ConceptBase: Managing Conceptual Models about Information Systems

Manfred A. Jeusfeld[1], Matthias Jarke[2], Hans W. Nissen[2], and Martin Staudt[3]

[1] KUB Tilburg University, Infolab, 5000 LE Tilburg, The Netherlands
[2] RWTH Aachen University of Technology, Informatik V, 52056 Aachen, Germany
[3] Swiss Life, Information Systems Research, 8022 Zürich, Switzerland

Summary. ConceptBase is a meta data management system intended to support the cooperative development and evolution of information systems with multiple interacting formalisms. It supports a simple logic-based core language, O-Telos, which integrates deductive and object-oriented features in order to support the syntactical, graphical, and semantic customization of modeling languages as well as analysis in multi-language modeling environments.

1. Multi-language Conceptual Modeling

Conceptual models offer abstract views on certain aspects of the real world (description role) and the information system to be implemented (prescription role) [You89]. They are used for different purposes, such as a communication medium between users and developers, for managing and understanding the complexity within the application domain, and for making experiences reusable. The presence of multiple conceptual modeling languages is common in information systems engineering as well as other engineering disciplines. The reasons are among others:

- the complexity of the system requires a decomposition of the modeling task into subtasks; a frequent strategy is to use orthogonal perspectives (data view, behavioral view, etc.) for this decomposition;
- the information system is decomposed into subsystems of different type, e.g. data storage system vs. user interface; experts for those subsystems tend to prefer special-purpose modeling languages;
- the modeling process is undertaken by a group of experts with different background and education; the experts may have different preferences on modeling languages;
- conceptual modeling has different goals (e.g., system analysis, system specification, documentation, training, decision support); heterogeneous goals lead to heterogeneous representation languages, and to heterogeneous ways-of-working even with given languages.

The pre-dominant approach to solve the integration problem is to "buy" an integrated CASE tool which offers a collection of predefined modeling languages and to apply it in the manner described in the manual. There are good reasons to do so: the method design has already been done and

the interdependencies between the multiple modeling languages have already been addressed by the CASE tool designers. Moreover, a CASE tool supports the standardization of information systems development within an enterprise.

Still, there are information systems projects that require more flexibility in terms of modeling language syntax, graphical presentations, and semantics of modeling language interactions. The Telos meta modeling language has been developed to address these concerns. Its implementation in ConceptBase, a meta data management system based on the integration of deductive and object-oriented technologies, supports an Internet-based architecture intended to support flexible and goal-oriented distributed cooperation in modeling projects.

1.1 A Brief History of Meta Modeling

In the mid-1970s, several semiformal notations supporting the development of information systems were developed. The use of some of these became standard practice in the 1980s, especially entity-relationship diagrams for data modeling and dataflow diagrams for function modeling. More recently, object-oriented methods have added notations for behavior modeling, such as Statecharts, giving a broader picture of the specification and an easier mapping to implementations in languages like C++ or Java.

It was recognized early on that managing large specifications in these notations posed serious problems of inconsistency, incompleteness, maintenance, and reuse. Conceptual modeling languages incorporate ideas from knowledge representation, databases, and programming languages to provide the necessary formal foundation for users with limited mathematics background.

In early 1980s, Sol Greenspan was the first to apply these ideas to requirements engineering, when he formalised the SADT notation in the RML language [GMB94]. This was a precursor to numerous attempts worldwide. Initially, these languages embodied a fixed ontology in which requirements engineering could be described. As early as 1984, it was recognized that modeling formalisms must be customizable. Jeff Kotteman and Benn Konsynski proposed a basic architecture that included a meta meta model (M2-model for short) as the basis for using different notations within a development environment [KK84]. ISO's Information Resource Dictionary System (IRDS) [ISO90] standard generalized this idea to propose an architecture that combines information systems use and evolution. Figure 1.1 shows its four-layer architecture applied to the conceptual modeling activity.

The **Instances and scenarios level** contains objects which cannot have instances. Examples are data, processes, system states, measurements and so on. Objects may have attributes and they may have classes (residing in the model level). During design, when the information system and therefore the instances do not yet exist, this level also contains scenarios of the intended use of the system.

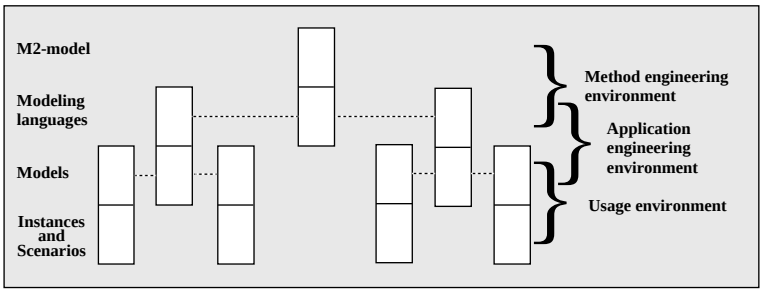

Fig. 1.1. The ISO IRDS architecture applied to conceptual modeling

The **Models level** represents the classes of the objects at the instance level. Those classes define the schema (attributes, properties) of the instance level objects as well as rules for manipulating these objects. At the same time the classes are themselves instances of the schema defined at the modeling language level.

At the **Modeling languages level**, meta classes define the structure of the objects (classes) at the model level. In other words, a model is instantiated from the meta classes of the modeling language level. In section 3., the modeling language level will be used to define specific graphical notations *and* their interrelationships.

The **M2-model level** contains meta meta classes (M2-classes). They are classes with instances at the modeling language level. Multiple modeling languages are possible by appropriate instantiations from these M2-classes. Moreover, the dependencies between the multiple languages can be represented as attributes between M2-classes in the M2-model level.

The four IRDS levels can be grouped in pairs that define interlocking environments, as shown on the right side of the figure: usage environments, application engineering environments, and the method engineering environment, which manages the interrelationships among modeling languages and the interactions among modeling tools. The interlocking between the models can be read down or up. Reading down, the architecture supports the *generation* of a distributed modeling environment; reading up, it supports the *integration* of existing environments. In either case, the choice of metamodels is crucial for the support the model definition and integration environments can offer.

However, modeling languages do not just have a programming language syntax which needs to be customized. The customization should also address graphical conventions of the modeling formalisms; for example, the mobile phone developer Nokia employs more than 150 method variants in terms of notation, graphics, and ways-of-modeling. Moreover, the correct usage of each formalism and the consistency of models that span across different modeling formalisms should be definable.

Since the late 1980s, more dedicated M2-models have been developed, as discussed in the next subsections. In parallel, the need to have generalized languages dedicated to meta modeling and method engineering was recognized by several people. In several iterations, a number of European projects [JMSV92] jointly with the group of John Mylopoulos at the University of Toronto developed the language Telos [MBJK90] which generalized RML to provide a meta modeling framework which integrates the three perspectives of structured syntax, graphical presentation, and formal semantics.

However, early attempts to implement the full Telos language (as in the first version of ConceptBase [JR88]) showed that its semantics was still too complicated for efficient repository support based on known technologies. Three parallel directions were pursued by different, but interacting and partially overlapping groups of researchers.

The MetaEdit environment developed at the University of Jyväskylä [KLR96] is a good example of an effort focusing on graphics-based method engineering, i.e. the graphical definition of graphical modeling formalisms.

Starting from early experiences with ConceptBase in the DAIDA project [JMSV92] the Semantic Index System developed in ESPRIT project ITHACA [CJMV95] focused on an efficient implementation of the structurally object-oriented aspects of the Telos language. It may be worth noting that the recently announced Microsoft Repository [BHS+97] has generalized such an approach to full object orientation based on Microsoft's Common Object Model.

Complementing these structural concerns, the first step in the further development of ConceptBase within ESPRIT project Compulog focused on the simplification of the logical semantics. The dissertation [Jeu92] showed that the non-temporal part of Telos, with very minor modifications, can be based on the perfect model semantics of deductive databases with negation [CGT90], resulting in the O-Telos dialect used in the present version of ConceptBase [1] [JGJ+95]. Thereby, the diagrams denoting the structure became explicit facts in the database (of concepts), the syntactical constraints are represented as deductive rules or queries or integrity constraints, and the manipulation services are expressed as restrictions on how to update the database. This simple formalization thus was a prerequisite of the re-integation of syntactical, graphical, and semantic aspects of meta modeling, as discussed in section 2. below.

1.2 Three Basic Modeling Methodologies

As observed in [Poh94], modeling processes proceed along three dimensions: representational transformation, domain knowledge acquisition, and stake-

[1] ConceptBase has been developed since 1988 and been applied in numerous modeling projects around the world. The current version of the system can be obtained from the address http://www-i5.informatik.rwth-aachen.de/CBdoc for research and evaluation purposes.

holder agreement. Existing methodologies tend to emphasize one of these dimensions over the others: the modeling *notations*, the available *knowledge* within a specific domain, or the *people* involved in the analysis project. All three methodologies have long histories, with little interaction between them. All of them use multiple modeling perspectives but the purpose of these and therefore the integration strategies are quite different.

Notation-oriented methods manifest their assistance in the set of modeling notations they offer. Their philosophy can be characterized by the slogan *In the language lies the power*. Examples of notation-oriented methods are structured analysis approaches, as, e.g., Modern Structured Analysis (MSA) [You89], and object-oriented techniques, as, e.g., the Unified Modeling Language (UML) [FS97]. A large number of CASE tools in the market offer graphical editors to develop models of the supported notations and check the balancing rules that must hold between models of different notations. The notations as well as the constraints are hard-coded within the tools and are not easily customizable by users.

A completely different strategy is employed by the **domain-oriented analysis methods**. For a specific application domain, e.g., public administration or furniture industry, they offer a predefined set of reference models. Reference models describe typical data, processes and functions, together with a set of consistency tests which evaluate relationships between the models. Reference models represent the knowledge collected in multiple analysis projects within a particular domain: *In the knowledge lies the power*. The reuse of reference models can strongly reduce the analysis effort. However, it can be inflexible since the user can tailor the notations, the constraints or contents only to the degree foreseen by the developers of the reference models, or completely loses the help of the method.

The ARIS Toolset [IDS96] offers a platform for working with reference models. It also offers hard-coded constraint checks within and across the models. These tests are programmed individually and new tests can be added manually, without a coherent theory, even though the concept of event-driven process chain (EPC) provides a semi-formal understanding [Sch94]. Towards a more formal approach, the NATURE project has defined formal problem abstractions [MSTT94] via a M2-model which defines principles for the specification of domain models.

Goal- and team-oriented approaches specifically address the objective to capture requirements from multiple information sources and to make arising conflicts productive. They incorporate stakeholder involvement and prescribe general process steps rather than notations or contents: *In the people lies the power*. Prominent examples include IBM's JAD (Joint Application Design) [Aug91], SSM (Soft Systems Methodology) [Che89], and PFR (Analysis of Presence and Future Requirements) [Abe95]. In these methods highly skilled group facilitators animate the participants, guide the analysis process and keep an eye on the compliance with the specified analysis goals. The

general idea is to get as much information as possible from different sources in a short time.

Teamwork remains very informal to enhance creativity. Neither notations nor analysis goals are predefined by the methods but specified by the participants according to the actual problem to be solved. To accommodate the change of goals during project execution, the customization of analysis goals and notations is required even during a running project. Outside ConceptBase few supporting tools are available beyond simple groupware tools. The main reason for this dilemma is the high degree of customizability the tools must offer. They must be extensible towards new notations and flexible enough to support changing analysis goals.

1.3 Goals and Architecture of ConceptBase

The design of ConceptBase addresses the following goals:

1. The system should include a feature to define *and interrelate* specialized conceptual modeling languages in an cost-effective way. The language should reflect the modelers' need of key concepts types and their interpretation of those concepts.
2. The system should be extensible at any time. when the need for a new concept type occurs, it should be possible to include it into the conceptual modeling language definition in terms of language constructs, graphical presentation, and semantic constraints.
3. The system should not only check the syntactic correctness within and between models, but also allow to memorize patterns that indicate semantic errors in the models. The memory of those patterns should be extensible and adaptable to the user's growing experience, thus support organizational knowledge creation [Non94].

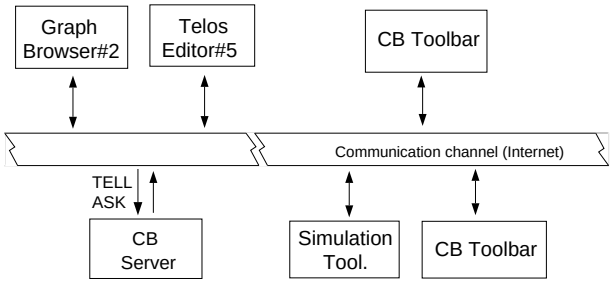

Fig. 1.2. ConceptBase is a client-server metadata manager

ConceptBase is realized in a client-server architecture (cf. figure 1.2). The ConceptBase server stores, queries, and updates Telos models. The server offers the method TELL for updating the object base and the method ASK

for querying its contents. Persistent object storage is implemented in C++. Reasoning services for deductive query processing, integrity checking, and code generation are implemented in Prolog.

A ConceptBase client is often a modeling tool, either graphical or textual, but it could be another application, such as a simulation tool. The Internet is the medium for the communication between the server and the clients. Programming interfaces for various toolkits, including Andrew, Tcl/Tk, Ilog Views and Java exist. The distributed version of ConceptBase includes a standard usage interface, along with advice on how to develop your own.

2. The O-Telos Language

Like other conceptual modeling languages, O-Telos offers a textual and a graphical representation. Both are structurally extensible through our meta modeling approach, encoded in the basic language structure. However, the distinguishing feature of O-Telos in comparison with other meta modeling approaches is its simple logical foundation which enables (a) efficient implementation using experiences from deductive database technology, (b) customization of the semantics of modeling formalisms, and most importantly, (c) customization and incremental organizational learning about the analysis of interactions between modeling formalisms. We first discuss the user view of the language (textual and graphical syntax), then the logical foundations and finally its usage in customization and model analysis.

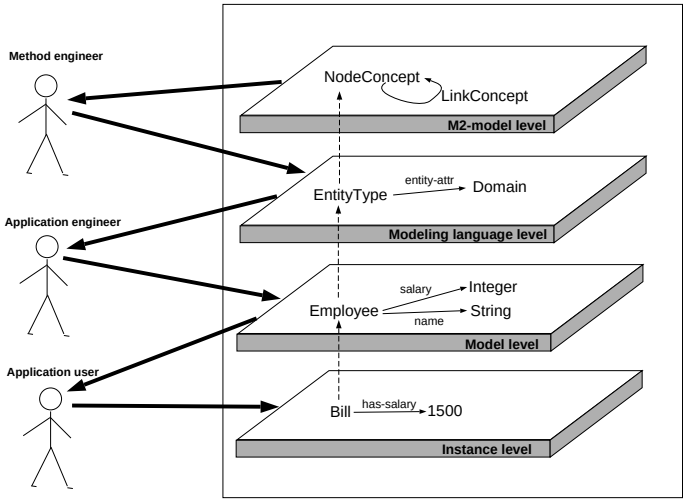

Fig. 2.1. IRDS and O-Telos

2.1 User View

The four IRDS levels discussed in the introduction define different user classes for O-Telos. Method engineers define a modeling language (here: ER) based on common principles (M2-classes `NodeConcept` and `LinkConcept` in the example). Application engineers learn such a modeling language (symbolized by the meta class `EntityType`) and develop a model (containing for example a class `Employee`). Finally, application users manipulate instance level objects that conform to the model.

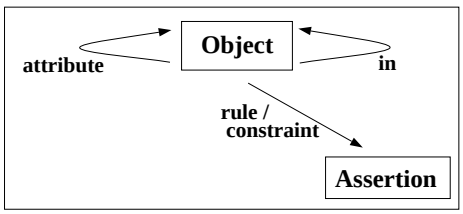

Fig. 2.2. O-Telos' builtin objects

A closer look at figure 2.1 reveals that any modeling facility supporting such an interlocked way-of-working requires at least three basic language concepts – one for self-standing labeled objects, a second one for labeled links between them, and the third one to express the instantiation relationship between the IRDS levels. In order to provide formal control over the usage of these base constructs, a fourth concept, that of a logical assertion, is also desirable.

As shown in figure 2.2, the kernel of the O-Telos language is just that. All other language facilities (such as generalization hierarchies, cardinality constraints, and so on) can be bootstrapped from this kernel.

In the *textual view* we group together all information for an object (e.g. `Employee`). The class (e.g. `EntityType`) of that object precedes the object name, the attributes of the object (e.g. `salary`) are sorted under *attribute categories* (e.g. `entity_attr`) which refer to the attribute definitions of the object's classes. Note that all objects, i.e. links and nodes, are instances of the builtin object `Object`.

```
Object EntityType with
  attribute
     entity-attr: Domain
end
```

```
EntityType Employee with
  entity-attr
     name   : String;
     salary: Integer
end
```

Besides inserting and modifying O-Telos objects (TELL function), the second main function of the server is the ASK facility. Queries are formulated like ordinary classes with a (membership) constraint [SNJ94]. They are recognized by the system via the keyword `QueryClass`. The query evaluator

computes the answers and establishes an intensional instantiation relationship between the query class and the answers.

The following example presents a query class `RichEmployees` computing all employees with salary greater than 120.000. We restrict the set of answers to the employees by defining the query class as a specialization of `Employee`. The attributes which should be part of the answer are specified as attributes of the query class. In the example we will get the `msalary` attribute for all computed employees. The constraint forms the membership condition, i.e. all only employees that satisfy this constraint become answers to the query class. For the example we require that the value of the `salary` attribute is greater than 120.000.

```
QueryClass RichEmployees isa Employee with
  attribute
      salary : Integer
  constraint
      c : $ salary > 120.000 $
end
```

Note that updates (TELL) and queries (ASK) may refer to any abstraction level. Thus, instance level objects are updated and queried in exactly the same way as the concepts of the modeling language level.

The ConceptBase user interface includes a customizable graph-browser. The base function is to display node objects like `Employee` and link objects like `Employee!salary`. The customization is done by assigning graphical types to nodes and links directly or via deductive rules. It is therefore possible to specify a certain graphical type to all instances of a specific object. An example of graphical customization will be given in section 3..

2.2 Logical Foundations of O-Telos

O-Telos is fully based on the framework of deductive databases, more precisely Datalog with stratified negation [CGT90]. It employs a single relation `P` to store nodes and links of a semantic network. *Nodes* are represented by self-referring objects `P(x,x,n,x)` stating that an object identified by `x` and labelled by `n` exists. An *attribute* labelled `a` of an object `x` having the attribute value `y` is written as `P(o,x,a,y)`. The attribute itself is regarded as a full-fledged object with identifier `o`. We distinguish two attribute labels with predefined interpretation: The fact that an object `x` is an *instance* of a class `c` is represented by an object `P(o,x,in,c)`. Moreover, the *specialization* relationship between two objects `c` and `d` is stored as an object `P(o,c,isa,d)` where `c` is sometimes called a *subclass* of its superclass `d`.

The P relation allows the representation of arbitrary semantic networks. It serves as the so-called *extensional database* in the deductive interpretation of O-Telos: all explicit information (e.g., a diagram) is stored as objects in

the P relation. It should be noted that instances and classes are uniformly represented as objects. Classes may be instances of objects themselves[2]. The ability of O-Telos to represent instances, classes, meta classes, M2-classes etc. uniformly as objects makes it a good framework to store information at different abstraction levels as presented in the subsequent sections.

The extensional database is accompanied by the so-called *intensional database*, i.e. a set of deductive rules and integrity constraints that are stored as attributes of objects. The rules and constraints are logical expressions that are evaluated against the extensional database. The formal interpretation of rules is based on a fixpoint semantics [CGT90] which precisely defines which facts can be derived from the database (extensional plus intensional part). Intuitively, the derivation follows the Modus Pones rule: if the condition A holds and we have a rule "A then B", then the fact B holds. Constraints are special rules of the form "if A does not hold then we have discovered an inconsistency". The object-oriented structure of O-Telos is defined on the simple P-relation via predefined rules and constraints included in any O-Telos database - the so-called O-Telos axioms.

```
forall o,x,c P(o,x,in,c) ==> (x in c)
```

If we explicitly state that `x` is an instance of `c` than the fact `(x in c)` holds.

```
forall o,x,c,d (x in c) and P(o,c,isa,d) ==> (x in d)
```

If `x` is an instance of a subclass, then it is also an instance of its superclasses.

```
forall p,c,m,d,o,x,l,y P(o,x,l,y) and P(p,c,m,d)
       and (o in p) ==> (x m/l y)
forall x,m,l,y (x m/l y) ==> (x m y)
```

The first rule derives an attribute predicate `(x m/l y)` whenever an attribute `o` is declared as an instance of another attribute `p` at the class level. The label `m` is called the *category* of the attribute `p`. The second rule omits the label of the instance level attribute.

Alltogether only 30 such rules were predefined in O-Telos [Jeu92]. The two important things to memorize are

- The single P relation is able to capture semantic networks ("nodes connected by links").
- Rules and constraints are used to fix the interpretation of abstractions like instantiation and specialization. These abstractions are predefined node and link types in the semantic network.

[2] If `x` is an instance of a class `c` and `c` is an instance of a class `mc`, then we refer to `mc` as a *metaclass* of `x`.

2.3 Conceptual Modeling Languages as Meta Models in O-Telos

The foundation of O-Telos just provides the facilities for representing graphs, plus to constrain and query them via logical conditions. In the following we show that this is enough for not only describing a large collection of conceptual modeling languages but also to relate them in a formal way.

O-Telos treats information at each abstraction level uniformly as objects. The fact that some object is an instance of a class at the upper level is represented as a (derived or stored) fact `(x in c)`. A meta class in O-Telos is simply a class `mc` which fulfills the condition `exists x,c (x in c) and (c in mc)`. Attributes are also full-fledged objects: attributes at a class level are the classes of the attributes at the instance level.

Constraints are employed to specify conditions on the instantiation of classes. Rules define information that is derived from explicit information. Note that constraints and rules can be defined at any abstraction level, even crossing several abstraction levels. For example, the instance inheritance rule above is applicable for objects at the model level as well as for objects at the M2-model level. We distinguish the following types of formulae according to the levels involved in the logical condition. As an example, we again use pieces of a formalization of the Entity-Relationship (ER) approach within O-Telos.

- *Model conditions.* Such formulae quantify over instances of classes defined at the model level. For example, there may be a class `Employee` at the model level with an attribute 'salary':

```
forall e,s (e in Employee) and (e salary s) ==> (s > 0)
```

- *Modeling language conditions.* Such formulae quantity over instances of meta classes. For example, the meta class `EntityType` could have a constraint that each instance (like `Employee`) must have at least one attribute (like `salary`):

```
forall c (c in EntityType) ==> exists d (d in Domain)
                                and (c entity_attr d)
```

- *M2-model conditions.* Here, the formulae talk about objects at the modeling language level. In our running example, we can think of the two M2-classes `NodeConcept` and `LinkConcept` that shall be used to define `EntityType` and `RelationshipType`. A M2-model condition could be that links connect nodes but not vice versa:

```
forall x,y (x connects y) ==>
 (x in LinkConcept) and (y in NodeConcept)
```

The reader should have noticed that there is no formal difference between those three kinds of formulae; they are just quantifying over objects at different abstraction levels. The uniform representation of O-Telos objects provides

this feature quasi for free. The above examples showed formulae quantifying over objects at the next lower level of abstraction (class to instance). It is also possible to express conditions spanning more than two IRDS levels. Such conditions are needed when the semantics of certain concept types (meta classes) can only be expressed in terms of the instances of the instances of the meta classes. As an example consider "key attributes" of entity types in the ER modeling language.

```
forall x,y,e,k,a,d,v  (x,y in e) and
       (e in EntityType) and P(k,e,a,d) and
       (k in Key) and (x a v) and (y a v)
        ==> (x = y)
```

The formula states that when two entities `x`,`y` of the same entity type `e` have the same value for the key attribute `a`, then they must be the same.

Such conditions are typical for formal interpretation of conceptual modeling languages. The interesting thing is that those conditions are expressible in the Datalog logic of O-Telos. Thereby, they can be added and evaluated to the (deductive) database at any time. This makes it possible to define specialized modeling languages just by storing appropriate meta classes with their axioms (rules and constraints) in the database. More examples of such formulae crossing multiple IRDS levels can be found in [JJ96].

3. Case Study: Design of a Customized Conceptual Modeling Environment

The following case study illustrates the management of conceptual models in the context of computer-support for an informal, teamwork-oriented analysis method used by a consulting company. Details and experiences can be found in [NJJ+96].

The consulting firm uses the analysis method PFR (Analysis of Presence and Future Requirements) for rapid, focused requirements capture in settings that alternate between team workshops and individual interviews:

1. In a two-day workshop, stakeholders define an initial *shared vision*. 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, drafts a redesigned business process, and identify the perspectives of some stakeholders as critical to success.
2. The perspectives identified as critical 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 or data flow models). The acquisition process is accompanied by a *cross-analysis* of the captured conceptual models 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 conceptual models 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.

Initially, standard modeling languages like Entity-Relationship diagrams were used both for describing the current procedures and the new (improved) procedures.

Problems with the standard tools emerged with respect to interpretation, extensibility, and analysis functionality. Regarding *interpretation*, customers complained that they wouldn't understand the difference between certain concepts of the modeling languages. For example, discussions emerged on whether a certain property of an entity would be a relationship or just an attribute. During these discussions, computer scientists would take the lead and the other participants would loose interest. Customers asked for a graphical method where one has just nodes and links.

Another issue mentioned was *extensibility*. The consulting company has developed its own approach to IT controlling where media was a central concept, i.e. the physical carrier of data like paper and floppy disk. Information on which data would be stored on which medium was important to decide how to improve the current workflow of the customer. Unfortunately, no CASE tool on the market fitted to these needs or could be easily adapted to it.

Finally, the *analysis capabilities* of standard packages were regarded as insufficient. Standard tools concentrate on syntactical correctness of the models and their interdependencies. However, the semantic correctness was seen as much more urgent. The following situation occured in a customer project: A complex data object (tax form) was modeled which contained a smaller data object (tax rate) as a part. A system function was provided to update the tax rate. In this application however, it was required that the numbers in the tax form are updated whenever the tax rate is changed. Since this dependency was detected only after implementation, major error correction costs were induced. As a consequence, the consulting company wanted to memorize this

pattern as a possible (not sufficient) cause for a semantic error in the system model.

3.1 Customizing ConceptBase

To tailor O-Telos to the standard PFR modeling languages the consulting firm first defined their syntax in O-Telos, as shown in figure 3.1.

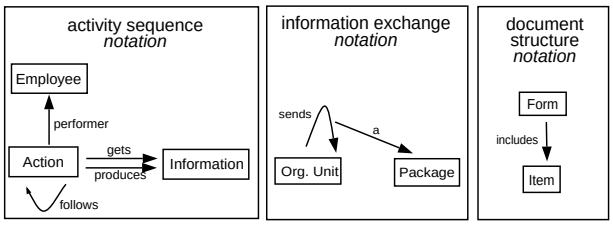

Fig. 3.1. Syntax of the standard PFR notations

The 'activity sequence' notation comprises the concept of an `Employee` who is the `performer` of an `Action`. The `Action gets` and `produces Information`. The `follows` relation describes dependencies between different `Actions`. The 'information exchange' notation captures `Organisational Units` which may `send a Package` to another unit. Finally, the 'document structure' notation comprises concepts to define a `Form` and the `Items` it `includes`.

The semantic properties of these notations are specified by integrity constraints and deductive rules. The PFR analysts required, e.g., that every information exchange between `Organisational Units` must be accompanied with the exchanged `Package`. The following integrity constraint expresses this requirement in a formal way:

```
forall s (s in OrgUnit!sends) exists p,a (p in Package) and
         (a in OrgUnit!sends!a) and From(a,s) and To(a,p).
```

Similar constraints specify the semantic properties of the modeling concepts of the other notations.

The semantic analysis of the individual conceptual models exploits the properties of the observed domain and the analysis goals of the specific project. The consulting firm specified the domain structure within a conceptual model on the M2-model level, shown in figure 3.2. The modeling language definitions in figure 3.1 form partial instances of this model which describes the corresponding perspective. It interrelates all three perspectives mentioned before. An `Agent supplies` another `Agent` with `the Medium`. This `Medium` may contain some `Data`. On this `Data` an `Agent performs` his `Activity`. The `Data` can be used as `input` or `output`. This model defines the extent of the analysis project: exactly the concepts mentioned in this model must be captured and modeled within the acquisition part of the project. It also reflects

some of the expected problems. The explicit distinction between the `Medium` and the `Data` it contains allows for the detection of optimizable workflows in the business process. Since the analysis goals may change from project to project, also this domain model may change to cover the actual problems to be investigated.

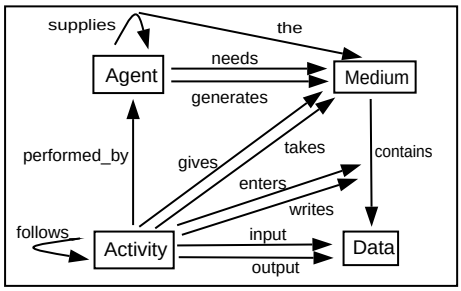

Fig. 3.2. A media-centered meta meta model for PFR

Beside the domain structure, the meta meta model contains the formalization of the analysis goals. They reflect the problems the analysis project is supposed to discover. Many customers of the consulting firm want to optimize their document flow. Therefore an analysis goal is to detect agents who get a document, but perform no activities on data contained on that document. Thanks to the formal semantics of O-Telos we are able to specify this analysis goal as a formal multi-level condition and to evaluate it on the contents of the object base. We use a special syntax to indicate multi-level literals: A literal of the form `(i [in] c)` describes an instantiation relationship between `i` and `c` that crosses multiple classification levels. A literal of the form `(a [m] b)` where `m` is an arbitrary label describes an attribute predicate that crosses multiple levels. In our case we use a label from the M2-model level to form a condition on the schema level.

```
forall supply,user,medium (supply [in] Agent!supplies) and
    (user [in] Agent) and (supply [to] user) and
    (medium [in] Medium) and (supply [the] medium)
     ==> exists info,action (info [in] Data) and
       (medium [contains] info) and (action [in] Activity)
       and (action [performed_by] user) and
       ((action [input] info) or (action [output] info))
```

In the example environment more than 80 standard analysis goals make semantic statements about single models, inter-relationships between multiple models and properties of the modeled business process. These analysis goals cannot be hard-coded because they may change from one project to another. Further experiences in applying the PFR method lead to the detection of further patterns of potential errors in business processes. These

patterns are then formulated as analysis goals to be available in following analysis projects. An example of such a pattern is the situation where an agent gets a document that contains only data that is already supplied to him by other media. This pattern does not always describe an error of the business process, but it is a hint for further investigation. It may indicate an unnecessary media supply which is subject for optimization. But it may also be an intended situation where the agent performs a comparison check of the same data located on different media.

The syntactic and semantic extension of ConceptBase is complemented by a graphical extension. A graphical type can either be specified for a specific object or for all the instances of an object.

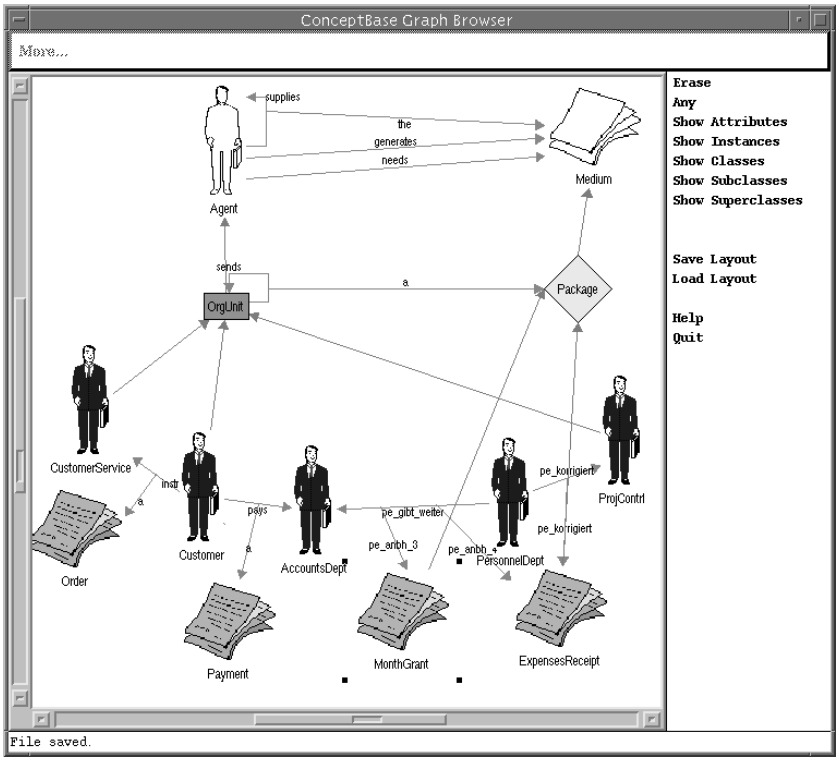

Fig. 3.3. The three levels within ConceptBase

Figure 3.3 presents a screendump of the ConceptBase graph browser. It shows a part of the three repository levels using the graphical types defined by the consulting firm. The part of the meta meta model defining the information exchange is shown on the top. The shape of a human is the graphical presentation of the object `Agent` and the shape of a set of papers of `Medium`. They used the shapes to indicate the abstract nature of these concepts. Below

these objects the notation of the corresponding conceptual models is shown. The `Organisational Unit` is presented as a rectangle and the `Package` as a diamond. On the bottom a small excerpt of the 'information exchange' model is given. For the modeled agents and documents they used the filled graphical types of the concepts of the meta meta model to indicate that these objects are more concrete.

4. Summary and Outlook

Conceptual modeling requires the use of multiple interdependent languages. Selecting the right collection of languages and focusing the analysis of their interactions is a not trivial task. For example, the mobile phone company Nokia claims to employ more than 150 different notations and/or methods in their software development processes. In such new application domains, standard languages may very well miss the modeling goal by distracting the modelers to details of notation instead to details of the domain to be modeled.

In O-Telos, as supported by the ConceptBase system, experts can define an adapted collection of languages via meta classes. The customized languages are interrelated via a meta meta model which encodes the overall modeling goals independently from details of the notation of the modeling languages.

Versions of ConceptBase have been distributed for use in research, teaching, and industrial development since the early 1990s. Currently, a few hundred groups worldwide use the system, a number of such efforts have resulted in spin-off products derived from the ConceptBase prototypes. The main applications have been in cooperation-intensive projects which we have here placed in contrast to notation-oriented standards such as UML or domain-oriented reference models as in the ARIS framework. Especially for the reference models, there is good reason to believe that this competitive situation should be turned into a cooperative one – a cooperative, customized, and goal-oriented modeling process should still be enabled to take advantage of external experiences as encoded in reference models. This attempt to bring goals, teamwork, and formal analysis into the customization process of component software strategies such as SAP or Baan is a major methodological goal of our ongoing research.

In order to support such methodological advances, some advances in the technical support by ConceptBase are also being investigated. The description in this paper corresponds roughly to version 4.1 of the system which has been distributed since 1996. In the following, we sketch some extensions which have been developed for integration into the next versions of the system.

Any extensions aim to preserve the decisive advantage of O-Telos, its firm basis on standard predicate logic with clear semantics. Its ability to uniformly represent instances, classes, meta classes, and attributes as objects makes it an ideal framework for meta data management and meta modeling.

Its implementation, ConceptBase, adds persistent storage of objects, a query evaluator, and a collection of graphical and frame-based tools.

In order to offer even more scalability in cooperative modeling tasks, the most important extension is the introduction of a concept of *modeling perspectives*, i.e. interacting modules, into the language such that models can be organized according to accepted principles of software architecture. In [Nis97, NJ97], the language M-Telos has been developed (and prototypically implemented) which is upward compatible with O-Telos and preserves the simple foundations based on Datalog$^{\neg}$.

A second important extension under development is a more active role the ConceptBase server can take with respect to its clients; an important special case is the transformation across notations (as opposed to just analysis queries). To preserve consistency, such transformations with materialized results should be incrementally maintainable over change. In [Sta96, SJ96], formalisms and algorithms to achieve incremental maintenance of materialized views not only inside the server, but also in external clients have been developed and implemented. The power of such algorithms and the user comfort are significantly enhanced if they are realized using mobile code that can move to the client without local installation effort. Starting from experiences with the CoDecide client that offers spreadsheet-like interfaces to the kind of data cubes used in data warehousing [GJJ97], a complete Java-based user environment is being developed.

Last not least, many cooperative modeling processes require inconsistency management not just for static logical interactions, but along possibly complex process chains. The current deductive approach only allows the analysis of process chains consisting of very few steps. Recently developed process reasoning techniques [BMR93] in a logical framework that is comparably simple to ours appear as a promising candidate for an integration into ConceptBase, without sacrificing its uniform framework and conceptual simplicity.

References

[Abe95] P. Abel. Description of the USU-PFR analysis method. Technical report, USU GmbH, Möglingen, 1995.

[Aug91] J.H. August. *Joint Application Design: The Group Session Approach to System Design*. Yourdon Press, Englewood Cliffs, 1991.

[BHS+97] P.A. Bernstein, K. Harry, P. Sanders, D. Shutt, and J. Zander. The microsoft repository. In *Proc. of the 23rd Intl. Conf. on Very Large Data Bases (VLDB)*, pages 3–12, Athens, Greece, August 1997.

[BMR93] A. Borgida, J. Mylopoulos, and R. Reiter. "... and nothing else changes": The frame problem in procedure specifications. In *Proc. of the Fifteenth Intl. Conf. on Software Engineering (ICSE-15)*, May 1993.

[CGT90] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer Verlag, 1990.

[Che89] P.B. Checkland. Soft systems methodology. In J. Rosenhead, editor, *Rational Analysis for a Problematic World*, pages 71–100. John Wiley & Sons, Chichester, 1989.

[CJMV95] P. Constantopoulos, M. Jarke, J. Mylopoulos, and Y. Vassiliou. The software information base: A server for reuse. *VLDB Journal*, 4(1):1–43, 1995.

[FS97] M. Fowler and K. Scott. *UML Destilled: Applying the Standard Object Modeling Language*. Addison-Wesley, 1997.

[GJJ97] M. Gebhardt, M. Jarke, and S. Jacobs. A toolkit for negotiation support interfaces to multi-dimensional data. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 348–356, May 1997.

[GMB94] S. Greenspan, J. Mylopoulos, and A. Borgida. On formal requirements modeling languages: RML revisited. In *Proc. of 16th Intl. Conf. on Software Engineering (ICSE)*, Sorrento, Italy, May 16-21 1994.

[IDS96] IDS Prof. Scheer GmbH, Saarbrücken. *ARIS-Toolset Manual V3.1*, 1996.

[ISO90] ISO/IEC International Standard. *Information Resource Dictionary System (IRDS) - Framework ISO/IEC 10027*, 1990.

[Jeu92] M.A. Jeusfeld. *Update Control in Deductive Object Bases*. PhD thesis, University of Passau (in German), 1992.

[JGJ+95] 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*, 4(2):167–192, March 1995.

[JJ96] M.A. Jeusfeld and M. Jarke. Enterprise integration by market-driven schema evolution. *Intl. Journal Concurrent Engineering Research and Applications (CERA)*, 4(3), September 1996.

[JMSV92] M. Jarke, J. Mylopoulos, J.W. Schmidt, and Y. Vassiliou. DAIDA: An environment for evolving information systems. *ACM Transactions on Information Systems*, 10(1):1–50, 1992.

[JR88] M. Jarke and T. Rose. Managing knowledge about information system evolution. In *Proc. of the SIGMOD Intl. Conf. on Management of Data*, pages 303–311. Chicago, Illinois, USA, June 1988.

[KK84] J.E. Kottemann and B.R. Konsynski. Dynamic metasystems for information systems development. In *Proc. of the 5th Intl. Conf. on Information Systems*, pages 187–204, Tucson, Arizona, November 1984.

[KLR96] S. Kelly, K. Lyytinen, and M. Ross. MetaEdit+: A fully configurable multi-user and multi-tool CASE and CAME environment. In P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, editors, *Proc of the 8th Intl. Conf. an Advanced Information Systems Engineering (CAiSE'96)*, pages 1–21, Heraklion, Creta, Griechenland, May 1996. Springer-Verlag, LNCS 1080.

[MBJK90] 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.

[MSTT94] N. Maiden, A. Sutcliffe, C. Taylor, and D. Till. A set of formal problem abstractions for reuse during requirements engineering. *Engineering of Information Systems*, 2(6):679–698, 1994.

[Nis97] H.W. Nissen. *Separation and Resolution of Multiple Perspectives in Conceptual Modeling*. PhD thesis, RWTH Aachen, Germany, (in German), 1997.

[NJ97] H.W. Nissen and M. Jarke. Goal-oriented inconsistency management in customizable modeling environments. Technical Report 97-12, RWTH Aachen, Aachener Informatik-Berichte, 1997.

[NJJ+96] H.W. Nissen, M.A. Jeusfeld, M. Jarke, G.V. Zemanek, and H. Huber. Managing multiple requirements perspectives with metamodels. *IEEE Software*, pages 37–47, March 1996.

[Non94] I. Nonaka. A dynamic theory of organizational knowledge creation. *Organization Science*, (1):14–37, 1994.

[Poh94] K. Pohl. The three dimensions of requirements engineering: A framework and its application. *Information Systems*, 19(3), 1994.

[Sch94] A.-W. Scheer. *Business Process Engineering*. Springer-Verlag, 1994.

[SJ96] M. Staudt and M. Jarke. Incremental maintenance of externally materialized views. In *Proc. of the 22nd Intl. Conf. on Very Large Data Bases (VLDB'96)*, pages 75–86, Bombay, India, September 1996.

[SNJ94] M. Staudt, H.W. Nissen, and M.A. Jeusfeld. Query by class, rule and concept. *Journal of Applied Intelligence*, 4(2):133–156, 1994.

[Sta96] M. Staudt. *View Management in Client-Server Systems*. PhD thesis, RWTH Aachen, (in German), 1996.

[You89] E. Yourdon. *Modern Structured Analysis*. Prentice Hall, Englewood Cliffs, New Jersey, 1989.