Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology

Hans W. Nissen, Manfred A. Jeusfeld,
and Matthias Jarke*
Georg V. Zemanek and Harald Huber †

Abstract

Informal, teamwork-oriented methods for requirements engineering can be strengthened by conceptual modeling tools for the analysis and interrelation of viewpoints expressed by multiple stakeholders in multiple different notations. Experiences gained with the meta data management system ConceptBase show that a simple but customizable meta modeling approach, combined with advanced query and view handling capabilities from the deductive database field, can significantly improve the quality of resulting requirements documents as well as focusing and accelerating the RE process. In terms of modeling strategy, these experiences advocate a modeling approach that exploits redundancy and tolerates inconsistency to provoke the elicitation of conflicts, with the goal of better understanding and deeper agreement.

Keywords: Requirements Engineering, Conceptual Modeling, Multiple Perspective Reconciliation, Formal Business Process Analysis

Acknowledgments. Work by RWTH Aachen was supported in part by the European Commission in ESPRIT projects COMPULOG II, NATURE, and the European-Australian cooperation project ISI. Thanks are due to Martin Staudt for his contributions to this work.

1 Introduction

Background. USU is a medium-size software and consulting firm that has catered to large information systems users in central Europe for the past 18 years. In addition to general database application projects, special expertise has been developed in client-server information systems for domains such as DP controlling, transfer pricing in large organizations, and resource management.

*full address: Informatik V, RWTH Aachen, Ahornstrasse 55, 52056 Aachen, Germany,Tel.: +49-241-80-21501, Fax: +49-241-8888-321, Email: {nissen,jeusfeld,jarke}@informatik.rwth-aachen.de

†full address: USU Softwarehaus Unternehmensberatung GmbH, Spitalhof, 71693 Möglingen, Germany, Tel.: +49-7141-4867-0, Fax: +49-7141-4867-20

Nowadays, user needs change all the time. Requirements engineering (RE) and business process analysis are therefore becoming ever more crucial for the success of these projects. Over the past few years, USU has therefore invested in the development of a teamwork-centered methodology called PFR (analysis of Presence and Future Requirements) that specifically addresses the need for requirements capture from multiple information sources. The application of this methodology has become so successful in the market that RE projects have become a profitable USU product in their own right.

PFR is targeted towards the rapid and focused requirements capture in a setting that alternates between team workshops and distributed interviews, in order to avoid the well-known problem that requirements change while they are being analyzed. Highly skilled group facilitators emphasize openness and tolerance of conflicts between requirements perspectives in order to use these conflicts productively for further requirements elicitation. While they offer default notations for each of the different perspectives considered, they are also open to adapting them to specific customer situations.

The Problem. This methodology, while effective, has also shown some problems which we believe to be quite common in current RE (cf. [2, 4]): PFR creates a lot of visual, partially overlapping and conflicting requirements perspectives in a very short time. The successful evolution of an RE project depends on the rapid and accurate cross-analysis of these viewpoints for possible incompleteness, inconsistency, and conflicts of perception or opinion.

These discreparencies should initially just be captured and maintained. It is a separate issue if and when conflict recognition is followed by corresponding negotiations and viewpoint revisions, in order to deepen understanding or approach resolution. The rapid knowledge acquisition process and the large number of potential conflicts to be supervised simultaneously causes analyst overload. The problem is exacerbated by the necessity to customize notations to user habits.

It may seem an obvious solution to capture and relate the different perspectives using a CASE environment. However, most existing CASE environments are too inflexible for this kind of task. Firstly, they offer only fixed notations which cannot be customized to specific organizations or projects. Second, cross-checks between these notations, if present at all, are hardcoded. Third, they do not offer a flexible process of actually dealing with recognized conflicts, and of monitoring multiple conflicts simultaneously.

The Approach. In the past few years, *meta modeling* has been promoted as a step towards solving such problems. Meta modeling allows you to specify modeling languages in a declarative manner and to generate tools for their support, thus solving the customization problem. The meta model can also specify the relationships between modeling languages, and thus indirectly define possible relationships between the individual perspectives. To become robust against variations of notations used in different organizations or situations, you can further abstract the different modeling languages to an axiomatic meta meta model (short: M2-model) which defines the principle concepts of "what information systems are" [12]. Existing, extended, or even new modeling languages instantiate this M2-model. We go to the extreme on this scale by arguing that even the M2-model should be changeable.

To support the required co-evolution of M2-model, meta models and requirements models, they need to be jointly managed by a shared repository. Its query language can then be used to support a flexible, conflict-tolerant requirements analysis process which is defined at the meta meta level and thus to some degree independent of the specific notations used.

Of course, jointly managing multiple levels of abstraction and handling queries and updates across these levels places serious demands on the knowledge representation mechanism used in the repository. This is enabled by the meta data management system ConceptBase developed at RWTH Aachen ([7], see also Sidebar 2). The meta modeling language Telos [16], supported by ConceptBase, was designed exactly with the above goals in mind. A concept of query classes, defined on top of Telos and implemented in ConceptBase, assists the analysis process.

The Experiences. Analysts from USU have gained experiences with using ConceptBase for the representation and analysis of multiple requirements perspectives, gained through PFR, in several commercial RE projects. The perspectives and modeling languages of the PFR methodology were formally described and abstracted to a dedicated M2-model which stresses the typical RE problems encountered with USU's customers. The analysts identified about 80 different classes of conflicts and coded them as ConceptBase query classes. Their simultaneous monitoring turned out to strain the capabilities of ConceptBase but also led to better requirements documents and faster analysis processes.

The Lessons. Although the experiences reported in this paper do not constitute a full scientific validation of the approach, some generalizable lessons can be drawn. The main lesson learned is that the meta modeling approach sketched above is commercially feasible with present (albeit prototype) technology. It does contribute significantly to relieving the problems of supporting a flexible, informal teamwork-oriented RE methodology with formally based, automated analysis tools. The experiences also stress the advantages of being able to tailor meta models and integration strategies to the major problems expected.

Paper Overview. The remainder of this paper works out the above issues. First, we summarize background on meta modeling and how it is supported by the ConceptBase system. Starting from an informal description of the PFR methodology, section 3 reports USU's experiences with the process of formalizing it and abstracting it to a M2-model, developing analysis tools for this model, and applying them in customer projects. Section 4 summarizes generalizable lessons drawn from these experiences.

2 Meta Model-Based Perspective Resolution

2.1 Meta Modeling

The goal of conceptual modeling is to formally describe a part of the world to support its understanding and communication. Most conceptual modeling languages suffer from the fact that they contain a fixed framework of how to view of the world. Typically, this predefined linguistic framework is designed with a concrete ontology in mind and therefore eases the modeling of a particular aspect of the application domain. However, at the same time it limits the application of the language to exactly this aspect.

Meta modeling is the activity of constructing models about models. While requirements models are abstract representations of an existing or desired real world, meta models are abstract representations of existing or desired requirements models and their interrelationships.

As early as 1984, an M2-model was proposed for the PLEXSYS system [12] to define modeling languages used to model the schema of target information systems. This enabled the

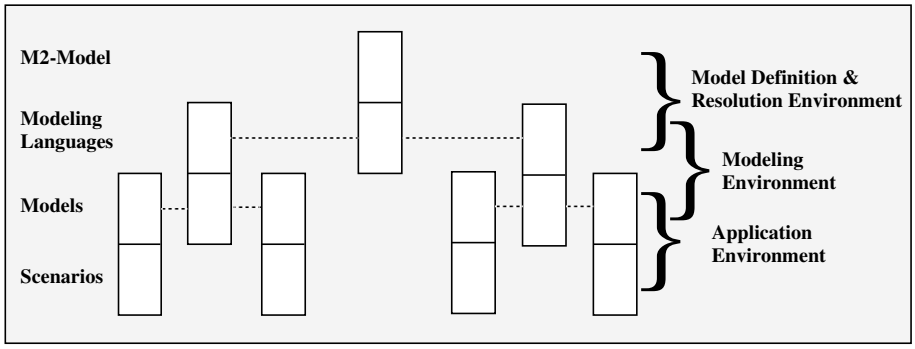

Figure 1: The ISO-IRDS architecture applied to requirements engineering

representation of models derived from different notations in a single system. Furthermore, the comparison of such heterogeneous notations became possible.

The ISO standard for *Information Resource Dictionary Systems* (IRDS) [6] extends this three-level concept by representing actual scenarios of the modeled reality at the instance level. Figure 1 provides a diagrammatic view of this four-layer architecture, with the roles of each level in RE shown on the left.

The four levels can be grouped in level pairs defining interlocking environments, as shown on the right-hand side of the figure for the case of requirements modeling : application-level systems, the actual modeling tools, and the model definition and resolution environment which manages the interaction among modeling tools.

The interlocking between these models can be read in downward or upward fashion. Downwards, the architecture supports the *generation* of a distributed modeling environment, upwards the *integration* of existing environments. In both cases, the choice of the meta model is crucial for the support the model definition and integration environment can offer.

2.2 What Constitutes a Good M2-Model?

As in any modeling activity, the choice of a M2-model – what aspects of models to abstract from – depends on its purpose, and on the perceived obstacles in achieving this purpose.

Traditionally, the main purpose of defining M2-models has been the generation or integration of different modeling notations. The M2-model was simply a database language, for example, SQL in ISO-IRDS. Since the early 1990's, more dedicated M2-models have been developed. For example, the PROMOD CASE environment marketed by Cap debis employs a detailed M2-model of database modeling notations to support forward and reverse transformation between design and implementation models of database applications [9]. The MetaEdit environment of the University of Jyväskylä (Finland) uses the M2-model GOPRR with the basic concepts of Graph, Object, Property, Role, and Relationship for a graphical specification and automated generation of graphical requirements modeling notations [15].

However, once we have meta modeling available, we can go beyond notation mappings. Equally important are the ontological aspect of shared understanding and the social aspect of evolving agreement about perspectives.

In a given modeling effort, the different perspectives reflect the same reality. Therefore, the modeling constructs should also be traced back to a shared understanding what concepts of

reality are likely to be interesting for modeling. For example, in many of USU's customer organizations, there is still a significant problem of manual data transfer and re-entry. Therefore, the M2-model of PFR distinguishes explicitly between the data to be communicated and the media by which this is done, such that bottlenecks due to media change can be represented easily.

Integration of models is an ongoing social process which is not likely to lead to fully consistent perspectives even at the end. Research on viewpoint-oriented software development has recently pointed out the advantages of just maintaining information about the conflicts between perspectives rather than forcing consistency at each step [17]. In such cases, a distinction between workspaces representing the perspectives of individual group members and those representing the status of agreement in the whole requirements team need to be made [14]. In traceability meta models [18], the interrelationships between perspectives are therefore not just represented statically but record the process of integration as well.

Since the notations used in modeling are supposedly already targeted towards the relevant domain concepts, a good M2-model will usually address notational integration and domain integration simultaneously. However, especially in requirements engineering it is very important that the M2-model still remains simple. This combination of precise analysis and simplicity cannot be achieved by a universal M2-model for all application areas. Instead, we need a customizable meta modeling environment in which even the M2-model can be adapted to the bottlenecks of the particular modeling environment it is supposed to integrate. The Telos language supported by the ConceptBase system offers such an environment.

2.3 The ConceptBase Kernel Language

Telos offers a simple generic data model that is extensible to specific application needs and provides mechanisms for perspective integration.

The simplest conceivable diagrammatic notation consists of just two concepts: nodes (representing objects) and links (representing attributes and relationships). To allow any kind of formalization, we need a third concept, that of a logical assertion. In Telos, the user can employ assertions as deductive rules to derive additional information automatically, or as integrity constraints to control the entry of information by users. Finally, to talk about different notations, we need at least one abstraction mechanism – classification – which enables us to talk about classes and their instances. Figure 2 presents the data model as a semantic network.

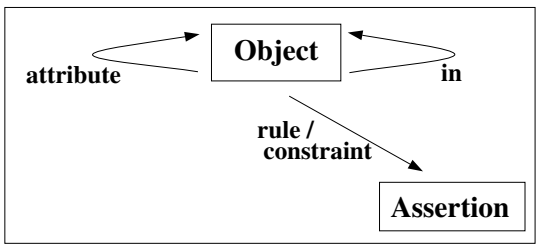

Figure 2: The Telos data model

Telos offers a special object to express the classification relationship among objects – the `in` relation. Characteristic features of this classification approach are:

- There is no limitation of the classification/instantiation hierarchy: classes can again belong to other (meta) classes which themselves can belong to (meta meta) classes, and so on.
- Classification in conjunction with object structuring specifies a form of typing from classes to instances, i.e., classes define the common structure of objects belonging to that class. Instances are allowed to instantiate attributes defined for their classes.
- Multiple classification is allowed: an object can belong to multiple classes. This can be interpreted in several different ways. One important aspect, taken by the query classes discussed below, is that multiple instantiation represents multiple abstractions of the same reality.

The kernel of the Telos language is just that. All other language facilities can be bootstrapped from this kernel of objects and attributes, assertions, and classification.

2.4 Customizing and Interrelating Requirements Perspectives

Using Telos' generic data model in conjunction with the unlimited classification hierarchy, we employ meta modeling to specify modeling languages and even the M2-model, and specify formal dependencies among different perspectives exploiting the assertion language.

2.4.1 Perspective Definition

Tailoring Telos to specific modeling languages is done by first embedding the structural regulations of the language (i.e. its syntax) into Telos, second giving the new modeling constructs a formal semantics by defining appropriate assertions, and third introducing the diagrammatic presentation of the language by assigning graphical type descriptions to the modeling constructs.

The infinite levels of classification available in Telos enable the creation of meta models which define attribute classes to be used at the level below and thus fix the language in which requirements models can be defined. The meta model acts then as a conceptual model of the structural part (the syntax) of the modeling technique.

Most implemented approaches to meta modeling cover the structural part well [13, 19] but offer semantic extensibility only within a predefined set of constraint types (e.g. cardinality constraints). Telos' assertion objects make it possible to specify the semantics of language extensions as part of the corresponding meta model. The formal behavior, defined in the form of integrity constraints and deductive rules, can be directly attached to the corresponding class definitions.

Traditional modeling languages are limited to the two-level class-instance paradigm: a formula is specified for a particular class and constrains the behavior of instances of that class but not, e.g., the behavior of instances of these instances. In ConceptBase, semantic extensibility is assisted by so-called *meta formulas* [10] which are assertions about objects which reside two or more instantiation levels below the objects they are attached to. For example, meta formulas attached to the M2-model constrain the behavior of requirements models defined in modeling languages which themselves are described under this M2-model.

By simply attaching graphical object types as conclusions to such meta formulas, graphical extensibility can also be achieved.

2.4.2 Perspective Resolution

Telos supports perspective resolution on basis of two forms of relationship specifications: a *specific* one which is specified for concrete notations on the meta class level, and a *general* one which is defined for the M2-model and valid for all notations.

A general relationship is fixed by defining a M2-model of which the different notation meta models used in the perspectives are instances; the perspectives for a particular project are instances of these instances (cf. section 2.1). When instantiating the M2-model with particular notations, meta formulas associated with the meta meta classes are specialized via partial evaluation (a logic program transformation technique) to rules and constraints interrelating specific perspectives. Rules enforce automatic and immediate propagation of changes, while integrity constraints prevent undesired perspective states.

Integrity constraints and deductive rules can be used to define relationships between the modeling primitives of notations. However, immediate constraint enforcement is known to cause undesirable interruptions during model development [2], and to suppress creativity. To enable an autonomous development process, delayed constraint enforcement or even just continuous monitoring is preferable.

In Telos, inconsistencies are observed and maintained as instances of so-called *query classes* [20]. A query class is a normal Telos class, but with forced class membership defined by a constraint. It is a `<name>`d object which selects from the intersection of a set of `<superclasses>` those that satisfy a first-order logic `<condition>`, and returns certain `<answer attributes>` which may be either stored data or computed by the condition:

```
QueryClass <name> isA <superclasses> with
  attribute
    <answer attributes>
  constraint
    <condition>
end
```

3 Experiences: Formalizing and Supporting PFR

USU applied the meta modeling approach to their PFR methodology and used the resulting combination of informal and formal methodologies in several customer projects. In the first of these projects, the first author helped in the use of ConceptBase, in subsequent projects the USU analysts used the system directly. We first give an informal overview of the methodology, then describe the development of the M2-model used for formalization, and finally give a few examples of query classes used with the M2-model for analysing interrelationships among the perspectives. The goal of this description is to illustrate the process, and to highlight some positive and negative experiences.

3.1 PFR: Description and Problems

The PFR methodology precedes traditional analysis and design methodologies such as ER/SA or OMT and comprises three steps:

1. An initial *shared vision* is defined in a two-day stakeholder workshop. The group makes a rough analysis of the current business processes (mostly in terms of information exchange among organizational units), analyses the goal structure behind the current pattern, identifies goal changes, and proposes a draft of the re-designed business process in which each process item must be linked to the goal structure.

2. The workshop identifies the *perspectives* of some individual stakeholders and/or existing documents as critical to success. These perspectives are then captured in detail by interviews, workflow analyses, and document content studies. This step has the goal of testing the initial vision against the existing and expected organizational context, and to elaborate it, both in terms of deepened understanding and in terms of more formal *representations* (e.g. in the form of activity sequences, data flow models, entity relationship diagrams or object models). The acquisition process is accompanied by a *cross-analysis* of the captured information for consistency, completeness, and local stakeholder agreement.

3. A second workshop is intended to draw the individual perspectives together and to achieve global *stakeholder agreement* on the requirements. The step is accompanied and followed by the development of a comprehensive requirements document of typically several hundred pages.

Even for rather complex projects, the goal is to complete the whole process in a matter of weeks rather than months. A major obstacle in achieving this goal has been the cross-analysis of heterogeneous perspectives in step 2. Due to time pressure, this analysis often remained incomplete. This led to repeating cycles of steps 2 and 3 due to problems detected only during the second workshop. In a few cases, it even led to problems in the final requirements document which showed up later as errors in the design, coding, or even usage testing phase.

From a representational viewpoint, the PFR methodology comprises a set of source perspectives as captured in the first two steps, a set of result perspectives which represent the delivered requirements (with the intent of presenting them to users or to use them in subsequent design tasks), and a set of business-oriented analysis perspectives encapsulating some generic knowledge about business processes.

Cross-perspective analysis mainly consists of a comparison of these perspectives, to detect discreparencies, modeling errors, gaps, and properties of the business process. The results of this comparison guide further interviews which clarify conflicts and complete the models.

USU's experience in applying this method to a large number of projects has shown that an analysis by hand based on the informal visual notation of the perspectives is time-consuming, haphazard, faulty, and often incomplete. A supporting tool should therefore

1. represent the information from all perspectives in a formal but natural way (which may be different from customer to customer or from project to project) such that they can be easily communicated to stakeholders,

2. represent perspectives even if they are violating the formal properties of the used notation or contradicting the contents of other perspectives, such that the analysis can take place in a separate step,

3. enable the comparison of diagrams represented according to different (semi-)formal notations, and maintain the detected conflicts during repair actions, since old conflicts may disappear and new problems may surface.

Initially, it seemed natural to use the same tools as in subsequent phases of analysis and design, such as ARIS [5], ADW [11], Bachman, Excelerator, or DecModel [3]. However, all of these models offer fixed paradigms of modeling that cannot be easily adapted to the specific needs of new methodologies or altered with growing experiences. For example, ARIS is mostly intended for supporting the reuse of pre-defined standard reference models. DecModel has its strength in the quantitative simulation of business processes. ADW, Bachman, and Excelerator focus on data and dataflow modeling. The formal analysis functions coming with these tools are specialized to their specific notations and tailored to the main intended purpose, rather than allowing for general-purpose consistency, completeness, and inter-personal conflict analysis. Where these tools are semantically extensible, these extensions have to be programmed in imperative style rather than declaratively specified. This also holds for more general meta data management tools such as RDD-100 [1] or MetaEdit.

3.2 ConceptBase Training

Training of an initially small number of USU analysts in using the ConceptBase system and its modeling formalism took about a week. Some of the formal basis had to be covered because the analysts also had to develop the M2-model and the associated query classes. For analysts just using the tuned system, a more surface-oriented approach – how to work with Telos models and the various notations offered for individual perspectives and cross-perspective analysis at the ConceptBase user interface – should be sufficient.

The prototype nature of the system and the fact that it runs under Unix led to some extra training needs as well since USU developers were not used to this environment. Still, the analysts were able to use the system without regular help by the developers after the first pilot project.

3.3 Customizing a M2-Model Towards the PFR Method

The construction of the M2-model was not a straightforward task. The model passed through several iterations until a stable version was reached. The difficult part, however, turned out to get the M2 model right, while actually changing it in the system was very easy.

The development started with a first theoretic discussion about the representation of PFR perspectives. Based on a five page introduction, we tried to identify modeling concepts necessary to express the workshop results. We came up with a set of powerful but complex models including constructs to describe the overall organisational structure, composition and decomposition of activities, and much more. The discussion at that time was more concerned with the specification of all imaginable modeling concepts than the restriction to useful ones.

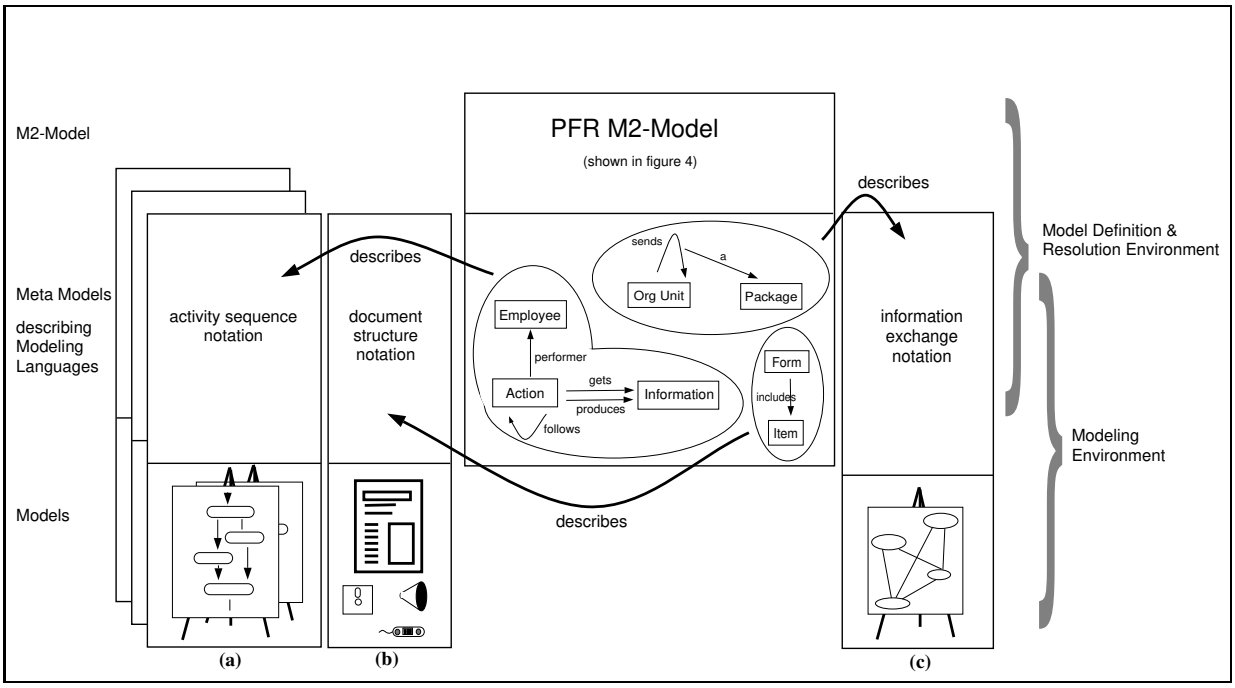

Figure 3: The PFR perspectives within the IRDS architecture

Trying the models in the PFR workshop for the internal pilot project led to the observation that the initial models were unnecessarily complicated and did not really help conflict analysis. Discussions with experienced PFR group facilitators using hypothetical cases, clarified the importance of each construct in the initial model for actual applications. Besides the expressiveness we were also concerned with the readability of the model. A simple model that can be easily communicated to stakeholders turned out to be a major goal. The ConceptBase system helped this refinement process because it made it very easy to change the M2-model, as long as the underlying meta models and application models had not grown too big.

In hindsight, it can be said that USU employed the following heuristic criteria in defining a "good" M2-model:

- The M2-model should contain a small number of easy-to-understand basic constructs such that it can serve as a basis for human communication.
- It should cover all of the individual methods used in the given methodology, such that each construct of the M2-model is covered by at least two different methods. Similarly, the process of capturing requirements should make sure that crucial items are captured from at least two different perspectives (different methods or at least different instances of the same method).

In other words, the approach aims at eliciting the maximum degree of potential conflict for a given RE effort. Note that this is in strong contrast to most current object-oriented modeling techniques which seem to aim at orthogonal modeling constructs with as little redundancy as possible. We contend that while orthogonality is very suitable for the design phase, it is not very productive in requirements elicitation.

The end result of this meta modeling effort is shown in figure 3, a concretization of the upper levels in figure 1. Each modeling environment (a) to (c) represents one of the source perspectives. The data part of the model definition & resolution environment describes and links the descriptions of all supported notations. According to the approach reported in section 2.4.1 this model covers the syntactic, semantic, and graphical specification of each notation.

Not visible in this diagram is the realization of the data part of each perspective as an individual knowledge base partition. ConceptBase provides a context mechanism to partition a knowledge base into independent units. We can therefore easily identify each environment and handle multiple instances of the same perspective type. This is a rudimentary solution to the problem of addressing the process of agreement; the actual process was done manually by the group facilitator and analysts, and just recorded in the system.

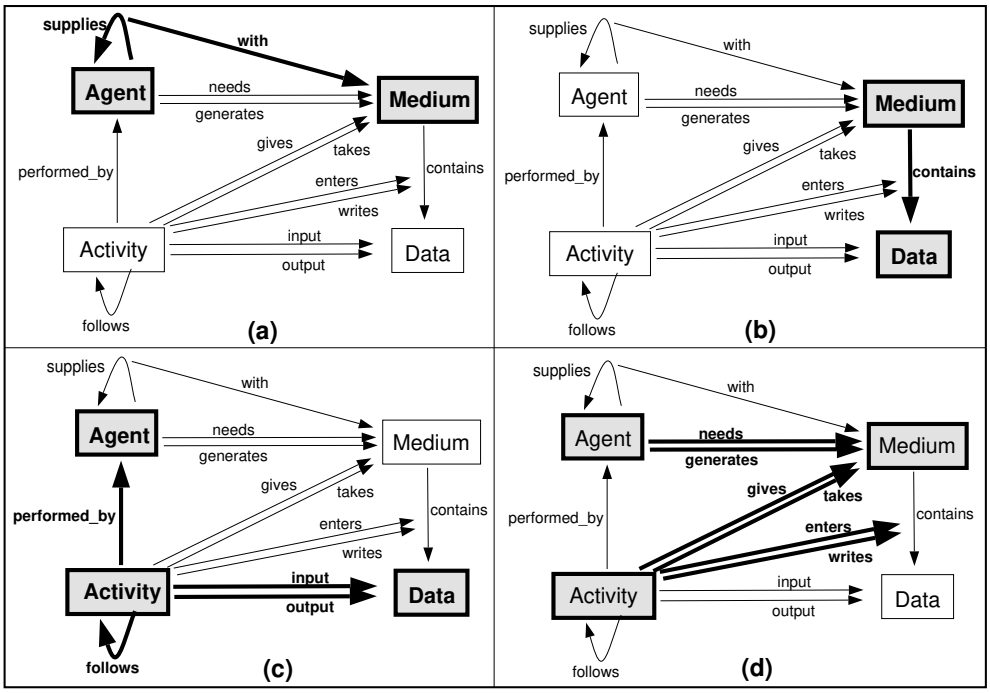

Figure 4: The PFR perspectives within the M2-model

The integration was performed in two steps: the syntactic and the semantic integration. For the syntactic integration we abstracted from the individual perspective models of figure 3 to the common M2-model of figure 4.

Similarities between the concepts, as, e.g., between `Package` and `Form`, led to multiple instances of the same meta meta class. This was not the only possible strategy for an syntactic integration. The design of one integrated M2-model for all perspective schemata, and the organisation of the individual and the integrated model within a specialization hierarchy were also considered as alternatives. USU favored the above solution because of its simplicity and extensibility. After a short introduction even customers were able to understand the four main concepts of the M2-model.

We gradually explain the M2-model through these perspectives which is also the way how it was developed.

The shaded nodes in part (a) of the figure are instantiated with the *information exchange between organisational units*, as captured in the "collage" of the initial workshop[1]. We model an organisational unit as an `Agent` who `supplies` another agent `with` a `Media`. The earliest version of the M2-model had this simply as a data flow but, observing the participants of the first pilot project, we recognized that agents do not really exchange information, but the medium that carries the data. A medium can be something persistent, like a piece of paper, a form or a disk, or a transient thing like the voice that carries words.

Part (b) represents *the structure of exchanged media* by explicitly distinguishing the `Medium` and the `Data` it `contains`. This distinction is essential to talk about phenomena such as empty and completed forms, reading from and writing to a medium, replicating the same piece of data on multiple media manually or automatically, and agents that get a medium but perform no activity that needs or produces any data located on that media. For example, in an application project, the same data was captured and re-captured several times in a workflow, with very good and expensive quality controls, except in the last step! Here, the model helped to explain why there was bad quality despite high quality control costs.

The conceptual model of the *individual activity sequence of stakeholders* is shown in part (c). An `Activity` is `performed_by` an `Agent`. A partial order on activities (workflow) is expressed by the `follows` relation. An activity is an atomic action that takes some information or `Data` as `input` and generates new `Data` as `output`. Our semantics of the `output` relation is very rigorous: The activity must create this data for the first time, i.e. no other activity can also create this data.

In addition to the source perspectives directly captured by the PFR method and shown in the parts (a) to (c), the M2-model includes knowledge supporting the analysis of the acquired information. This knowledge creates new perspectives which overlap with the source perspectives and therefore provide the glue between them. Although this knowledge can be deduced from the source perspectives, our approach captures it explicitly in interviews and integrates it into the model. This highly redundant information leads to numerous conflicts which guide further clarification steps to reconcile the perspectives. Part (d) highlights the following examples: `needs` and `generates` describe which media agents require from other agents, and which one they produce, i.e., bring for the first time into the document flow. This information can be used first to validate the actual flow and second to optimize the document flow between agents. The bridge between the "information flow between organisational units" and the "activity sequence working on data" is provided by `gives` and `takes`. They enable the refinement of the perspective covering the relationship between agents and media to the perspective on the relationships between the corresponding activities and the media. A further refinement towards the activity-oriented source perspective is provided by `enters` and `writes`: For an activity the concrete access to information on a medium, which is reading or writing, can be expressed.

3.4 Resolution of PFR Perspectives

Perspectives were related semantically by query classes which define consistency of knowledge *within* an individual perspective as well as the consistency *between* different perspectives.

[1] If the workshop were conducted in a electronic meeting room (which it currently is not), the collage interface types could easily be created in ConceptBase using the graphical extensibility feature of Telos, by simply introducing meta rules that have multimedia objects as their conclusions.

The analysts started with defining simple queries for the information exchange perspective: Which organisational unit doesn't supply anything? Which is not supplied with anything? This definition of a simple formal semantics for the perspective models was quite helpful in understanding the perspectives.

In the first pilot projects, the analysis team of USU was confused about the large set of semantic conflicts they detected especially within the information exchange perspective. The group facilitator pointed out that this was intentional. The generation of much, even inconsistent information is considered more important than the generation of less but consistent information. During the workshop, he accepted the inconsistent use of, e.g., links denoting an information flow. He argued that too many interruptions and corrections, i.e. a too rigid guidance of the development team, would destroy the desired stimulating atmosphere. He preferred a deferred but computer supported consistency check for this method.

The USU application projects identified more than 70 constraints describing the consistency of the captured information, which could largely be reused across all application projects.

About a dozen additional query classes served to analyse the properties of the finally reconciled business process model. This includes questions like "What is the trace of form X305 ?", to detect the reason for the long handling time of the form X305. All together, USU produced over 80 query classes.

The development of these queries was begun while developing the M2-model structure and continued throughout the first two application projects. By that time, a fairly stable state was reached and coded in an analysis handbook. For short queries it was relatively easy to understand their meaning by looking at their specification. But after defining some more complex ones we observed that an additional textual description is necessary. USU therefore set up a 'use handbook' containing the query specification, the technical process of calling it using ConceptBase, a textual explanation of the indended meaning, and a set of possible interpretations of the answers. This was quite helpful in follow-up projects where also untrained consultants were able to analyse the perspectives just by following the instructions in this handbook.

A further step clustered queries about a particular perspective type and discussed their interrelationships. This resulted in a sequence of queries guiding the analysis process. In this more systematic process, milestones depend on results of the analysis queries: If all queries regarding the information exchange perspective return no more conflicts, the perspective is declared to be consistent and the project achieved a milestone. As further commercial projects indicated, this way of specifying the state of the analysis process increased the reliability in the whole methodology.

Some of the complex queries initially took a long evaluation time. To solve this problem, query classes can be parameterized to limit the search space to a manageable size. This resulted in an extended set of fast queries; the size of this set depends on the size of the actual model.

In the following, we present examples of query classes and answer interpretations. The examples are organized according to the principle character of the queries, namely the analysis of a single perspective, the analysis of interrelationships among perspectives, and the analysis of the whole business process. For readability, we simplified the condition parts of the queries in the sense that we omitted all statements specifying the meta level where the condition should be evaluated.

Analysis of a single perspective. Consider the activity sequence perspective. The query class below realizes the constraint that data can only be used by an activity (indicated by the `input` relation) after it was created (via the `output` relation). The query deduces data that are used as `input` before they are produced.

```
QueryClass Data_UsedBeforeProduced isA Data with
  attribute
     early_user : Activity
  constraint
     c : $ (early_user input this) and exists producer
           (producer output this) and
           (producer transitively_follows this) $
end
```

The symbol `this` used in the condition refers to potential answer objects.

The `transitively_follows` relation denotes the transitive closure of the `follows` relation and is specified by a recursive Telos rule. The answer can be interpreted as

1. a mistake, if the interviewed employee indicated a wrong sequence.
2. a mistake of the interviewer who modeled an `output` relation to `Data` instead of an `input` or a wrong direction of the `follows` link.
3. a conflict, where two `data` objects are used as homonyms.
4. a conflict between the activity sequences of two agents.
5. a not expressible phenomenon, where the conflicting second `output` is a correction of the `data` resulting from the initial `output`

Analysis of interrelationships among multiple perspectives. The consulting projects often surfaced contradictions between the high-level information exchange perspective acquired mainly from managers, and the detailed activity sequence perspective captured from the real working agents. An often violated interrelationship states that the medium flow among agents must correspond to the data demand of agent's activities, i.e. the supplied media must contain some data that is required by an activity and, conversely, all required data must be contained on some delivered medium. The following query class implements the first part and deduces all media that is supplied to an agent who performs no activity that needs any data carried by that medium.

```
QueryClass NotUsedMedium isA Medium with
  attribute
     not_user : Agent
  constraint
     c : $ exists supply
           (supply in Agent!supplies) and (supply to not_user) and
           (supply with this) and
            not exists action, info
              (action performed_by not_user) and (this contains info) and
              ((action input info) or (action output info)) $
end
```

The answers are the media together with the agent who gets the media but does not use it. They can be interpreted as follows:

1. A mistake caused the answer: either the `input` or `output` relationship of an activity to a data particle of the medium, or a whole activity was overlooked.

2. There exists a mismatch between the captured perspectives: the management and the concrete employees view the process in different ways. Further clarification interviews are necessary to reconcile the contradicting views.

3. The model is correct. The agent actually gets and sends the medium without any interest on the data. The business process can be improved by optimizing the media flow.

4. The model is correct. The business process is ok. The activity that works on the medium does not require any information from the medium.

In practice, we often observed the problem described in 2. As an example of interpretation 4, a secretary collected the monthly reports of the employees of a department to give them as one piece to the manager of that department.

Business process analysis. Much of the Business Process Re-engineering literature contends that the document flow within a company offers potential for optimization. The following query analyses the media flow between agents and searches for agents who get a document that contains only data that is already supplied to them by other media.

```
QueryClass MultipleDataSupply isA Agent with
  attribute
    superfluous : Medium
  constraint
     c : $ exists supply
           (supply in Agent!supplies) and (supply to this) and
           (supply with superfluous) and
            forall data (superfluous contains data) ==>
            exists other_medium (other_medium <> superfluous) and
             (other_medium contains data) $
end
```

The results can be interpreted as follows:

1. The supply of the deduced medium is unnecessary and the whole data is also available from other media. Either this is an intended situation and the agent should choose among the different media, or it is not intended and the delivery of that medium is subject to further optimization studies.

2. The agent performs a comparison check of the same data located on different media.

3. The agent must perform the same activity on the same data carried by different media. So far, this fact cannot be expressed by our M2-model.

3.5 Using the Analysis Tools

As stated before, the stabilization of the M2-model and of the set of analysis query classes proceeded in parallel to a number of customer projects in which these models and query

classes were actually used. In each of these projects, the full variety of problems indicated in the previous subsection were encountered: simple clerical errors in writing down requirements, incomplete capture of requirements, conflicting understanding of stakeholders, unsatisfactory reality. The rapid identification of conflicts focused the attention of analysts and thus made the process faster and more problem-oriented.

The perspectives of the PFR methodology are captured and conflicts are recognized quite well, even when the methodology was adapted to specific customer habits. Nevertheless, analysts felt that an important element of requirements information could not yet be captured in the system, let alone exploited for better conflict handling: the discussion of specific usage scenarios at the instance level, e.g. illustrating problems or situations from which the perspective models are abstracted.

The final product of the combined informal capture and formal analysis process are quality-assured requirements documents delivered to customers. In these documents, a table or spreadsheet provides a short and clear overview of the captured perspectives and the remaining problems. It would have been desirable to have an automatic loading facility from ConceptBase to such a spreadsheet-like notation (which is now under development). This also holds for downloading graphics produced with the ConceptBase graph browser easily into word processors.

A further-going demand by customers was to keep the resulting requirements information model itself for purposes of traceability, explanation of system decisions, and maintenance. Realizing this wish would require a further simplification of system usage beyond what is needed for supporting the facilitator and analysts, and just presenting the results to stakeholders.

Even so, an important effect of using ConceptBase for conflict analysis was de-personalization of the conflict resolution process. Concerned employees accepted errors reported by the anonymous computer while they perceived errors detected by a human consultant as a personal attack. People often believed the computer supported analysis more than the human one. The possibility to visulize the analysis process by presenting the data, the queries, and the answers increased the confidence in the analysis results. The consultants therefore employed the computer-supported analysis not only to drive the analysis but also as a help in their argumentations. In one project, this phenomenon also helped (to some degree!) to reduce the tensions between different departments who had stakes in a planned reorganisation of the information processing environment resulting from the analysis.

3.6 Summary of Experiences

If we check the developed system against the requirements for tool support for the PFR method established in section 3.1, it can be said that the meta modeling approach satisfied most but not all of them:

1. By simultaneously addressing notational and domain integration, we got a relatively simple M2-model which still captures knowledge about business processes as the problem domain. We also used it to explain the answers to the analysis queries to customers. Its abstract and therefore simplified structure proved to be very useful exactly for this task.

2. Through the simple device of using query classes instead of integrity constraints to formalize the semantics of and the dependencies between perspectives, the system is able to tolerate (but monitor) the evolution of inconsistent information. However, the process for dealing with these inconsistencies is still left to the usage handbook, and we cannot report experiences how process guidance within the system would improve on this.

3. The handling of queries as classes enables the ad-hoc integration of upcoming new analysis approaches. But it turned out to be a problem to monitor inconsistencies and conflicts permanently during the perspective analysis process. An update which repairs a particular inconsistency in one perspective may cause a conflict in another one. To be sure to detect all of them, the system had to re-evaluate many of the 80 queries against the new knowledge base state. We are currently working on a mechanism to incrementally recompute only relevant queries in an optimized manner.

4 Lessons Learned

The USU experiences show that conceptual meta modeling technology can provide a valuable complement to informal teamwork methods of business analysis and requirements engineering. In particular, based on the representation and cross-analysis of multiple requirements perspectives, the identification of a wide variety of conflicts and, perhaps more importantly, the ongoing monitoring of these conflicts can be facilitated.

However, the experiences show equally clearly that the above statements have to be qualified. The definition of fixed relationships between meta models is not robust enough against the variations in the form and usage of modeling notations dictated by the different practices in consulting or user companies, or by the different bottlenecks driving the modeling process.

The ability of a modeling formalism to adapt M2-level models does not only provide this robustness to a large degree but also allows to focus meta modeling on domain-oriented as well as on notation-oriented aspects, thus increasing the degree of support. To achieve this, a number of guidelines for developing M2-models have been identified. These guidelines are not only backed by the specific experiences reported in this paper but also by similar projects using ConceptBase for model integration in scientific and engineering settings [8]:

- The perspectives and the M2-model have to be jointly optimized as to provide a sufficient overlap in order to provoke the detection of interesting conflicts. The design-time idea of orthogonality, present in some object-oriented modeling technologies, is not appropriate for understanding which requires a lot of redundancy.

- The M2-model can become an important communications medium for discussions of stakeholders about the relationships between their perspectives – it represents a shared basic understanding of the problems at hand. Of course, the M2-model can only serve this purpose if it is simple enough, i.e., does not contain more than maybe half a dozen basic concepts. It is obvious that such a simple M2-model cannot be at the same time be precise in revealing kernel RE problems and general across application domains.

- The desirable combination of robustness against minor methodological variations with powerful conflict identification and maintenance requires a formal background expres-

sive enough to cover a broad variety of constraints, yet simple enough to allow computerized reasoning. The experiences have shown deductive database technology to be a good starting point, but also show the need of innovative uses of such implementation technologies, e.g. to handle meta-formulas and their automatic specialization.

The experiences also pointed out some challenges for further technology improvement which the ConceptBase team is currently addressing. First, the parallel monitoring of a very large number of conflict analysis views is a performance bottleneck which needs to be addressed by novel incremental view maintenance techniques. Second, the current version of the system only supports the facilitator or analyst in recording and analysing what happens in a group; additional problems need to be addressed when going from a central workshop setting to distributed requirements engineering where stakeholders directly enter and negotiate their perspectives. Temporally, this can go beyond the scope of the RE project itself as customers keep the created requirements knowledge base after the process for purposes of traceability and maintenance.

Last not least, a major gap in the approach, both from the technological perspective and from the methodological perspective, is that conflict analysis is only driven from the top, the shared M2-model. The complementary grounding of negotiations, shared scenarios of reality rather than shared abstractions, was often observed in application projects but could not yet be directly captured in the ConceptBase models. We hypothesize that scenarios may be a way to go beyond conflict identification and monitoring towards active support for conflict resolution. The basic architecture shown in figure 1 includes this level but significant research will be required to work out its formal and technical interaction with the approach presented here.

References

[1] M. Alford. Attacking requirements complexity using a separation of concern. In *Proc. of the First Intl. Conf. on Requirements Engineering (ICRE)*, pages 2–5, Colorado Springs, Colorado, April 18-22 1994.

[2] R. Balzer. Tolerating inconsistency. In *Proc. of the 13th Intl. Conf. on Software Engineering (ICSE)*, Austin, Texas, USA, May 13-17 1991.

[3] Digital Equipment Corporation. *DECmodel Modeler's Guide*, 1993.

[4] M.S. Feather and S. Fickas. Coping with requirements freedom. In R. Balzer and J. Mylopoulos, editors, *Proc. Intl. Workshop Development of Intelligent Information Systems*, pages 42–46, Niagara-on-the-Lake, Ontario, Canada, April 21–23 1991.

[5] IDS Prof. Scheer GmbH, Saarbrücken. *ARIS-Toolset Manual V1.0*, 1994.

[6] ISO. *Information Technology - Information Resource Dictionary System (IRDS) Framework, ISO/IEC 10027.* ISO International Standard, 1990.

[7] M. Jarke, R. Gallersdörfer, M.A. Jeusfeld, M. Staudt, and S. Eherer. ConceptBase - a deductive object base for meta data management. *Journal of Intelligent Information Systems, Special Issue on Deductive and Object-Oriented Databases, M. Kifer, editor*, 4(2):167–192, March 1995.

[8] M. Jarke, M.A. Jeusfeld, and P. Szczurko. Three aspects of intelligent cooperation in the quality life cycle. *Intl. Journal of Intelligent and Cooperative Information Systems*, 2(4):355–374, 1993.

[9] M. Jeusfeld and U. Johnen. An executable meta model for re-engineering of database schemas. In P. Loucopoulos, editor, *Proceedings of the 13th International Conference on the Entity-Relationship Approach*, pages 533–547, Manchester, UK, December 13-16 1994. Springer-Verlag, LNCS 881.

[10] M.A. Jeusfeld. *Update Control in Deductive Object Bases.* PhD thesis, University of Passau (in German), 1992.

[11] KnowledgeWare. *Application Development Workbench User Guide*, 1993.

[12] J.E. Kottemann and B.R. Konsynski. Dynamic metasystems for information systems development. In *Proc. of the 5th Intl. Conf. on Information Systems*, Tucson, Arizona, November 28-30 1984.

[13] K. Lyytinen, P. Kerola, J. Kaipala, S. Kelly, J. Lehto, H. Liu, P. Marttiin, H. Oinas-Kukkonen, J. Pirhonen, M. Rossi, K. Smolander, V.-P. Tahvanainen, and J.-P. Tolvanen. MetaPHOR: Metamodeling, principles, hypertext, objects and repositories. Technical Report TR-7, University of Jyväskylä, December 1994.

[14] N. Maiden, P. Assenova, P. Constantopoulos, M. Jarke, P. Johannesseon, H.W. Nissen, G. Spanoudakis, and A. Sutcliffe. Computational mechanisms for distributed requirements engineering. In *Proc. of the 7th Internatioal Conference on Software Engineering and Knowledge Engineering*, pages 8–15, Rockville, Maryland, USA, June 22-24 1995.

[15] P. Marttiin. Towards flexible process support with a CASE shell. In G. Wijers, S. Brinkkemper, and T. Wassermann, editors, *Proc. of the 6th Intl. Conf. on Advanced Information Systems Engineering (CAiSE'94)*, pages 14–27, Utrecht, The Netherlands, June 1994. Springer-Verlag, LNCS 811.

[16] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325–362, October 1990.

[17] B. Nuseibeh, J. Kramer, and A. Finkelstein. Expressing the relationships between multiple views in requirements specification. In *Proc. of 15th Intl. Conf. on Software Engineering (ICSE)*, Baltimore, Maryland, USA, May 17-21 1993.

[18] K. Pohl. *Process Centered Requirements Engineering.* John Wiley & Sons Ltd, UK, 1996. to appear.

[19] M. Saeki, K. Iguchi, K. Wen-yin, and M. Shinohara. A meta-model for representing software specification & design methods. In N. Prakash, C. Rolland, and B. Pernici, editors, *Proc. of the IFIP WG8.1 Working Conference on Information System Development Process*, Como, Italy, 1-3 September 1993.

[20] M. Staudt, H.W. Nissen, and M.A. Jeusfeld. Query by class, rule and concept. *Journal of Applied Intelligence*, 4(2):133–156, 1994.