steps. In the first, generic step (section 6) we consider the underlying generic structure of strategies for modeling. We discuss how QoMo's goals for modeling can be linked to a rule-based way of describing processes for modeling. Such process descriptions hinge on *strategy frames* and *strategy descriptions*, which may be used descriptively (for studying/analyzing real instances of processes) as well as prescriptively (for the guiding of modeling processes). We present a set of concepts for describing quality-oriented strategies.

In the second, implementation step (section 7) we consider an example implementation involving a concrete operational workflow language. We present results of a case study in which a specialized version of our generic framework is applied to the description of an elementary method for requirements modeling, as taught in the 2nd year of an Information Science Bachelor's curriculum. We discuss and exemplify how concepts from the generic framework were used, and in some cases how they were amended to fit the task at hand.

2 Background

Interest in frameworks for quality and assessment of conceptual models has been gradually increasing for a number of years. A generic overview and discussion can be found in (Moody, 2006). A key framework for analysis of the quality of conceptual models is the SEQUAL framework (Krogstie et al., 2006; Krogstie, 2002; Krogstie and Jorgesen, 2002) . This framework takes a semiotics-based view on modeling which is compatible with our own (Hoppenbrouwers et al., 2005a). It is more than a quality framework for models as such, in that it includes not just the model but also the knowledge of the modelers, the domain modeled, the modeling languages, agreement between modelers, etc. (see section 2); it bases quality assessment on relations between such model-related items, i.e. respects the broader context of the model.

As argued in (Hoppenbrouwers et al., 2005b), in addition to analysis of the quality of models, the *process* of which such models are a product should also be taken into account. We briefly summarize the main arguments here:

1. Though some have written about detailed stages in and aspects of "Ways of Working" in modeling, i.e. its process or procedure (for example, see Halpin, 2001), the detailed "how" behind the activity of creating models

is still mostly art rather than science. There is, therefore, a purely scientific interest in improving our understanding of the operational details of modeling processes.
2. In addition, such a study should enable us to find ways of improving the modeling process (for example, its quality, efficiency, or effectiveness; from a more methodological angle: reproducibility, stability, and precision; also traceability, and so on).
3. Indeed, some aspects of quality, it seems, can be better achieved through a good modeling process than by just imposing requirements on the end product and introducing a feedback cycle (iteration). This holds in particular (though not exclusively) for matters of validation and grounding in a socio-political context.
4. A score of more practical arguments follow from the ones above. For example, for improvement of case tool design, a more process-oriented approach seems promising.

As mentioned, models as such are not the only product of a modeling process. The related knowledge development, agreements, etc. are arguably as important. Hence, our process-oriented view suggests that we take aboard such additional *model items*, and quality concepts related to them. The latest SEQUAL version explicitly allows for this approach, though it does not make explicit use of a meta-concept such as "model item".

The importance of the modeling process in view of model quality is commonly confirmed in the literature in general (Poels et al, 2003; Nelson and Monarchi, 2007). If we want to evaluate a modeling process, we can take (at the least) the following four different points of view:
1. Measure the success of the process in fulfilling its goals. This boils down to using the current SEQUAL framework as-is, and directly link such an analysis of the model items to the success of the process. However, one might also analyze *intermediate steps* in the process against intermediate or sub-goals set. By evaluating partial or intermediary products (possibly in view of a prioritization of goals), the process may be *steered* along the way. Also, the steps described will eventually become so small that *essential* modeling goals/activities can perhaps be identified (steps in the detailed thinking process underlying modeling), opening up the black box of the modeling process.
2. Calculate the cost-benefit ratio: achievements set against the cost. Such cost again can hold for the process as a whole, but also for specific parts.
3. We can look at the process and the working environment as an information system. This then is a 2nd order information system: an IS that serves to develop (bring forth) information systems. The information system underlying the modeling process (probably but not necessarily including IT support) can be evaluated in a way similar to evaluation of information

systems in general. In particular, aspects like usability and actability but also traceability and even portability are relevant here.

4. Views 1-3 concern operational evaluations of particular process instantiations. At a higher level of abstraction, we can also look at properties and control aspects of a process in terms of, for example, repeatability, optimization, etc. (Chrissis et al., 2006).

In this paper, we focus on 1 and 2. View 3 depends very much on implementation and support of some specific process (in particular, tooling), which is outside the grasp of our current study. View 4 is essential in the long run yet stands mostly apart from the discussion in this paper, and will not be elaborated on any further now. Admittedly, many fundamental aspects of process quality (process control) are expected to be covered by 4, rather than 1 and 2. However, 1 and 2 do provide the concepts we direly need for applying 4 in any concrete way. This paper, therefore, is arguably a SEQUAL-based 'step up' towards analysis at the level of viewpoint 4.

Our main contribution in this chapter twofold: 1. a preliminary version of a framework for Quality of Modeling (QoMo), which not only takes into account the products of modeling but also *processes*. Analogous to SEQUAL, QoMo is not yet an operational method for model-oriented quality assessment and management. The framework is expected to evolve as our knowledge of dealing with quality of modeling processes increases. 2. We provide a set of concepts for capturing the basics of strategies for realizing the QoMo goal. The concepts are made operational for the first time in context of a case implementation.

3 The SEQUAL framework: overview

Since we cannot, nor wish to, present an elaborate overview or discussion of the SEQUAL framework, we will provide a short summary of its key concepts, as based on its latest substantial update (Krogstie et al., 2006). However, we take into account all SEQUAL concepts, including those not explicitly mentioned in that update. We have rephrased some of the definitions or added some explanations/interpretations of our own, yet made sure we did not stray from the intentions of the SEQUAL framework.

SEQUAL Model Items:

- **G**: goals of modeling (normally organizationally defined).
- **L**: language extension; set of all statements that are syntactically correct in the modeling languages used.
- **D**: the domain; the set of all statements that can be stated about the situation at hand.

- **D^O**: the optimal domain; the situation the organization would or should have wanted –useful for comparison with the actual domain D in order to make quality judgments.
- **M**: the externalized model; the set of all statements in someone's model of part of the perceived reality written in a language.
- **K_s**: the relevant knowledge of the set of stakeholders involved in modeling (i.e. of the audience at large).
- **K_m**: a subset of K_s; the knowledge of only those stakeholders actively involved in modeling.
- **K^N**: knowledge need; the knowledge needed by the organization to perform its tasks. Used for comparison with K_s in order to pass quality judgments.
- **I**: the social actor interpretation, that is, the set of all statements that the audience thinks that an externalized model consists of.
- **T**: the technical actor interpretation, that is, the statements in the model as 'interpreted' by the different modeling tools.

SEQUAL quality definitions:

- **Physical quality**: how the model is physically represented and available to stakeholders; a matter of *medium*.
- **Empirical quality**: how the model comes across in terms of *cognitive ergonomics*, e.g. layout for graphs and readability indexes for text.
- **Syntactic quality:** conformity to the syntax of the modeling language, involving L.
- **Semantic quality**: how well M reflects K_s.
- **Ideal descriptive semantic quality**: Validity: $M/D=\varnothing$; Completeness: $D/M=\varnothing$.
- **Ideal prescriptive semantic quality**: Validity: $M/D^O=\varnothing$; Completeness: $D^O/M=\varnothing$.
- **Domain quality**: how well the domain fits some desired situation: D compared with D^O.
- **Quality of socio-cognitive interpretation**: how an individual or group interprets the model, i.e. how I matches M, in view of how M was intended to be interpreted by one or more of its modelers.
- **Quality of technical interpretation**: similarly, how a tool or group of tools interprets the model, i.e. how T matches M.
- **Pragmatic quality -actability**: how the model, or the act of modeling, influences the actability of the organization. Note that this enables description of the effect of the modeling process even in case the model as such is discarded.
- **Pragmatic quality -learning**: how the modeling effort and/or the model as such contribute to organizational learning.

- **Knowledge quality**: how well actual knowledge K_s matches knowledge need K^N.
- **Social quality**: the level of agreement about the model among stakeholders (individuals or groups) about the statements of M.

4 Product goals and process goals

There has been some preliminary process oriented work related to the use of SEQUAL (e.g. going back to Sindre and Krogstie, 1995). Our current proposal concerns a strongly goal-oriented approach. The model items and qualities of the SEQUAL framework can be used as an abstract basis for expressing product quality of a model process, and alternatively to specify goals for model quality. Note that during a modeling process, each model item (for example the model (M), stakeholder knowledge (K_s), or the state of agreement about M (Covered by "Social quality" in SEQUAL) may change, in principle, at any step in the process. The SEQUAL framework can therefore be used not only for expressing how well a process as a whole has done in terms of achieving its goals, but also which specific steps or series of steps in the process have contributed to specific advances in achieving specific (sub)goals. We can thus directly use SEQUAL concepts for expressing *product goals*: both *product end-goals* and *intermediary product goals*.

In addition, it should be possible to link product goals and sub-goals, and the level of achievement in view of these goals, to the notion of *benefit*, and weigh this against its *cost*. Note that cost (for example in terms of time or money) is something that can be especially well calculated in view of a work process and the people and resources involved in it. This entails that, as is common in process modeling, account should be taken of the tasks or actions involved as well as the people (typically, *roles*) performing them. Both the cost of the entire process and, again, the cost of steps/parts in the process can then be calculated. This entails that the cost-benefit ratio for an entire process, or parts of it, can be calculated. As argued earlier, this is a useful way of evaluating a modeling process.

In (van Bommel et al., 2006) we presented an initial list of *modeling goals* (slightly amended here) of which the relation to SEQUAL will be made clear. The goals are, or course, generically covered by G in SEQUAL, but they also relate to most of the other SEQUAL concepts.
Usage goals (including actability and knowledge goals): These stand apart from our "modeling goals": they represent the *why* of modeling; the modeling goals represent the *what*. In SEQUAL, the usage goals are covered primarily by the Pragmatic qualities (both learning and actability) and, related to the former, Knowledge quality. The overall cost-benefit ratio will mostly relate to

Usage goals, but optimizing (aspects of) modeling methods in an operational sense requires us to look at the other goals, the "modeling goals":

Creation goals (list of model items/deliverables): This relates to what we might generalize as "required deliverables": M, in a very broad sense (i.e. also including textual documents etc.). If made explicit, K_s and/or K_m are to be included here. Creation goals are primarily related to the SEQUAL notions of *completeness* (as part of "Ideal descriptive/prescriptive semantic quality") and *validity* as defined under "Ideal descriptive/prescriptive semantic quality". Note that "Completeness" in an operational sense would in fact be defined as $K_s/M = \varnothing$ ((Krogstie et al., 2006) has it as $M/D = \varnothing$). Validity would then be $M/K_s = \varnothing$. There is a complication, however, because some definitions of validity also strongly involve Social Quality (see Validation goals below), linking validation with levels of agreement with (parts of) the model by specific actors. We observe that SEQUAL allows us to differentiate between these two notions of validity, and yet combine them.

Validation goals: These are related to Social Quality: the level and nature of agreement between stakeholders concerning the model. As discussed, our analysis allows us to differentiate between two common notions of "validity": one now falling under Creation Goals, one (the one related to Social Quality) under Validity Goals.

Argumentation goals: In some cases, arguments concerning particular model items may be required. Though a weak link with Social Quality can be suggested here, it seems that this type of modeling goals is not as of yet explicitly covered by SEQUAL. Argumentation goals arguably are an extension of Validation goals.

Grammar goals: Language (L) related: concerns syntactic quality.

Interpretation goals: Related to quality of socio-cognitive interpretation, and possibly also to technical interpretation. The latter may be covered by Grammar goals if the language and its use are fully formal and therefore present no interpretation challenges whatsoever. Note that Interpretation Goals may be seen as a refinement of Validation Goals.

Abstraction goals: This is as of yet a somewhat obscure category. It boils down to the question: does the model (or parts of it) strike the right level of abstraction? This seems to be a crucial matter, but also one that is terribly hard to operationalize. There seems to be a link with Semantic quality (and it is a different link than the one covered by Creation Goals), but its precise nature is yet unclear to us. Quite probably, Abstraction Goals are sub-goals of Usage Goals, related to the utilitarian relevance of various levels of and details in models.

5 Achieving goals by means of strategies

Given usage goals, modeling goals, and a modeling environment, strategies can be formulated to best execute the modeling process. In other words:

Usage goal + Modeling goal + Modeling environment ⇒ Modeling strategy

Moving to concepts that are more specifically related to actual modeling processes, we will now briefly present an approach to describing the detailed steps in modeling processes. This approach can be used either descriptively or prescriptively.

It is customary in run-of-the-mill method description to view procedures or processes in terms of fairly simple flows or "steps". In view of the many entwined goals and sub-goals at play in modeling, and the different sorts of equally entwined actions taken to achieve these, it seems more helpful at a fundamental level to let go of (work)flow in the traditional sense as a metaphor for modeling procedures. Instead, we advocate a rule-based approach in which an analysis of the states of all relevant model items, in combination with a set of rules describing strategies, leads to "run-time" decisions on what to do next. Importantly, the framework is capable of capturing *ad hoc* activities of modeling as well as tightly pre-structured ones. In order to achieve this, we define constraints on states and activities, i.e. not necessarily fully specified goals and strategies. As for workflow-like procedures: these are likely to play a role at some level, as reference procedures. However, we believe that while such procedures may be valid instruments for supporting a modeling process, they are not so well suited to capture or reflect actual, individual modeling processes.

Our view of methods and the processes they govern takes shape in the use of the following concepts. These concepts are based on some initial studies of modeling processes (for example, Hoppenbrouwers et al., 2005b).

Results of steps in a method are *states*. States are typically described by means of some logic, or a semi-natural verbalization thereof. *Goals* are a subtype of states, and typically describe desired states-of-affairs in the project/domain; *situations* are also a sub-type of state, that describe states-of-affairs that contribute towards achieving their related goal(s). Importantly, situations are not mere negative versions of a goal they are linked with: they are positively stated descriptions of a state of affairs that is a way point on the way to achieving a goal. If no helpful state-of-affairs can be identified, the situation is "void": no particular precondition state is assumed.

States cannot be labeled situations or goals in an absolute sense; such labeling is linked to the use of some state in relation to some *transition(s)* and some

other state(s). In other words, a state that is a situation in view of some goal can also be a goal in view of another situation. Combinations of states and transitions can be graphically depicted by means of plain directed graphs. States can be described at type level as well as at instance level. This means that a state description may be an abstract description of one or more aspects of actual (instantiated) states. Situation descriptions as well as goal descriptions are typically used as abstract patterns to be matched against factual (instance level) descriptions of the method domain as the operational process takes place.

The combination of, minimally, two states (a situation and a goal) linked by a transition, we call a strategy *context*[1]. Complex strategy contexts are allowed; they are composed of three or more states, and two or more transitions. A *strategy* is a course of action associated with one or more strategy contexts. We distinguish three types of strategy in our framework for method description:
Ad hoc strategies. In ad hoc strategies, no concrete modeling guidelines are available. We know what we have (input) and we have an indication of what we should produce (output), but we do not know how we should produce this.
Guided strategies. In guided strategies, a concrete guided description of the actions to be performed is available. This description can for example have the form of (a) a piece of text in natural language, (b) a piece of text in some controlled language such as a subset of natural language (c) an algorithm written in pseudo code or (d) a graphical notation (for example, a workflow).
Composed strategies. A composed strategy consists of more (sub) strategies. The nesting structure of composed strategies may be cyclic, as long as eventual termination by means of ad hoc or guided strategies is guaranteed.

Some very basic notions of *temporality* are also part of our framework. A strategy (complex or not) may be defined as preceding another strategy: "strategy s occurs before strategy t". This may be captured in a (logical) rule constraining the occurrence in time of s and t. It allows for other strategies to be executed between execution of s and of t. In addition, a stricter notion of immediate order is used: if a strategy t is defined as occurring immediately after strategy s, this means that no strategy may be executed between the execution of s and t. For the moment, these simple temporal notions are all we need for our purposes.

[1] We use a broad definition of "strategy". This term corresponds closely to what, for example, (Mirbel and Ralyté, 2006) call "guidelines". We use our own terminology for the moment because our current discussion has a relatively generic focus on method modeling, but this is not to say we reject Mirbel and Ralyté's conceptual distinctions as such. We acknowledge that proper alignment of terminology will require due attention in the near future.

The basic concepts being in place, we need to refine those generic concepts for more specialized categories of strategy. For this, we use the goals discussed in the previous section. Note that the distinction in goal types is based strictly on the terms in which their states are described, and that the various strategy/goal types can be combined and even tightly interwoven in the fabric of a method model. For example, validation strategies can be combined with creation strategies at a low level; alternatively, validation may be seen as an activity separated in time from creation. To our regret, in the remainder of this chapter we only have space to consider examples for two goal types: grammar goals and creation goals. We emphasize, however, our conviction that the rule-based approach to method modeling can be successfully used beyond the detailed study and guidance of modeling in some particular modeling language.

6 A Generic Rule-Based Meta-Model for Methods and Strategies

We now first focus on generic descriptions of composed modeling strategies. Our examples in this section concern grammar strategies related to a technique for formal conceptual modeling called Object Role Modeling (Halpin, 2001). We then proceed with the introduction of the descriptive concepts of strategy space and strategy frame, and we elaborate on these in order to achieve further refinement.

6.1 Basic description of (grammar) strategies

In this subsection we focus on basic grammar-related strategy descriptions. Trivial though the differences between grammar strategies may seem to the casual observer, they have given rise to considerable debate among experienced modelers, and even schisms between schools of conceptual modeling that are in principle closely related. In addition, novice modelers often feel uncertain about ways to proceed in modeling. Where to start? Where to go from there? What to aim for? Similar questions are raised in any modeling situation, for any modeling language. In practice, several main ORM grammar strategies have been observed to co-exist. Several styles are in fact used in the literature. We consider some styles which are in part reflected in ORM-related literature and practice. Here we first consider an "object-type-driven" modeling strategy. This strategy starts with the introduction of "object types":

Strategy description 1 *Object-type-driven strategy (basic).*
a. Provide object types.
b. Provide fact types.
c. Provide facts and constraints.
d. Provide objects.

Thus, strategy description 1 focuses on grammar goals, where a (partial) modeling procedure is initiated by the specification of object types, followed by fact types. Next, fact instances and constraints are specified. The modeling procedure is concluded by providing the necessary object instances. Modeling of other styles of object-type driven strategies is of course also possible, for example if the requirement is posed that constraints and object instances are specified before fact instances. However, such an approach goes beyond existing methodological descriptions of ORM modeling. Note that we do not claim that the "object-type-driven strategy" is superior (or inferior, for that matter) to other strategies; finding out which strategy works best in which situation concerns a research question not addressed here.

We make a distinction between model-items and model-item-kinds. Items (instance level) reflect the model-in-progress, whereas item kinds (type level) stem from the method (in this case, the modeling language) used for describing the domain. Examples of items in our grammar strategy are: 'John', 'person', 'John works for the personnel department', and 'persons work for departments'. The corresponding item kinds are: 'object', 'object type', 'fact', and 'fact type', respectively.

6.2 Strategy space

In order to facilitate effective definition of composed modeling strategies we define the notion of strategy space, which is a light-weight representation of the meta-model of a modeling language. For other approaches to meta-modeling, see for example (Kensche et al., 2005; OMG, 2001; Saeki, 1995). A strategy space P is defined as a structure $P = \langle S,T \rangle$ consisting of states S and transitions T. States are associated with item kinds. We assume $T \subseteq S \times S$ and a transition $\langle x,y \rangle$ has $x \neq y$. A strategy space provides the context for setting up strategies, but also for refining and using strategies.

Example 1.
As an example, figure 1 presents the strategy space p_1. This space contains the following states: object, fact, object type, fact type, and constraint. It contains the transitions c0 , . . . , c9 , where for instance c4 = <object,fact>. In section 6.3 we show how strategy description 1 is embedded within the space p1. When drawing strategy spaces, a transition is drawn as an arrow. When writing S and T we assume that the corresponding space P is clear from the context. If this is not the case, we write SP and TP or $S(P)$ and $T(P)$.

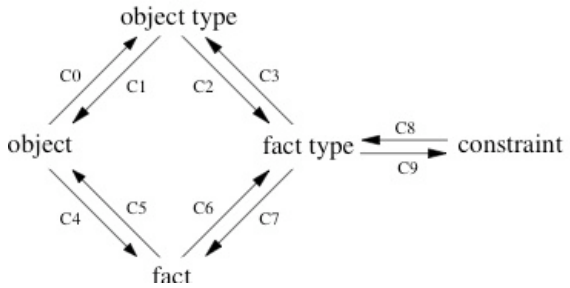

Figure 1: Example strategy space p_1

6.3 Strategy frame

Many different composed modeling strategies can be described in the context of a single strategy space. In order to embed the various strategies within a single strategy space, we define the notion of **strategy frame**. A strategy frame *F* in the context of a space *P* is a spanning subgraph of *P*:

– A frame contains all states, so *SF* = *SP*.
– A frame contains some of the transitions, so TF ⊆TP.

The set of all frames of a strategy space *P* is denoted as frames(*P*). Note that a frame is not necessarily connected.

Example 2.
As an example, the left hand diagram of figure 2 presents the strategy frame f1 ∈ frames(p1). The frame f1 is used as underlying structure of strategy description 1. In the strategy space p1 the frame f1 focuses on an object-type-driven strategy by first specifying object types, followed by fact types via transition c2. Then constraints are specified via transition c9 and facts and objects are specified via transitions c7 and c5.

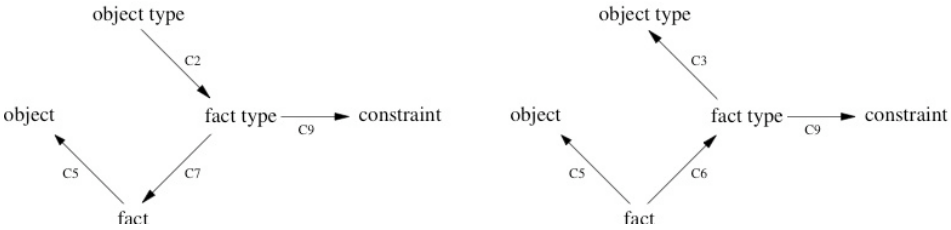

Figure 2: Example strategy frames f1 and f2

In basic grammar strategy descriptions we usually have a breadth-first traversal of the underlying strategy frame. Refinements of this default approach are discussed later; see for example section 6.7.

According to the definitions of strategy space and strategy frame, each strategy space is a strategy frame as well, or P ∈ frames(*P*). Also, each strategy frame can be used as a new strategy space consisting of subframes. This enables the systematic treatment of sub-strategies.

Example 3.
If the frame f1 in figure 2 were used as a new strategy space, we would have several possible subframes. We have, however, only a single connected subframe in this case, which is the frame f1 ∈ frames(f1).

6.4 The rules underlying a strategy frame

We now provide an example of the rules underlying strategy frame f1 . The relevant goal descriptions for strategy frame f1 are the following[2].

G1 There is at least one FactType

This is an instance-level goal. The rest of the goals are type-level; they express syntactic requirements as dictated by the ORM meta-model (note that we use a mere selection from the complete set of rules that define the meta-model). The other goal-rules relevant to f1 are:

G2 Each FactType is populated by at least one Fact
G3 Each ObjectType belongs to at least one FactType
G4 Each FactType is constrained by zero or more Constraints
G5 Each Fact has at least one Object

Next, we define the one situation that is relevant to f1:

S1 There is at least one ObjectType

So we assume that one or more ObjectTypes have already been identified (presumably as a goal of another strategy) and that these ObjectTypes are used as input for f1. Next, a series of contexts is defined, consisting of states (either situations or goals) linked by a transition. Note that these related contexts are a 1:1 reflection of f1. The states are selected from the rules above.

[2] For sake of readability, we use simple natural language statements for state/rule description, in ORM verbalization style. Each of these descriptions can in be easily represented in logic.

The transitions are verbalized as “SHOULD LEAD TO”.

S1 There is at least one ObjectType
C2 SHOULD LEAD TO
G3 Each ObjectType belongs to at least one FactType
G3 Each ObjectType belongs to at least one FactType
C7 SHOULD LEAD TO
G2 Each FactType is populated by at least one Fact
G2 Each FactType is populated by at least one Fact
C5 SHOULD LEAD TO
G5 Each Fact has at least one Object
G3 Each ObjectType belongs to at least one FactType
C9 SHOULD LEAD TO
G4 Each FactType is constrained by zero or more Constraints

So far, our definition does not include temporal ordering. Adding this, we get:

C2 occurs before C7
C2 occurs before C9
C7 occurs before C5
C5 occurs immetiatelyAfter C7

These temporal rules will be elaborated on below. The transitions are to be linked to further strategies (ad hoc, guided, or composed, as explained in section 3), which suggest how each particular transition is to be achieved. This concludes our example. For reasons of space, we will not define other strategy frames at rule level, as the same descriptive principles hold across all examples.

6.5 Another composed modeling strategy

Next we consider a fact-driven strategy. In this strategy, is is mandatory that facts are introduced first. Suppose we have the following basic strategy description:

Strategy description 2 *Fact-driven strategy (basic).*
a. Provide facts.
b. Provide fact types and objects.
c. Provide object types and constraints.

Note that strategy description 2 is indeed fact-driven rather than fact-type-driven. A fact-type-driven strategy would require the specification of fact types prior to the specification of fact instances. Comparison of strategies is considered in more detail in section 6.6.

Example 4.
The right-hand side of figure 2 presents the strategy frame f2 ∈ frames(p1).

The frame f2 is used as underlying structure of strategy description 2. In the strategy space p1 the frame f2 focuses on the specification of facts including their types via transition c6 and their objects via transition c5, followed by the specification of object types via transition c3 and constraints via transition c9.

6.6 Comparison of modeling strategies

In order to compare composed modeling strategies, we examine the differences between their underlying frames. We first consider the reversal of individual transitions by the reversal operator rev. Let $x = \langle y,z \rangle \in T$ be a transition. Then the effect of rev_x is that $\langle y,z \rangle$ is removed and a new transition $\langle z,y \rangle$ is added.

Example 5.
As an example we compare the frames of object-type-driven and fact-driven strategies. We see that these frames are quite similar. No transitions are added or deleted, and some transitions are reversed while others are not. Using the reversal operator we thus may have:

$$\mathrm{f1} = \mathsf{rev}_{c6}\,(\mathsf{rev}_{c3}\,(\mathrm{f2}))$$

In the above example only a selection of individual transitions has been reversed. Next we consider dual modeling strategies. For a given strategy frame $x \in \mathsf{frames}(P)$, the dual frame $\mathsf{dual}(x)$ is obtained by reversal of all transitions. Note that the dual frame is not necessarily a frame of the same strategy space. Only if each transition in a strategy space is accompanied by its reversal, the dual of a frame is again a frame in that same space:

$$P = \mathsf{dual}(P) \quad \Leftrightarrow \quad \forall x \in \mathsf{frames}(P)\ [\mathsf{dual}(x) \in \mathsf{frames}(P)]$$

The above is a basic property of frame duality. Note that the dual frame is only one way of deriving an entire strategy with completely different organization. Another way to derive an entirely different strategy is based on the notion of complement strategy. For a given frame $x \in \mathsf{frames}(P)$ the complement frame $\mathsf{compl}(x,P)$ contains all states, and exactly those transitions from P which are absent in the original frame x.

Example 6.
In figure 3 we see two other example strategy frames f3 and f4 . In the left hand diagram, the frame f3 expresses a dual view on object-type-driven strategies, since f3 = dual(f1). In the right-hand side, the frame f4 expresses a complement view on fact-driven strategies, because f4 = compl(f2).

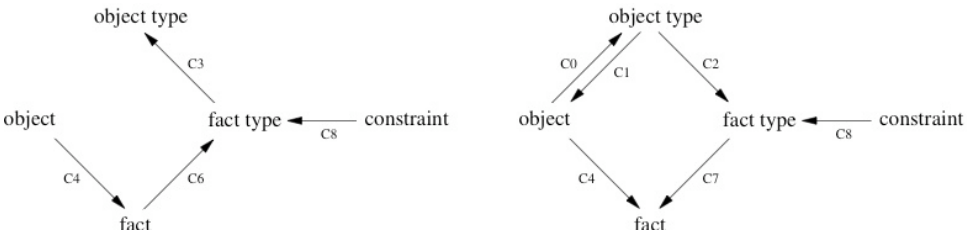

Figure 3: Example strategy frames f3 and f4

The frames f3 and f4 reflect several interesting properties of strategy frames. These will be considered in later sections. We now first express the basic property that a complement frame is again a frame of the same space:

$$\forall x \in \mathsf{frames}(P)\ [\mathsf{compl}(x,P) \in \mathsf{frames}(P)]$$

6.7 Strategy refinement

Here we discuss the refinement of basic strategy descriptions. Consider the following refined object-type-driven strategy:

Strategy description 3 *Object-type-driven strategy (refinement).*
a. Provide object types.
b. Provide fact types including constraints.
c. Provide facts including objects.

In the above description, we use a refinement of the breadth-first approach that was assumed in strategy description 1. To be able to express such refinements, the transitions in a strategy frame will be ordered.

We let $<$ be an ordering relation on transitions. For two transitions $x, y \in T$, the intention of $x < y$ is that transition x is handled prior to transition y.

Example 7.
The transition from the basic strategy description 1 to the refined strategy description 3 is obtained by the additional ordering requirement c9 $<$ c7 in frame f1.

Next we consider the following refined fact-driven strategy:

Strategy description 4 *Fact-driven strategy (refinement).*
a. Provide facts including objects.
b. Provide fact types including object types.
c. Provide constraints.

In terms of the ordering of transitions, the above refinement is expressed as follows.

Example 8.
The transition from the basic strategy description 2 to the refined strategy description 4 is obtained by the additional ordering requirements c5 < c6 and c3 < c9 in frame f2.

Besides the < constraint, we also have a stronger temporal constraint. This stronger constraint expresses that one transition must be handled immediately after another transition. Note that more temporal constraints may be embedded within our framework, for example notions occurring in workflow specifications. At this moment, these constraints are not needed for our purposes, though.

7 Implementing goals and strategies in a concrete workflow language

In this section, we show how the framework presented thus far has been used in the implementation of a reference method for requirements modelling as taught in the 2nd year Requirements Engineering course of the BSc Information Science curriculum at Radboud University Nijmegen. Please note that the method as such is not subject to discussion in this paper, just the way of describing it. This section is based on work by Jeroen Roelofs (Roelofs, 2007). The original work focused strictly on strategy description; in this paper, some examples of related goal specification are added. The strategy description was implemented as a simple but effective web-based hypertext document that allows "clicking your way through various layers and sub-strategies" in the model (see below).

7.1 Case study and example: requirements modeling course method

The main goal behind the modelling of strategies of the case method was to provide a semi-formal, clear structuring and representation thereof that was *usable for reference purposes*. This means that the rule-based nature of the framework was played down, in favour of a clear and usable representation. A crucial step (and a deviation of the initial framework) was taken by replacing the plain directed graphs (section 6) by workflow-style models in the formal

YAWL language (Yet Another Workflow Language: van der Aalst and ter Hofstede, 2005). For a further discussion of this adaptation of the framework, see section 7.3. Below we show the basic concepts of YAWL (graphically expressed), which were quite sufficient for our purposes. We trust the reader will require no further explanation.

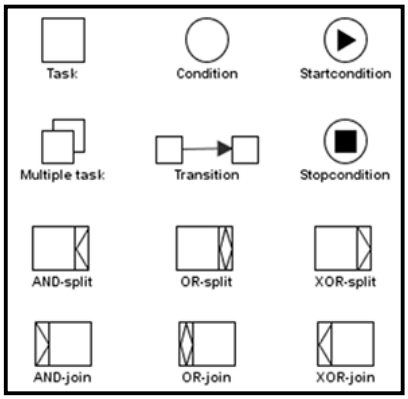

Figure 4: basic graphic concepts of YAWL

The main (top) context of the method is depicted in the following schema:

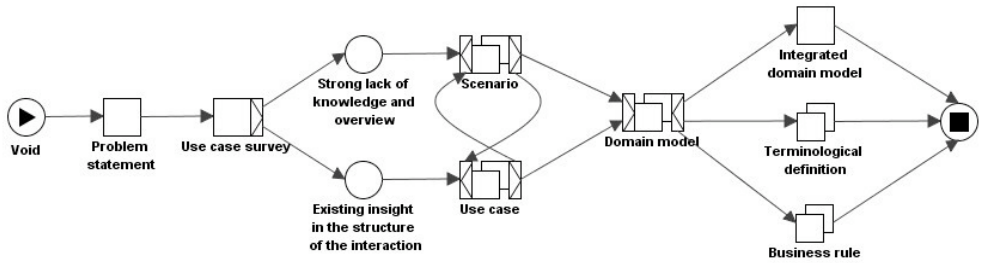

Figure 5: The top strategy context "Create a Requirements Model"

Note that in Figure 5, the square-based YAWL symbols correspond to the QoMo strategy framework in that they represent states (goals/situations). The actual strategies match the arrows between states: the actions to be taken to effectuate the transitions between states. In other words, the diagram is a very concise way of representing a strategy *context*. A useful operational addition to the framework is the use of conditions (circles) for choosing a goal (in going from "Use Case Survey" to either "Scenario" or "Use Case"): this was explicitly part of the existing method and possible in YAWL, and therefore gratefully taken aboard.

All arrows in the diagram have been labelled with *activity names* (which are another addition to the framework). Underlying the activities, there are strategies, which in turn consist of one or more *steps* (another addition). The complete strategy description of the activity "create requirements model" which is graphically captured by the top context (fig. 5) is the following:

1. **Create problem statement**
2. **Create use case survey**
3. **Create use case based on use case survey AND create scenario based on use case**
3. **Create scenario based on use case survey AND create use case based on scenario**
4. **Create domain model based on use case**
4. **Create domain model based on scenario**
5. **Create terminological definition**
6. **Create business rule**
7. **Create integrated domain model**

All steps listed are represented in boldface, which indicates they have underlying *composed* strategies (which implies that each step is linked to a further activity which is in turn linked to an underlying strategy). Concretely, this means that in the hypertext version of the description, all steps are clickable and reveal a new strategy context for each deeper activity. For example, if "**Create domain model based on use case"** is clicked, a new (rather smaller) context diagram in YAWL is shown, with further refinement of what steps to take (strategy description). We will get back to this particular strategy, but before we do this, some explanation is in order concerning the irregular numbering of steps above. The occasional repetition of numbers (3. 3. and 4. 4.) serves to match the textual description with the YAWL diagram: the XOR split and AND-join in figure 5. In addition, the two possible combinations of steps before the AND-join needed to be combined using an "AND" operator, but note that the activities linked by AND are separately clickable.

Let us now return to the "**Create domain model based on use case**" strategy. It concerns the creation of a "Domain Model" (ORM) based on a "Use Case", which (roughly in line with examples from section 6) boils down to a basic description of steps in making an ORM diagram based on the interaction between user and system that is described stepwise in the use case (please note the participants in the course are familiar with ORM modelling and therefore need only a sketchy reference process description). The related strategy context is a fragment of the one in the top context:

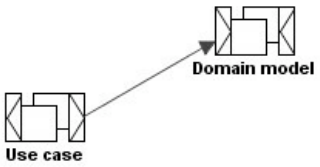

Figure 6: another strategy context –"create domain model based on use case"

Apart from this context, the underlying strategy is shown:

1. **Identify relevant type concepts in use case**
2. **Create fact types**
3. Create example population
   - Make sure the example population is consistent with the related scenario(s)
4. Make constraints complete

Steps one and two, represented in boldface, by way of more activities refer to more compositional strategies, so they are clickable and each have an underlying strategy. Activities 3. and 4. are represented differently, respectively signifying a *guided strategy* (underlined and with additional bulleted remark) and an *ad hoc strategy* (normal representation). A guided strategy is a strategy of which a description of some sort is available that helps execute it. In the example, this guidance is quite minimal: simply the advice to "Make sure the example population is consistent with the related scenario(s)". In view of our general framework, this guidance could have been anything, e.g. a complex process description or even an instruction video, but crucially it would not be part of the compositional structure. In context of our case method, we found that a few bulleted remarks did nicely.

There still is the ad hoc strategy linked to the activity name "Make constraints complete" (step 4.). It simply leaves the execution of the activity entirely up to the executor. As explained in section 6, it is an "empty strategy" –which is by no means a useless concept because it entails an explicit decision to allow/force the executing actor to make up her own mind about the way they achieve the (sub)goal.

In addition to the strategy context diagrams and the textual strategy descriptions, the hypertext description provided a conceptual diagram (in ORM) for each strategy, giving additional and crucial insights in concepts mentioned in the strategy and relations between them. The ORM diagram complementing the "create domain model based on use case" strategy is given in figure 7[3]. In

[3] The ORM diagrams in this paper were produced by means of the NORMA case tool developed by Terry Halpin and his co-workers at Neumont University: http://sourceforge.net/projects/orm.

context of the case, the inclusion of such a diagram had the immediate purpose of clarifying and elaborating on the main concepts used in the strategy description. In a wider context, and more in line with the more ambitious goals of the general strategy framework, the ORM diagram provides an excellent basis for the creation of formal rules capturing creation goals. We will discuss an extension to that wider context in the next section.

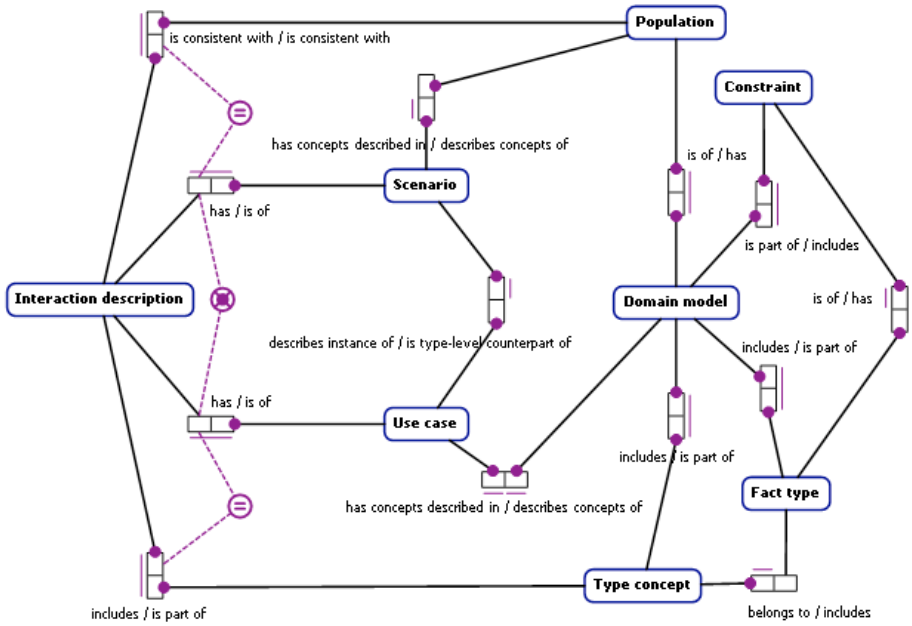

Figure 7: ORM diagram complementing the strategy description

7.2 Goal and procedure rules added to the case

The case strategy description as worked out in detail by Roelofs (2007) stops at providing a workable, well-structured description of the interlinked strategies and concepts of a specific method. Though has been found useful in education, the main aim of creating the description was to test the QoMo strategy framework. However, it could in principle also be a basis for further reaching tool design involving intelligent, rule-based support combining classical model checking and dynamic workflow-like guidance. In order to achieve this, indeed we would need to formalize the goals and process rules of the strategy descriptions to get rules of the kind suggested in (van Bommel et al., 2006) and in section 6.4 of this paper. We will go as far as giving semi-formal verbalizations of the rules.

Fortunately, such rules (closely related to FOL descriptions) are already partly available even in the case example: they can be derived from, or at least based on, the ORM diagrams complementing the strategy descriptions, and the YAWL diagrams that represent the strategy contexts. For example, fig. 6 corresponds to a (minimal) strategy frame as introduced in section 6.3. The corresponding goals is:

G1 There is at least one Domain Model

This is an instance-level goal. Next, we define the one situation that is relevant to the example strategy "create domain model based on use case":

S1 There is at least one Use Case

So we assume that one or more Use Cases have already been identified (presumably as a goal of another strategy) and that these are used as input for the strategy "create domain model based on use case". We now can weave a rule-based definition combining G1, S1, and various C-rules that correspond to the transitions captured in the strategy description. The key rules raising demands that correspond to steps in the strategy are represented in boldface.

S1 There is at least one Use Case (*situation*)
C1 SHOULD LEAD TO
G1 There is at least one Domain Model (*main goal*)
G1.1 Each Use case has concepts described in exactly one Domain model.
G1.2 Each Domain model describes concepts in exactly one Use case.
C2 SHOULD LEAD TO
G2.1 It is possible that more than one Type concept is part of the same Domain model and that more than one Domain model includes the same Type concept.
G2.2 Each Type concept, Domain model combination occurs at most once in the population of Type concept is part of Domain model.
G2.3 Each Type concept is part of some Domain model.
G2.4 Each Domain model includes some Type concept.
G2.5 Each Type concept that is part of an Interaction description of a Use case that has its concepts described by a Domain model should also be part of that Domain model (*goal underlying step 1*).[4]
C3 SHOULD LEAD TO
G3.1 It is possible that more than one Domain model includes the same Fact type and that more than one Fact type is part of the same Domain model.
G3.2 Each Fact type, Domain model combination occurs at most once in the population of Domain model includes Fact type.
G3.3 Each Domain model includes some Fact type.
G3.4 Each Fact type is part of some Domain model.
G3.5 Each Fact type that is part of a Domain model should include one or more Type concepts that are part of that same Domain model (*goal underlying step 2*).
C4 SHOULD LEAD TO
G4 Each Fact type of a Domain Model is populated by one or more Facts of the

[4] In expressing this complex rule, we use a controlled language called Object Role Calculus: see (Hoppenbrouwers et al., 2005c)

Population of that Domain Model.[5] **(*goal underlying step 3*)**
C4 SHOULD LEAD TO
G5.1 Each Scenario describes concepts of exactly one Population.
G5.2 Each Population has concepts described in some Scenario.
G5.3 It is possible that the same Population has concepts described in more than one Scenario.
G5.4 **Each Fact that is part of a Population which describes concepts of a Scenario should include at least one Instance concept that is included in that Scenario. (*goal underlying the note with step 3)***
C5 SHOULD LEAD TO
G6.1 **Each Fact type has some Constraint. (*goal underlying step 4)***
G6.2 Each Constraint is of exactly one Fact type.
G6.3 It is possible that the same Fact type has more than one Constraint.

Note that further restrictions could be imposed on G6.1, demanding explicitly that the constraints applying to a fact type should correspond to the population related to that fact type, and so on. This constraint is left out because it is also missing in the informal strategy description (step 4).

So far, our definition does not include temporal ordering. The following orderings are applied in the C-rules:

C1 *no restriction*

This reflects the achievement of the main goal, which lies outside the temporal scope of the strategy realizing it. For the rest, rather unspectacularly:

C2 occurs before C3
C3 occurs before C4
C4 occurs before C5

For a somewhat more interesting example of temporal factors, consider the XOR-split and AND-join in fig. 5. (splitting at "use case survey" and joining at "domain model"). Obviously, such split-join constructions involve ordering of transitions:

C1 occurs before C2
C1 occurs before C3
C2 occurs before C3 (under condition Y) XOR
    C3 occurs before C2 (under condition Z)
C2 AND C3 occur before C6

These expressions of rules covering YAWL semantics are rough indications; a technical matching with actual YAWL concepts should in fact be performed, but this is outside the current scope.

[5] Rules G4 and G5.4 refer "Facts" and "Instance concepts", which are not included in figure 7 but in the ORM diagram (not presented in this paper) supporting a different strategy, namely "create domain model based on scenario". In the implementation, populations are defined as included in a domain model.

Finally, note that in the implementation, fulfillment of the main goal, "create domain model from use case", is achieved even if the domain model is not finished. However, the unfinished status of the domain model would lead to a number of "ToDo" items. This emphasizes that the strategy is a *initial creation* strategy (bringing some item into existence), which next entails the possibility that a number of further steps have to be taken iteratively (triggered by validity and completeness checks based on, for example, G-rules), hence not necessarily in a foreseeable order.

7.3 Findings resulting from the implementation

The implementation led to the construction of a specific meta-model reflecting the key concepts used in that implementation (figure 8). We will finish this section by presenting the most interesting findings in the implementation with respect to the generic framework, at the hand of fig. 8.

Sources and products
The specific flavour of the implementation led to the introduction of the concepts *source*, *product*, *intermediate product*, *raw material*, and *void*. They were needed to operationalize the only goal/strategy category explicitly used in the implementation: creation goals. Situations (which are state descriptions) took the shape of concrete entities (documents) that typically followed each other up in a straightforward order: void or raw material input leading to products, possibly after first leading to intermediary products. Clearly, these concepts classified the items created; such classification emerged as helpful from the discussions that were part of the implementation process.

Use of YAWL concepts
YAWL concepts (and their graphical representations) were introduced to capture strategy context, while a simple textual description format was used to capture the stepwise strategy description. The YAWL concepts were very helpful in creating easy-to-read context descriptions. In addition, they helped in operationalizing the concepts required to capture the workflow-like transitions between states (i.e. between creation situations/goals). Whether YAWL diagrams would be equally useful in describing contexts for other types of goal (for example, validation goals or argumentation goals) remains to be explored.

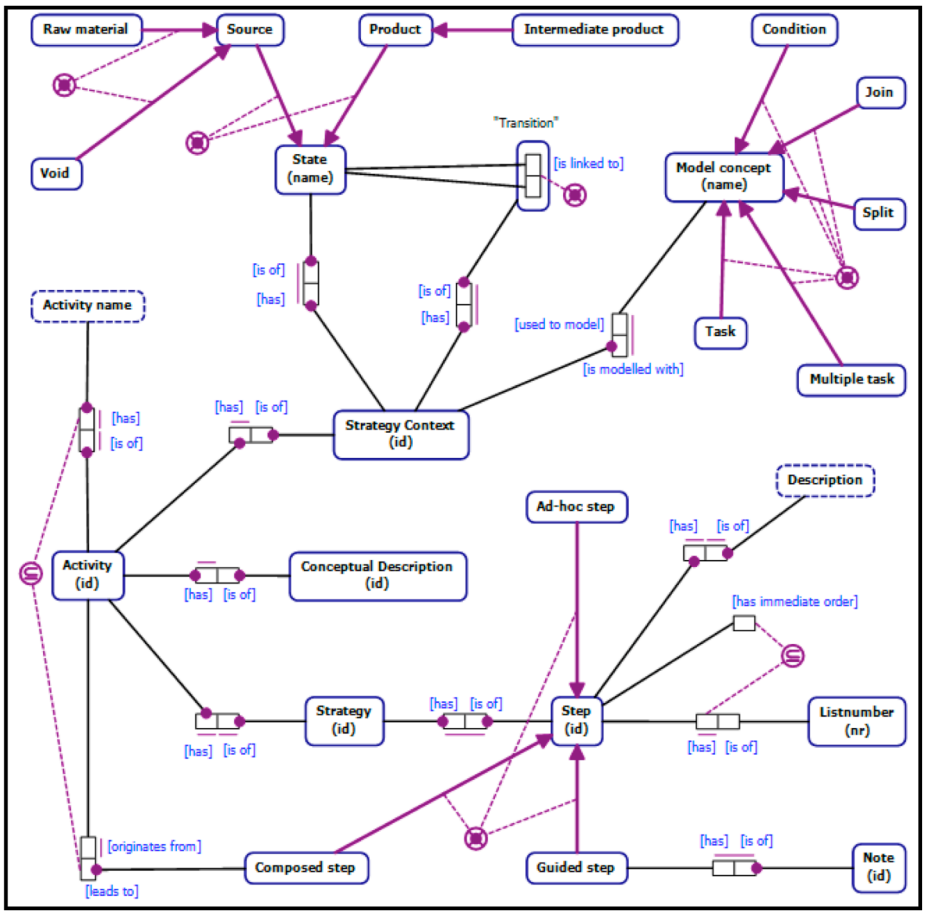

Figure 8: meta-model derived from strategy description case

In addition, YAWL concepts can be used as a basis for formal rule definition capturing the recommended order of steps. The formal underpinnings of YAWL would be extra helpful in case of automated (rule-based) implementation of the strategies, which was still lacking in the prototype (for more on this, see section 8: the "modelling agenda generator").

Activity descriptions, names, and steps

A simple but crucial refinement needed to operationalize the general framework was the introduction of the "activity" and "activity name" concepts. These allowed for the successful implementation of the recursive linking of *activities* to *strategy contexts* to *strategies* to *strategy steps* to further *(sub)activities*, and so on. We expect this amendment to be useful at the generic level, and henceforth we will include it in the main framework.

"Immediate" concept not used
The "immediate" concept was in principle included in the case study implementation but in the end was not used. We still believe it may be required in some strategy descriptions. The ordering in the creation strategies in the case is basic step-by-step. In more complex, dynamic setups, the availability of both immediate and non-immediate ordering still seems useful. However, admittedly the actual usefulness of the "immediate / non-immediate" distinction still awaits practical proof.

7.4 Lessons learned from the case study

Apart from he conceptual findings discussed in the previous section, some other lessons were learned though the case study:

- Syntax-like rules can be successfully applied beyond actual modelling language syntax (which amounts to classic model-checking based on grammar goals) into the realm of more generic "creation goals" which may concern various sorts of artefacts within a method.
- The case has shed some light on the fundamental distinction between creation and iteration in dealing with creation goals. While iteration is essentially unpredictable and thus can only receive some ordering (if any at all) through rule-based calculations based on rules and state descriptions, for initial creation people do very much like a plain, useful stepwise description of "what to do": a reference process. Only after initial creation, the far less obvious iteration stage is entered. Also, if a robust rule-based mechanism for guiding method steps is in place, participants may choose to ignore the recommendations of the reference process. This can be compared by the workings of a navigation computer that recalculates a route if a wrong turn has been made.

8 Conclusions and further research

This chapter set out to present a plausible link between the SEQUAL approach to model *product* quality and our emerging QoMo approach to *process* quality in modeling, and to provide basic concepts and strategies to describe processes aiming for achievement of QoMo goals. We did not aim to, nor did, present a full-fledged framework for describing and analyzing modeling processes, but a basic set of concepts underlying the design of a framework for capturing and analyzing 2nd order information systems was put forward.

We started out describing the outline of the QoMo framework, based on knowledge state transitions, and a goal structure for activities-for-modeling. Such goals were then directly linked to the SEQUAL framework's main concepts for expressing aspects of model items and its various notions of quality, based on model items. This resulted in an abstract but reasonably comprehensive set of main modeling process goal types, rooted in a semiotic view of modeling. We then presented a case implementation of how such goals can be linked to a rule-based way of describing strategies for modeling, involving refinements of the framework. We added concrete examples of rules describing goals and strategies, based on the case implementation.

These process descriptions hinge on strategy descriptions. Such strategies may be used descriptively, for studying/analyzing real instances of processes, as well as prescriptively, for the guiding of modeling processes. Descriptive utility of the preliminary framework is crucial for the quality/evaluation angle on processes-for-modeling. Study and control of a process requires concrete concepts describing what happens in it, after which more abstract process analysis (efficiency, cost/benefit, levels of risk and control) may then follow. Means for such an analysis were not discussed in this paper: this most certainly amounts to future work.

Besides continuing development and operationalization of the QoMo strategy and goal framework for quality modeling by applying it to new and more complex cases, we need to push forward now to implementations that actively support our rule-based approach. An initial implementation, using Prolog and a standard SQL database, is in fact available, but has not been sufficiently tested and documented yet to report on here. This "modeling agenda generator" dynamically generates ToDo lists (with ordered ToDo items if C-rules apply) based on the model states as recorded in the repository. We will finish and expand this prototype, testing it not only in a technical sense but also its usability as a system for supporting real specification and modeling processes. In the longer term, we hope to deploy similar automated devices in CASE-tool like environments that go beyond the mere model or rule editors available today, and introduce advanced process-oriented support and guidance to modelers as required in view of their preferences, needs, experience, competencies, and goals.

9 References

Van der Aalst, W. And ter Hostede, A. (2005): YAWL: Yet Another Workflow Language. *Information systems, 30*(4), 245-275.

Bommel, P. van, S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper, and Th.P. van der Weide (2006): Exploring Modeling Strategies in a Meta-modeling Context. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful In-*

ternet Systems 2006: OTM 2006 Workshops, volume 4278 of *Lecture Notes in Computer Science*, pages 1128-1137, Berlin, Germany, EU, October/November 2006. Springer.

Chrissis, M.B., M. Konrad, and S. Shrum (2006): *CMMI: Guidelines for Process Integration and Product Improvement*, Second Edition. Addison-Wesley.

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

Hoppenbrouwers, S.J.B.A., H.A. (Erik) Proper, and Th.P. van der Weide (2005a): A Fundamental View on the Process of Conceptual Modeling. In *Conceptual Modeling - ER 2005 - 24 International Conference on Conceptual Modeling*, volume 3716 of *Lecture Notes in Computer Science*, pages 128-143, June 2005.

Hoppenbrouwers, S.J.B.A., H.A. (Erik) Proper, and Th.P. van der Weide (2005b). Towards explicit strategies for modeling. In T.A. Halpin, K. Siau, and J. Krogstie, editors, *Proceedings of the Workshop on Evaluating Modeling Methods for Systems Analysis and Design* (EMMSAD'05), held in conjunction with the 17th Conference on Advanced Information Systems 2005 (CAiSE 2005), pages 485-492, Porto, Portugal, EU, 2005. FEUP, Porto, Portugal, EU.

Hoppenbrouwers, S.J.B.A., H.A. (Erik) Proper, and Th.P. van der Weide (2005c). Fact Calculus: Using ORM and Lisa–D to Reason About Domains. In R. Meersman, Z. Tari, and P. Herrero, 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, October/November 2005: Springer–Verlag.

Kensche, D., C. Quix, M.A. Chatti, and M. Jarke (2005): GeRoMe: A Generic Role Based Metamodel for Model Management. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE – OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005*, Proceedings, Part II, Agia Napa, Cyprus, EU, volume 3761 of Lecture Notes in Computer Science, pages 1206–1224. Springer–Verlag, October/November 2005.

Krogstie, J. (2002). A Semiotic Approach to Quality in Requirements Specifications. In L. Kecheng, R.J. Clarke, P.B. Andersen, R.K. Stamper, and E.-S. Abou-Zeid, editors, *Proceedings of the IFIP TC8 / WG8.1 Working Conference on Organizational Semiotics: Evolving a Science of Information Systems*, pages 231-250, Deventer, The Netherlands, EU, 2002. Kluwer.

Krogstie, J., and Jorgensen H.D (2002): Quality of Interactive Models. In M. Genero, Grandi. F., W.-J. van den Heuvel, J. Krogstie, K. Lyytinen, H.C. Mayr, J. Nelson, A. Olivé, M. Piattine, G. Poels, J. Roddick, K. Siau, M. Yoshikawa, and E.S.K. Yu, editors, *21st International Conference on Conceptual Modeling (ER 2002)*, volume 2503 of *Lecture Notes in Computer Science*, pages 351-363, Berlin, Germany, EU, 2002. Springer.

Krogstie, J., G. Sindre, and H. Jorgensen (2006). Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, 15:91-102, 2006.

Mirbel, I., and J. Ralyté (2006): Situational Method Engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, 11:58-78, 2006.

Moody, D.L., (2006): Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data and Knowledge Engineering*, (55):243-276, 2006.

Nelson, H.J., and Monarchi, D.E. (2007): "Ensuring the Quality of Conceptual Representations". In: *Software Quality Journal*, 15:213-233. Springer

Object Management Group OMG (2001): *Common Warehouse Metamodel (CWM) metamodel, version 1.0*, Februari 2001.

Poels, G., Nelson, J., Genero, M., and Piattini, M. (2003): "Quality in Conceptual Modeling – New Research Directions". In: A. Olivé (Eds.): *ER 2003* Ws, LNCS 2784, pp. 243-250. Springer.

Ralyté, J., Brinkkemper, S., and Henderson-Sellers, B., eds., (2007): *Situational Method Engineering: Fundamentals and Experiences*. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland. Series: IFIP International Federation for Information Processing , Vol. 244.

Roelofs, J. (2007): *Specificatie van Strategieën voor Requirement Engineering*. Master's thesis, Radboud University Nijmegen; in Dutch.

Saeki, M., (1995): Object–Oriented Meta Modelling. In M.P. Papazoglou, editor, *Proceedings of the OOER'95, 14th International Object–Oriented and Entity–Relationship Model ling Conference*, Gold Coast, Queensland, Australia, volume 1021 of Lecture Notes in Computer Science, pages 250–259, Berlin, Germany, EU, December 1995. Springer.

Sindre, G. and J. Krogstie (1995): "Process heuristics to achieve requirements specification of feasible quality". In: *Second International Workshop on Requirements Engineering: Foundations for Software Quality* (REFSQ'95), Jyväskylä, Finland.