Exploring Modelling Strategies in a Meta-modelling Context

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

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

Abstract. We are concerned with a core aspect of the *processes* of obtaining conceptual models. We view such processes as information gathering dialogues, in which strategies may be followed (possibly, imposed) in order to achieve certain modelling goals. Many goals and strategies for modelling can be distinguished, but the current discussion concerns *meta-model driven* strategies, aiming to fulfil modelling goals or obligations that are the direct result of meta-model choices (i.e. the chosen modelling language). We provide a rule-based conceptual framework for capturing strategies for modelling, and give examples based on a simplified version of the Object Role Modelling (ORM) meta-model. We discuss strategy rules directly related to the meta-model, and additional procedural rules. We indicate how the strategies may be used to dynamically set a modelling agenda. Finally, we describe a generic conceptual structure for a *strategy catalog*.

1 Introduction

This paper reports on an intermediary result in an ongoing effort to analyse and eventually improve processes for conceptual modelling (domain modelling) in information and knowledge system development. In [6], we have argued that such modelling processes can be viewed as specialized information gathering dialogues in which knowledge of a domain expert is gradually made explicit and refined until the resulting representations conform to relevant aspects of the domain expert's conception and also to the demands on well-formedness posed by the modelling language (usually a formal or semi-formal one).

As argued at length in [8], not much research has been conducted concerning stages in, and aspects of, ways of working used in modelling efforts. The *how* behind the activity of creating models is still mostly art rather than science. A notable exception is work by Batra et al. for example [1, 3]. Though their work includes rules very much like the ones presented in section 3 of this paper (but for ER instead of ORM), they are not concerned with the rules as such, but in applying them in a knowledge-based system for conceptual modelling support. This is also a long-term goal for us, but we approach it from "within" the modelling process, whereas Batra et al. seem to rather "work their way in

R. Meersman, Z. Tari, P. Herrero et al. (Eds.): OTM Workshops 2006, LNCS 4278, pp. 1128–1137, 2006.

from the outside". Thus, there is a small overlap between their and our own work, but there is also a difference in approach and focus.

There are various reasons for improving our understanding of the details of the modelling process, and how to improve it. Some aspects of model quality (most in particular, validation) can be better achieved through a good modelling process rather than by just imposing requirements on the end product. Valid models are only attainable by viewing a model in context, and therefore in relation to the actors who have created and agreed with some model [7]. The process of modelling thus is a crucial link between the end product and its context. Another important reason for studying which course of action leads to good models concerns human resources. Generally, few experts are available who are capable of, or willing to, perform high-quality modelling. Anything that helps guide modelling behavior and support the process would be helpful. Eventually, we therefore hope to contribute to the creation of intelligent guiding systems that aid, and where required control, the modelling process. First, however, detailed study of (aspects of) the modelling process is required. One such aspect is dealt with in this paper.

2 Modelling as a Structured Dialogue

As briefly stated in the introduction, we view modelling processes as dialogues: the bringing forth of series of statements (including questions, propositions, etc.) by the participants in a modelling activity, together making up the step-by-step creation of a model (including rejection or revision of previous statements) [7, 4]. The main roles played by participants distinguished by us are:

Domain Expert (DE) – provides valid information about the domain modelled, and is capable of deciding which aspects of the domain are relevant to the model and which are not.

Model Builder (MB) – is responsible for choosing the modelling meta-concepts appropriate for the modelling task at hand; i.e. the concepts fit for correct modelling conform the chosen modelling paradigm.

Modelling Mediator (MM) – helps the DE to phrase and interpret statements concerning the domain that can be understood or are produced by the MB. The MM also helps to translate questions the MB might have for the DE, and vice-versa.

Let us consider in some detail six classes of modelling goals we have distinguished so far for modelling processes:

Over-all goals – The most general goals, the central ones of which are the *completeness*, and *consistentness* goals. Though there surely exist strategies that help achieve these goals, they are relatively straightforward.

Validation goals – These goals concern the level of knowledge sharing involved in the information exchange underlying a modelling effort [6].

Argumentation goals – Such goals make for the description of (some aspect of) the argumentation behind a particular modelling choice; in an increasing number of model types, such argumentation is included.

Meta-model oriented goals – Goals imposed by the modelling language, and the correctness of the statements made in it. If a modelling language is strictly defined (syntactically, semantically, or both), these goals are at the core of the strategies that lead to well-formed models in that language.

Abstraction goals – These goals are set in order to assure optimal choices in the level of abstraction applied in modelling. Though such choices may be partly engrained in the meta-model, choosing a good level of abstraction is crucial for good modelling.

Interpretation goals – Goals raising the question whether the meaning of concepts or statements used in the model but not further defined or formalized are sufficiently clear for all participants.

Currently we focus on the fourth type: meta-model oriented modelling goals.

Modelling languages (especially formal ones) impose requirements not only on model structure but as a consequence also on modelling processes. Typically, if a concept is introduced into the model, often some clearly defined co-concepts are also required, as well as particular relations between the concepts involved. For example, in the ORM modelling paradigm the central modelling concepts are Objects (*Game,Student*) and Roles (*plays*), that together constitute Fact Types (*Students play Games*). Thus, Objects and Roles are typical co-concepts.

Since we regard modelling as a process there is also a temporal aspect involved: certain concepts (of some type), or combinations thereof, are required or allowed to be added to the model in view of previously introduced concepts. The metaphor we suggest is that of slot filling. Each meta-concept has its place in the general slot structure for a (part of) the model under construction. The meta-concepts provide pull for information about the model conform the meta-model. In a way, they represent potential, abstract questions asked by the MB to the DE (possibly with the MM as intermediary) in order to achieve the meta-model oriented goals. Addition of a new concept may thus trigger a sequence of consecutive modelling actions. In this way, the meta-model (or modelling language) for a considerable part *drives the information gathering and creation process*, by setting detailed goals for the model's syntax and structure. By imposing constraints on the meta-structure (let us call them *meta-constraints*) it is possible to structure the questions generated by the meta-model over time. The meta-model, combined with the (meta-)constraints, constitutes a rule system that dynamically restricts the modelling possibilities as modelling progresses.

For example, fact types in ORM are usually complemented by constraints that are imposed on the population of these fact types. A number of different constraint types can be used, but the main ones are similar to existential and universal quantifiers in predicate logic. Thus, the constraints restrict allowed populations for the predicates. Concretely, if a fact type (*Students play Games*) has been added to the model, the question is raised what constraints hold for that fact type; a request along such lines is added to a *modelling agenda*. It is

our purpose to *make explicit such a modelling agenda and the rules that generate it*. With respect to our constraints example, there are some choices to be made with respect to the precise meta-constraints on constraints:

- Are constraints (possibly of a specific sub-type) obligatory or not?
- Do certain constraints need to be added to a predicate directly after it is introduced, or can this be postponed?
- Do certain constraints have to be verifiably derived from a known (i.e. modelled) population, or not?

3 Strategies for Modelling

As discussed, a meta-model Way of Modelling has an immediate influence on the way of working used (or *to be* used) by some participant(s) in a modelling process. In addition, variations in meta-constraints can be used to compose procedural rules for different modelling strategies. In ORM practice, for example, the following main over-all strategies have been observed to co-exist (in part some styles are reflected in ORM-related textbooks, e.g. [9, 2, 5, not intended to be a comprehensive overview]:

```
"ObjectType Driven" Strategy
a. Provide ObjectTypes
b. Provide complete FactTypes (adding Roles to the ObjectTypes)
c. Possibly, provide Facts (including Objects) illustrating the FactTypes
d. Provide Constraints (possibly based on the collection of Facts)

"Fact Driven" Strategy
a. Provide Facts
b. Create FactTypes
    (c. Provide ObjectTypes)
    (d. Provide Roles)
e. Provide constraints (strictly based on the collection of facts)
```

Trivial though the differences between these over-all strategies may seem to the casual observer, they have given rise to considerable debate among experienced modelers. In addition, novice ORM modelers often feel uncertain about ways to proceed in modelling: where to start? Where to go from there? What to aim for? Below, we will provide a more detailed analysis of over-all modelling strategies and rephrase them as *meta-model based modelling (sub)strategies*. Though all examples are ORM related, we indeed believe our approach to be useful for many other modelling languages, both formal and semi-formal. We intend to explore strategies for various modelling languages (and parallels and differences between them) before long.

So far, we have discussed procedural choices and steps in modelling under the assumption that the meta-model involved is stable. However, it is not unthinkable that meta-concepts in a modelling language are progressively switched on, for example in case a static model is created first, after which temporal concepts are added [10]. This would result in a modelling procedure that spans several types of model, and allows for evolution of one model type to another. We have already worked out conceptual details of such an approach. We refrain from elaborate discussion of this direction in this paper.

Meta-model choices that have been set as part of the ORM paradigm can themselves be represented as a conceptual schema. In this paper, for illustration purposes only, we use a simplified version of the ORM meta-model, presented below in verbalized ORM form (as opposed to an ORM schema). For the additional procedural rules, we added some straightforward but admittedly ad hoc temporal terminology. Please note that this ORM meta-model is used for illustration purposes only, and in no way is intended to reflect the "ultimate ORM meta-model".

Tightening our definitions, meta-model related modelling strategies are linked to three sets of modelling choices to be made:

Meta-model choices: The entire set of potential "concept slots" that are to be filled; the "maximum meta-model".

Modeling procedure choices: The meta-model parameters (see above) can be set and reset both *before* and *during* the modelling process. Keeping to the slot filling metaphor, the parameters result in decisions whether or not particular slots related to the maximum meta-model actually need to be filled. In addition, rules are added that render an *order* in which slots are to be filled.

Rationale choices: In some cases, items in the model can be introduced for various different reasons, and based on various different sources.

For example (based on ORM):

- facts obligatory / facts optional [*meta-model choice*]
- constraints optional / constraints obligatory [*meta-model choice*]
- objects first / fact types first / roles first [*procedure choice*]
- constraints immediately after fact types
- facts first (driving elicitation) / facts later (completing elicitation) [*procedure choice combined with rationale choice*]
- facts for concept validation / facts for constraint derivation and validation [*rationale choice*]
- facts as example (made up) / facts as concrete link with domain [*rationale choice*]

Such choices once made, can be crystallized into *rules for modelling*.

Below we specify two related sets of rules that together provide a reasonably complete basis for meta-model based modelling goals and strategies, for the Strict Fact-based approach to ORM modelling. A generic advantage of declarative rules is that they can restrict the course of a modelling process as much as is needed, without the obligation to comprehensively specify all possible modelling sequences. Admittedly, the distinction between goals and strategies is somewhat blurred here. We choose fact-based approach to ORM because it is a relatively restrictive flavour of modelling which allows us to nicely demonstrate the operation of our restrictive rules. In the example, the first set of rules ("M-rules") simply reflect the ORM meta model. The rules are expressed here in a semi-natural, controlled language that is standard for ORM verbalizations and that can easily be formalized. The second set of rules also includes some terms for expressing procedural aspects; this is not standard ORM, but any sort of straightforward temporal modelling should be able to deal with them. In addition, we use the term "Action", meaning the objectified combination of a meta-model item (e.g. FactType, Object) and a mutation of the model (e.g. add, modify).

```
META-MODEL FOR STRICT FACT-BASED MODELLING
M1  Each ObjectType is populated by one or more Objects
M2  Each Object populates exactly one ObjectTypes
M3  Each ObjectType has exactly one Type
M4  Each Type is of one or more ObjectTypes
M5  Each ObjectType plays one or more Roles
M6  Each Role is played by exactly one ObjectType
M7  Each Object is of one or more Facts
M8  Each Fact has one or more Objects
M9  Each Fact has one or more Roles
M10 Each Role is of one or more Facts
M11 Each Role is of exactly one FactType
M12 Each FactType has one or more Roles
M13 Each Role has one or more constraints
M14 Each Constraint applies one or more Roles
M15 Each Constraint has exactly one ConstraintType
M16 Each ConstraintType is of zero or more Constraints

TEMPORAL CONSTRAINTS FOR STRICT FACT-BASED MODELLING
P1 an Action that involves Addition of an Object is immediately followed by an Action that
    involves Addition of an ObjectType that is populated by that Object
P2 an Action that involves Addition of an Object Type takes place later than an Action that
    involves Addition of a Fact that has a Role that is played by that same ObjectType
P3 an Action that involves Addition of a Role is immediately followed by an Action that
    involves Addition of an ObjectType that plays that Role
P4 no Action that involves Addition of an Object takes place before an Action that involves
    Addition of a Fact that has that same Object
P5 no Action that involves Addition of an ObjectType takes place before an Action that
    involves Addition of an Object that populates that same ObjectType
P6 no Action that involves Addition of a Role takes place before an Action that involves
    Addition of a Fact that has that same Role
P7 an action that involves Addition of a Constraint coincides with the Addition of a
    ConstraintType that is of that Constraint
```

Not all these rules are to be enforced absolutely and immediately. Some only set an agenda, while others allow for only one possible next step in the modelling process.

The initial agenda sets a first move. As can be derived from the rules (most prominently, P2,P4,P6), the only way to start modelling in Strict fact-Based Modelling is to introduce a Fact. Thus, in such a situation there is only one modelling action on the agenda: "add Fact". Note that the meta-model as such does not prevent the model from being empty. The rules merely determine that if an initial step is taken (starting either from an empty model or from another stable model), this *has* to be the addition of a Fact.

We assume for now that the agenda is merely set on the basis of slots that are actually filled in (i.e a concrete intermediate model version). In other words: no agenda points are inferred merely one the basis of existing agenda points. Once an initial action is performed (for example, the fact "John plays tennis" has been added), then various M-rules dictate the setting of the following agenda points:

```
M8: add one or more Objects that are of the Fact ‘‘John Plays tennis’’
M9: add one or more Roles that are of the Fact ‘‘John Plays tennis’’.
```

At this point, no P-rules enforce an order. Hence, either one or more Roles or one or more Objects may be added. Suppose an Object is added: "John". In relation to the meta-model, this implies that 'The Object "John" is of Fact "John Plays Tennis" '. This action de-activates rule M8 (the minimum of one object per fact has been reached); however, note that it is still *allowed* to add Objects to the

Fact. The addition of "John" also activates another rule. This results in the following new agenda points:

```
M2: add one ObjectType that is populated by Object ‘‘John’’.
```

In addition, a P-rule now becomes active:

```
P1: immediately add one ObjectType that is populated by Object ‘‘John’’.
```

There is another P-rule that is activated: P2. This rule requires that "an action that involves addition of an ObjectType takes place later than an action that involves addition of a Fact that has a Role that is played by that same ObjectType". In other words, we need the Fact's Role even before we get to the ObjectType. So:

```
P2: immediately add one or more Roles that are of the Fact ‘‘John Plays tennis’’
P1: immediately add one ObjectType that is populated by Object ‘‘John’’.
```

In response, first a Role is added to the fact "John plays Tennis"; this is "plays". Thus, 'The Role "plays" is of Fact "John plays tennis" '. This action activates rule M6; this results in a combination between P1 and M6 that adds a request for addition of a particular ObjectType. Also, another M-rule is activated: M13. So now we have the following agenda:

```
P1: immediately add one ObjectType that is populated by Object ‘‘John’’.
M6: add one ObjectType that plays Role ‘‘plays’’.
M13: add one Constraint that applies to Role ‘‘plays’’.
```

In response, ObjectType "Person" is added: 'The ObjectType "Person" is populated by Object "John" '. Also added: 'The ObjectType "Person" plays Role "plays". Two words from the Fact have now been qualified. Only the word "tennis" is yet unqualified. Indeed, in the current setup it is not obligatory to qualify all words in a Fact, yet a rule to this effect could well be added. Let us assume that the modeler, by her own initiative, first adds "tennis" as an Object (i.e. before the required constraint is tackled). Thus we get 'The Object "Tennis" is of Fact "John Plays Tennis" '. This triggers a few new agenda point, repeating a pattern already shown:

```
P2: immediately add one or more Roles that are of the Fact ‘‘John Plays tennis’’
P1: immediately add one ObjectType that is populated by Object ‘‘Tennis’’.
```

In response, we get 'The Role "is played by" is of Fact "John plays tennis'. This again triggers M6 and M13, so we now have:

```
P1: immediately add one ObjectType that is populated by Object ‘‘Tennis’’.
M6: add one ObjectType that plays Role ‘‘is played by’’.
M14: add one Constraint that applies to Role ‘‘is played by’’.
```

The response is 'The ObjectType "Sport" is populated by Object "Tennis" '. Now two agenda points are left:

```
M14: add one Constraint that applies to Role ‘‘plays’’.
M14: add one Constraint that applies to Role ‘‘is played by’’.
```

As mentioned, in this paper we only provide an example, without claiming to present some definite meta-model or set of rules. So, to wrap things up: Response: 'Constraint with ConstraintType "SharedUniqueness" is placed on Role "plays" '; 'Constraint with ConstraintType "SharedUniqueness" is placed on Role "is played by" '. This results in a Uniqueness Constraint being placed on the combined Roles "plays" and "is played by". No agenda points are left; a "stable model" has been reached. All meta-model oriented rules are satisfied until a further addition is made. We may or may not return to the "add Fact" agenda, depending on the Over All goals (see section 2), which are not discussed in detail here. This concludes our example.

4 Strategy Catalog

Our framework emphasizes meta-model driven modelling strategies. In order to facilitate the construction of these modelling strategies and the reasoning about strategies, we propose an underlying structure of strategies called a *strategy catalog*. The different parts of a strategy catalog are used to capture components of modelling agendas and modelling dialogues. These components, such as model mutations and phasing of modelling actions are considered basic building blocks for modelling strategies.

In a strategy catalog we will find *items* which are of a a particular *item kind*. Examples of items are: John, person, John works for the personnel department, and persons work for departments. The corresponding item kinds are: entity, entity type, fact, and fact type, respectively. Items are inherited from the application domain, whereas item kinds come from the meta-model of the technique used for modelling that domain.

The basic building blocks for modelling strategies are mutations and actions. As an example, consider mutations such as addition, modification, and deletion. A modelling action then is characterized as a mutation applied to an item kind. For example: addition of an entity type, addition of an entity, addition of a fact type, modification of an entity type, and the addition of a constraint. Clearly these modelling actions deal with item kinds rather than items. However, the execution of actions does indeed affect the items, for example by the addition of entity type *person*, and the addition of entity *John*.

At this stage, the modelling actions discussed above, are randomly ordered. In some situations this will be fine, but for most more restrictive modelling procedures we require a specific order. Modelling actions may be related to each other via *ordering* and *succession*. The first is used to express that a certain action should be performed after another action, whereas the second is used to require that an action is the *immediate* successor of another action. This allows for sequential as well as parallel actions. To capture more of the properties of modelling actions, our catalog records further properties such as *urgency* and *phasing*:

Urgency. Examples are: forbidden, required, optional, allowed.
Phasing. Examples are: first, last, middle.

Urgencies are used as a refinement of *must* and *can*, while the phasing of modelling actions is used as a shorthand for certain orderings. The intention of modelling actions and their properties is that modelling strategies can be defined in a flexible manner. Strategies may then be varying from a very strict application where modelling is a predefined path, to more liberal applications where modelling is more like a pathless land. Exact concepts for procedural rules will vary; we make no decision here, we just provide examples.

The execution of modelling actions yields specific items occurring in the application domain. These items may have certain properties related to the application environment. For example, we have a *purpose* and *reality value* as follows:

Purpose. Examples are: validation, constraint derivation, type introduction.
Reality value. Examples are: real, fictive, expected.

These aspects deal with rationale choices discussed in section 3. Summarizing, the purpose and reality value of items may be used in different ways, for example:

- Certain entities are real and are used for type introduction.
- Certain facts are real and are used for constraint derivation.
- Certain facts are fictive and are used for validation.

5 Next-Generation Studies of Modelling Strategies

In this paper we have presented a concise overview of our work in the area of meta-model related modelling strategies. Using modelling agendas with an underlying strategy catalog, we arrive at the situation where reasoning about (a) the construction of modelling strategies, and (b) the behaviour of modellers during strategy execution, becomes possible. For example, if the input of the modelling process is seen as some kind of informal specification coming from a domain I, and the output is a model expressed in a more formal language L, a strategy S has the following signature: $S : I \longrightarrow 2^L$

So for some input $i \in I$ a strategy S yields the models $S(i) \subseteq L$. Of course if the result of the modelling process is required to have certain predefined characteristics, a strategy should support this. A typical example is the production of models in a particular normal form. In order to deal with such postconditions, suppose P is a property of models, which can be defined as a predicate over L. A strategy supports this property if the property holds for all possible output models: $\forall_{i \in I, x \in S(i)} [P(x)]$

If the strategy does not fully support the postcondition, it often can be guaranteed by an additional requirement Q on the informal specification, expressing the appropriate precondition: $\forall_{i \in I} \left[Q(i) \Rightarrow \forall_{x \in S(i)} [P(x)] \right]$

In order to compare two strategies S_1 and S_2, we suppose $i \in I$ is a given informal specification. On the one hand, the strategies are *orthogonal* for specification i if they can not result in the same model: $S_1(i) \cap S_2(i) = \emptyset$

On the other hand, the different strategies S_1 and S_2 are *equivalent* if they result in exactly the same models. We then have $S_1(i) = S_2(i)$. In most cases we

will be somewhere between these extremes, having overlap in $S_1(i)$ and $S_2(i)$. Note that this also allows us to study the effect of changing a given strategy. We then want to compare the strategies S_1 and $S_2 = r(S_1)$ where r is a reorganization operator.

The above sketch of the reasoning framework leaves open many parameters. Reasoning may have other purposes as well, for example a dual approach where two different input specifications are compared for the same strategy. Clearly much work is still to be done. The formalization of rationale choices in strategy rules need further attention. Deterministic strategies with $|S(i)| = 1$ may be studied. Moreover, automated simulation of non-deterministic modelling processes with $|S(i)| > 1$ allows for the characterization of different cognitive identities of system analysts.

References

1. S. Anthony, D. Batra, and R. Santhanam. The use of a knowledge–based system in conceptual data modeling. *Decision Support Systems*, 41:176–190, 2005.
2. G.P. Bakema, J.P.C. Zwart, and H. van der Lek. Fully Communication Oriented NIAM. In *Proceedings of NIAM–ISDM 2*, pages 1–35, August 1994.
3. D. Batra and S. Antony. Consulting support during conceptual database design in the presence of redundancy in requirements specifications: an empirical study. *IBM Journal of Research and Development*, 54:25–51, 2006.
4. P.J.M. Frederiks and Th.P. van der Weide. Information Modeling: the process and the required competencies of its participants. *Data & Knowledge Engineering*, 2005.
5. T.A. Halpin. *Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design*. Morgan Kaufmann, San Mateo, California, USA, 2001.
6. S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper, and Th.P. van der Weide. A Fundamental View on the Process of Conceptual Modeling. In L. Delcambre, C. Kop, H.C. Mayr, J. M ylopoulos, and O. Pastor, editors, *ER 2005 – 24 International Conference on Conceptual Modeling, Klagenfurt, Austria, EU*, volume 3716 of *Lecture Notes in Computer Science*, pages 128–143. Springer, June 2005.
7. S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper, and Th.P. van der Weide. Formal Modelling as a Grounded Conversation. In G. Goldkuhl, M. Lind, and S. Haraldson, editors, *Proceedings of the 10th International Working Conference on t he Language Action Perspective on Communication Modelling (LAP'05)*, pages 139–155. Linköpings Universitet and Hogskolan I Boras, Sweden, EU, June 2005.
8. S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper, and Th.P. van der Weide. Understanding the Requirements on Modelling Techniques. In O. Pastor and J. Falcao e Cunha, editors, *17th International Conference on Advanced Information Systems Engineering, CAiSE 2005, Porto, Portugal, EU*, volume 3520 of *Lecture Notes in Computer Science*, pages 262–276. Springer, June 2005.
9. G.M. Nijssen. *Universele Informatiekunde*. PNA Publishing, The Netherlands, EU, 1993. In Dutch.
10. H.A. (Erik) Proper, S.J.B.A. Hoppenbrouwers, and Th.P. van der Weide. A Fact–Oriented Approach to Activity Modeling. In R. Meersman, Z. Tari, and P. Herrero, editors, *OTM Workshops – OTM Confederated International Workshops and Posters, Agia Napa, Cyprus, EU*, volume 3762 of *Lecture Notes in Computer Science,* pages 666–675. Springer, October/November 2005.