Reusing Domain Knowledge in Requirement Analysis

Christian Kop, Heinrich C. Mayr

Institut für Informatik
Universität Klagenfurt
Universitätsstraße 65-67
A-9020 Klagenfurt

e-mail: {kop | mayr}@ifi.uni-klu.ac.at

Abstract

This paper addresses the systematic reuse of domain specific information in the early phases of the information system life cycle, i.e., requirement analysis and, in part, conceptual design. In this context, compared to the usual approaches that concentrate on reusing software (documents) on the more technical levels, reuse receives a new facet: the enduser in his role of the requirements provider. He has to decide on what information is to be changed, reused (i.e., valid) or rejected. In order to be able to do so, he must be provided with the informations in question in a way that allows him a quick understanding and that sets him into a position to easily cope with the complexity that is inherent to descriptions of real application domains. For that purpose, an additional phase, called conceptual predesign, is introduced into the information system life cycle between requirement analysis and conceptual design. It comes with a glossary model, the vocabulary of which demands less abstraction abilities than semantic models as they are known from methods for conceptual design, resp. object-oriented analysis. It will be shown, how information reuse might be done and supported within such a framework.
1. Introduction

The (re)use of existing software (programs, design documents, data etc.) during the different phases of an information system development process is practiced since the early days of software construction and is investigated systematically since the beginning of the eighties: Subroutine packages, adaptive application systems, method base systems, type systems, schema reuse as well as reuse supporting architectures and process models are stations on this way. Actually, within the context of reengineering or, more generally, of the so-called ’re-techniques’ on the one hand and of object-oriented techniques on the other, reuse of any kind of information again is one of the central concepts under investigation. Within this paper we address the question of reusing informations about a given Universe of Discours (UoD) as input to the phase of Requirement Analysis. The sources of such informations can be preceding development processes, results of redocumentation projects etc. Reuse in that early phase comes with one substantial difference to reuse in later phases: the involvement of the requiring persons and future users into reuse decisions. I.e., information that potentially might be reused has to be presented in a way that it is not only understandable to system analysts, designers and developers but also to the so-called enduser. He is the one to whom it is to decide which information will be entered in the new design process, i.e. reused, which information has to be changed or rejected.

If one assumes that conventional semantic (data) models or models as used in the numerous OOA (Object-Oriented Analysis) techniques are a viable platform for the communication with endusers, a first approach to information reuse for requirement analysis could consist in the reuse of (parts of) given conceptual schemata. Notice within this context, that OOA is conceptual design [KKM93, Ma94] using semantic models for the description of static and dynamic aspects of a given UoD. As a rule, however, endusers are not willing or able to set on a level of abstraction that is induced by such models - and that has to be taken in order to reach the goals of conceptual design, namely completeness and inambiguity. Another obstacle is the variety of methods used for the different UoD aspects, e.g., entity relationship or object models, dataflow models, petri nets, state machines etc. It is one of the authors experiences, gained from many information system (IS) development projects in several business branches, that the employment of such models and techniques in cooperation with endusers leads to a rather naive use and, accordingly, to rough results that have to be completed by the analyst/designer without any chance for the enduser to judge about the validity of such completions. On the other hand, when using natural language for requirements collection, designers pay a lot of work for the mapping of natural language descriptions into notions of the conceptual model - with the more or less successful task to explain what they specified. To overcome some of these shortcomings, method developers
provide designers with some heuristics for the transformation of user requirements into conceptual schema expressions [CY91, Ja93, Ru91]. This may support the modeling process but does not solve the fundamental communication problem between endusers and designers.

For that reasons, we propose to introduce, between requirement analysis and conceptual design, a further phase, that we call conceptual predesign, into the IS life cycle (see figure 1): Requirement analysis should be carried out on an informal, natural language level for supplying as much informations and UoD facets as possible. Conceptual predesign then is to bridge the gap between this informal level and the abstract level of conceptual design. Conceptual predesign again is conceptual because of the fact that it uses a semantic model, i.e., a controlled vocabulary with explained (agreed) instead of defined semantics (e.g. by recurring to an implemented language). Thus, the intended lower level of abstraction is gained by the choice of the particular notions of the vocabulary and, to some extent, the concept of representation of predesign results. For the latter we use a glossary approach.

The paper is organised as follows: Section 2 introduces the most important notions of our vocabulary (semantic model) for conceptual predesign and its representation. Section 3 discusses the aspects of reuse within the framework of requirement analysis and conceptual predesign. In section 4 we then describe how glossaries (a conceptual preschema) are transformed into a conceptual schema. The paper is...
2. The Glossary Model

Developing a model for conceptual predesign, we tried to satisfy both, the demands of the designer and those of the enduser. The result is a model which is based on glossaries, a concept that was first suggested within the scope of the DATAID project [DD83, DD85a, DD85b]. It bases on the assumption that endusers are able to understand contents of glossaries more easily than other formalized information representations because they are used to work with similar objects, namely list and tables, most of time within their organizational enviroment. Also designers might accept glossaries because these provide a structured basis for the transformation to the conceptual level, a task that in addition may be automated to a large extent as will be seen later on. In order to allow for such a basis, the elements of the glossary model were choosen such that each information expressed in a conceptual schema could also be represented in the glossary schema and vice versa. This guarantees an upward compatibility for re-engineering activities. The danger of again introducing an inadequate level of abstraction by following that strategy was estimated small in comparison to the mentioned advantages.

To allow for a comprehensive description of a given UoD, five basic notions are suggested: **thing, connection, operation, event** and **constraint**. Note that these notions are used as usual in a generic sense, i.e., have instances (extensions) in a preschema, that again have instances (extensions) in the UoD. For each notion a glossary is introduced that proposes slots for attributes that characterize aspects of the notion in question. In addition a socalled **sentence glossary** is used as a repository for the natural language requirement statements and that is referenced by entries of the other glossaries. Each of these glossaries represents an important aspect of an UoD.

Static aspects are mainly captured by the thing and by the connection glossary:

A **thing** may be a natural or juridical person (e.g. customer, employee), a material or immaterial thing (product, car, contract) or an abstract concept (customer-area, country). The concept of "thing" differs from the ER concept "entity-type" [Ch76] in that it must, within its UoD, not be identifiable independently from other things nor has to have any properties. Hence, something like a "name" or a "number" of a customer is also a thing in our model. The descriptor "thing" was chosen, since other descriptors (object type, data, entity type etc.) are heavily overloaded by homonyms in the literature. The thing glossary (see figure 3 in the
appendix) contains besides of purely descriptive informations like internal identification, name and description other relevant informations like hints on synonyms, examples for instances, quantity declarations and value constraints. Quantity declarations describe how many instances of a thing may exist in a the UoD at least and/or at most. Value constraints allow the designer to describe how valid instances should look like. A series of other attributes is omitted here since it would beyond the scope of this paper. For a detailed insight see [Ko93, Ku92].

The next notion to explain is that of **connection**. What is its semantics? Things do not exist isolated one from another. They are connected to each other in statements like "Our customers have a customer number, a name, an address ... " or "Customers may buy several products". In the DATAID glossary model, this kind of information was distributed among the operation and the data glossary and, on the logical level, among entity and relationship forms. In contrast to that, we consider such information as independent and we want to handle it on the same level as information about things. Therefore we decided to introduce an extra glossary (see figure 4 in the appendix).

Although the notion of connection looks like that of relationship within in the entity-relationship model, there is a fundamental difference: connections relate arbitrary things and not only entity-like ones. Furthermore, connections may be things. A statement like "A customer can have one or more telephone numbers and a telephone number belongs to a customer", leads to a connection instead of an attribute as would be the case using an ER-model. The connection "buy" between customers and products may be in parallel modeled as a thing "purchase". This concept of connecting things relates closer to the NIAM approach [NH 89] or to the model of [Ke83] than to the classic ER-model.

To catch the functional and dynamical aspects of an UoD, two more glossaries (an operation glossary and an event glossary) are proposed. The **operation glossary** (see figure 5) addresses the functional aspects: An operation is a task which can be executed. Examples of such tasks are "read customer number", "take an inventory ", "compute the commission". To describe an operation, slots for an operation identifier, a name, a description and the following aspects are offered by the glossary: Operation structure, executing thing and employment location. As can be seen from the examples in figure 5 operations range from simple ones to complex tasks. A task can consists of simpler tasks and a simple task can be involved in more than one complex task. The operation structure documents such facts. The notion of executing thing expresses that an operation belongs to a thing (comparing to methods in object-oriented approaches) and can be executed by that thing. By employment location, the designer can specify, which person or department is responsible for the execution of the operation. In
other words, he can define the access rights of various users to the functionality of the specified system.

Together with the operation glossary the event glossary completes the dynamic part. The event glossary contains information about the preconditions of an events, the operations that are executed if the event happens, and the postconditions that hold when the event has taken place.

The last two glossaries contain auxiliary information. The constraint glossary serves to incorporate information which cannot be classified and structured according to the attributes of the foregoing glossaries. The sentence glossary is to document all natural language requirements the endusers stated. The connection between these requirements and the entries in the other (derived) glossaries is realized by a standard column "sentence reference" in each of the other glossaries.

3. Reusing glossary contents

The function of the glossaries as a basis for communication and as a glue between enduser and designer demands was already mentioned. This function, however, makes it possible, to use glossaries not only for the documentation of requirements but also for the reuse of already documented requirements as it is still made in the conceptual design phase [EW92, EW93, BFM93, SM93]. They act as containers of knowledge from which information can be drawn as a starting point for discussion.

The main question is, what kind of information can be reused? A schema of an UoD as a whole will certainly be something that is individual. But pieces of that schema might be invariant within a given application domain and thus being drawn into the new schema.

Let us assume that things like "person", "manager", "employee", "customer", "representative", "name", "customer-number" are defined in a UoD X and that they are invariant. Then the designer can show these things (i.e., their glossary entries) to the enduser and ask him if he agree that they are part of the new schema Y. For a further example let us assume, that the designer has made the experience, that in other schemas the numbers of maximum customers (this information is presented by the column quantity description) ranges between 100.000 and 150.000. Then he may use this range as a first clue for the user.

In the same way we might reuse connections and related information as well as information
of the other glossaries.

Let us look to the above mentioned things. The connections which represent that "a customer is a person", "manager is a person", "an employee is a person" and "a representative is a person" may also be invariant if such connections are specified in several other UoD’s. Connections like "a person has a name" and "a customer number identifies a customer" or "customers buy products" may be also a basis for discussion if several others UoD’s use an equivalent kind of information. For the specification of the contents of the operation glossary, it might be important to know which operations exist, the operation structure of these operations, just as information about the employment location of an operation. Reusable kinds of information in the event glossary are again the operations involved as well as general specifications of the pre- and postconditions. As well information in the constraint and sentence glossaries may be subject of reuse.

Which kinds of information are reused and what detail this is done is within the responsibility of the designer and his requirements providers. In practice, only those aspects (column contents) will be considered that are understood by the enduser.

Reusing glossary contents requires further considerations: It must be possible for the designer to distinguish between reusable and other information without having to search for reusable information in all UoD’s of a certain context. I.e., he should be provided with a derived set of reusable information.

To do so, all UoD’s which have some analogies (that means they belong to the same context) are related to so-called UoD areas. The analogies should be conform to glossary informations like the same things, the same connections, the same operations and so on. In each UoD-area there exists one artificial UoD called, e.g., "knowledge-base". Invariant information are entered into the glossaries of that UoD. The advantage of such an UoD is that the media for invariant information are again glossaries. A separate person (administrator) should handle the administration of the UoD-areas and the artificial UoD’s. This person defines UoD-areas and decides which UoD should be related to which area. He can either edit the glossaries of the artificial UoD and put the information directly into the glossaries or he can retrieve and select them from existing glossaries of normal UoD’s of the same area and then copy them to the corresponding glossaries of the artificial UoD.

If a designer starts to specify a new UoD within a given UoD area then he can first copy the invariant information to the empty glossaries. These information are now the starting point of discussions with the enduser.
4. The process model

The process model is divided into an analysis and a transformation phase. In the analysis phase the glossaries are filled with information. Then these information are checked whether they are consistent or not. In the transformation phase a stepwise transformation of the glossary contents to conceptual static and dynamic schemas is executed. Checking and transforming contents should be supported semi-automatically by a tool.

The fact that the designer should offer the customer some reusable information was taken into account in the process model (see figure 2). The first step "initialization" is made for this purpose. In this step, the selection of the invariant parts from the glossaries of the artificial uoD is carried out. During the "information collection" the designer corrects and adapts available information and collects new requirements. If the designer and the enduser reach an agreement, that they have specified the UoD sufficiently, then the collection is finished for the moment and the glossary contents are checked whether they are consistent and complete (important columns filled, correct references, no cycles within the operation structures and special connections etc.). The designer should then get a list of inconsistent information. He has to decide what to do. Normally he will ask the enduser for further informations. With the answers he then tries to correct the inconsistencies. If no more inconsistencies are detected, the next step begins. From this moment a change in the process activities takes place. The tool starts with the transformation. It tries to transform the static contents into EER+-schemas and it transforms the dynamic contents into petri-nets.

The later transformation starts with an identification and a connection of common
preconditions of distinct events. Then a identification and a connection of common postconditions follows. At last internal matches (events which act as pre-and postcondition) are edited.

The static part of transformation is subdivided into three steps. The main suppliers of information are the thing and the connection glossary. In the first of the three steps (concept relation), the tool tries to relate things to concepts of the conceptual model (entity-type, attribute, domain). Rules are applied to determine which thing belongs to which concept of the conceptual model. We distinguish between several kind of rules. First they are divided into rules which relate a thing to the concept of "entity-type" and rules which relate a thing to the concept "attribute" and "domain". They are further classified according to their kind of determination (direct rules, indirect rules) and according to the strongness of determination (directions, proposals). Direct determination means, that a determination follows from information in the glossaries. Indirect determination means, that the tool can only relate a thing to a concept, if that thing is associated to another thing in the connection glossary and A was already related to a concept. A rule which can only relate one concept to a thing is called a directive. In such a case, other concepts related to the thing would cause a incorrect conceptual model. On the other hand, if a rule relates a concept to a thing because it seems to be more likely than other concepts, then this rule is called a proposal. A proposal it thus a weaker kind of rule. The concept relation step is based on simple heuristics. It considers carefully the collected rules for each thing. Then it decides which concept should be related to the thing or it delegates the decision to the designer when it is unable to find a relation.

In the next phase (transformation and restructuring) the tool tries to transform connections to relationships or attributes. This is possible since things are already related to concepts entity-type, attribute, or domain. For example if we have an connections where both participating things are entity-types, then this connection is a relationship. If one thing is an entity-type and the other participating thing is a domain, then the connection is modeled by an attribute. Some association will not be transformed directly and easily. For example, if we have related one participating thing to the concept "entity-type" and the other participating thing to the concept "attribute" (which is possible and legal during the concept relation step) then the tool restructures this connection with the help of the designer.

In the last step (completion) given entity-types are completed. That means, that the tool checks each entity-type if he has attributes and is identified. A warning is given to the user if this is not the case.

The result of all named steps of the static part is an initial EER+schema of the UoD. The
result of the dynamic part is an initial petri-net schema.

5. Conclusion

We introduced in this paper a new intermediate phase, called conceptual predesign into the IS life cycle with a model that is based on glossaries which register the relevant aspects of a given UoD. Further we explained that and how glossary contents can be reused. The stated assumption and explanations of the conceptual predesign should seen as ideas to start and propagate a discussion about new ways of analyzing and collecting requirements in general and glossaries as a requirements collection model in particular. At present parts of a prototype implementation for the conceptual predesign model is realized. The parts support the registering of requirements into the glossaries and supports also a piece of the transformation process. The next step will consist in the investigation of the question, to what extent the glossary model may be used as an 'interlingua' between natural language and conceptual models, so that natural language requirement specifications may be translated automatically into glossary entries and vice versa.

6. References

[BFM93] Bellizona, R.; Fugini, M. G.; de Mey, V.: Reuse of Specifications and Designs in a Development Information System. In [PRP93], pp. 79-98.


