An approach for systematic interface design from knowledge models

Taisa C. Novello*, Mara Abel*, Marcelo Pimenta*, Jesualdo Tomás Fernández-Breis**, Rodrigo Martínez-Béjar**

`novello@inf.ufrgs.br, marabel@inf.ufrgs.br, mpimenta@inf.ufrgs.br, jfernand@dif.um.es, rodrigo@dif.um.es`

*Instituto de Informática - Universidade Federal do Rio Grande do Sul Porto Alegre, Brazil
**Facultad de Informática, Universidad de Murcia, 30071 Espinardo, Murcia, Spain

Abstract. In this paper, an ontology-based approach for specifying interaction objects to build knowledge-based systems interfaces is presented. The ontology mapping enables the design of effective user interfaces for knowledge-based systems (KBS), according to the principle of structure-preserving design. This mapping from ontological objects onto interface issues is performed according to their role in the domain. For this purpose, the CommonKADS communication model has been extended in this work to include the definition of interface elements associated with domain knowledge primitives. A geological expert system has been the instrument for validating the approach.

1. Introduction

The motivation for this work comes from our experience in a project consisting in the development and maintenance of a real knowledge-based system (KBS) in geology. In this project, we had to face the problem of having to adapt already implemented knowledge models to user-understandable interfaces. In this work, we propose a systematic method to do this in the context of CommonKADS KBSs.

The classical software engineering lifecycle does not assume user interface design as a core activity. Moreover, traditional software development models do not clearly identify a role for Human Computer interaction (HCI) at any stage. User interface concerns are 'mixed in' with wider development activities. This may result in either HCI being ignored or it being relegated to an afterthought during the later stages of design. In either case, consequences can be disastrous because the later 'usability' issues are introduced in the development cycle, the higher the cost will be. In fact, since the early 1980's, there are several mechanisms or techniques that can be used to introduce HCI into the software development lifecycle like methods, guidelines and tools. For all of them, there are many steps to be performed before generating an initial user interface prototype: selection of interaction style, definition of presentation units, selection of interaction objects, etc. In fact, designing user interfaces involves the selection of preferred options from a number of competing alternatives. In this paper, we particularly analyse the possibility of having little or no support in selecting appropriate interaction objects for a specific task. Although there is an evident progress in providing methodological answers to some interface issues, in contrast, very little attention has been paid to the definition of complete approaches for automatically deriving interface components from application models.

KBSs dealing with non trivial information and following unpredictable steps of interaction may include additional difficulties in the already hard task of user interface definition. The typical dialogue between user and system involves some steps that must be performed using interface facilities provided by the system. It is developed following the knowledge model objectives, which are defined by the tasks. A task defines the specific information to collect from the user and the way the system will interact to display the partial or final inference results. This is usually not a linear input, transformation or output flow. On the contrary, this involves tasks sequences that start transactions, which are responsible for requiring and consuming information. Consequently, methodologies are frequently focussed on supporting the definition and control of the system dialog instead of interface design aspects.

Creating an initial prototype of the user interface from scratch is a handcraft task. In classical cyclic user interface developments, this first prototype passes through many sessions of user evaluation and further revisions until becoming accepted. Even so, producing such prototype is recognised as the most difficult stage in the user interface designing activity for a new system. However, we think that the relationship between ontological elements and interface issues can be used to assist in the definition of the interface objects and in the generation of the first version of a user interface in a KBS project.

A domain ontology describes the domain concepts and the underlying conceptual organisation ([7];[8]), which typically is an abstraction of the structure and behaviour of some particular domain, defined as the formal specification of a shared conceptualization [9]. Ontological engineering and knowledge acquisition tools are used for designing user-interfaces and for building ontologies in many contexts, for instance, in an interface system for an oil refinery plant [12]. We believe that any human interaction within a certain domain should be done by means of the objects recognized into the ontology and reflecting the expected behaviour. This principle, already explored through the structure-preserving design proposed in CommonKADS [14] can also be applied in the definition and choice of issues for user interaction.

This work shows the possibilities of modelling and designing interface components, according to the structure-preserving design principle.

2. CommonKADS

CommonKADS is the most well known software engineering approach for KBS development and it has become a full methodology supporting all KBS development aspects. This goal is reached by following a few basic principles [14]. The first principle approaches knowledge engineering as a *modelling activity*, in opposition to previous approaches in which knowledge should be *extracted* from the expert's mind. The knowledge-level principle states that, in knowledge modelling, it is necessary to concentrate first on conceptual knowledge structures, postponing programming details. The third principle recognises the stable structure of knowledge, so that specific knowledge types and roles can be distinguished. Finally, a software project that deals with knowledge cannot be developed in the traditional waterfall model. It must be managed by learning from its own experience, in a controlled spiral cycle, supported by the model.

Moreover, in CommonKADS, the system is developed as a suite of models which allows defining formally each aspect of the software development in a separate way. The *organisational model*, *task model* and *agent model* allow for the comprehension of the organizational context in which the knowledge intensive task is inserted in terms of benefits, costs and impacts. By their side, the *knowledge model* and the *communication model* deal with the nature and structure of the domain knowledge involved in the task and also how this knowledge and the new one produced by the system are transmitted to every producer or consumer agent (people or system). The whole requirements set collected in every suite mo del component is considered in the development of the *design model*, which provides the technical specification for system development.

The knowledge model is the core of a knowledge-based system, so that the concepts and the structure represented can be id entified through components from the implementation level. In fact, this is strongly enforced by the *structure-preserving design* principle, which asserts that the information content and the structure present in the knowledge model should be preserved in the system components. The design process is seen as the process of *adding* implementation details to a previously defined analysis model. Design specification fills the details that were left open during the analysis stages such as representational formats, computational methods of computing inferences, dynamic data storage or the communication media. The advantage of structure-preserving design, according to the CommonKADS authors, is that the knowledge level and the communication model act as high-level implementation documentation, keeping the relation with elements in the code which should be adapted in case of changing the model specification.

The main goal of CommonKADS communication model is to specify information exchange between tasks that are carried out by different agents. The *transaction* is the building block of dialogue between agents, and it is associated with information or knowledge exchanged by two agents. Therefore, the communication model, as it is specified in CommonKADS, describes the communication plan among system components and information content, but it is not concerned with design aspects or interface issues.

3. Extending the CommonKADS Communication Model with Interface Objects

3.1 Our Knowledge Model

Knowledge models in CommonKADS specify the knowledge and reasoning requirements of a knowledge-based system, and allow clarifying data and knowledge structures required for a knowledge intensive task. They are comprised of three distinct components: (1) domain knowledge specifies the declarative domain knowledge or how the concepts or objects are organized in that particular domain, expressed through the ontology of the domain (e.g., rocks, minerals), and, also, what we know related to some particular task (i.e, rules , restrictions and so on); (2) inference knowledge describes the basic inference steps that are usually followed in the domain to solve tasks (e.g., select, match); (3) task knowledge has information about the problem to be solved in the domain and how it can be decomposed to make it clear what the steps and information necessary for the solution are.

The knowledge model in our system, namely, *PetroGrapher* [2], is represented by a domain ontology. An ontology is commonly viewed as a specification of a domain knowledge conceptualization [15]. In this work, the ontological model is comprised of concepts, attributes, values and relations between concepts. The domain ontology describes types of concepts and attributes, which will compound later on the information items.
In the last years, ontologies have become more and more important for representing knowledge. Consequently, different tools and frameworks have appeared with different purposes. Some frameworks are devoted to facilitate the design of ontologies (i.e., Methontology [6]). On the other hand, there are different systems to specify ontologies such as OntoEdit (http://www.ontoprise.de). However, the underlying ontological model in most of such systems and frameworks does not offer many modelling facilities to the user. In this way, most of them only include the taxonomic relation. A survey of different ontological tools is shown in [5].This survey shows that no one of the currently available tools fully satisfies the needs of ontology developers. There is another research trend that attempts to integrate the knowledge contained in different ontologies (i.e, [13]).
With all, we developed our own ontology (knowledge) editor, so that the resulting ontology was a result of the knowledge acquisition process, in which the collection of *cases* in a geological domain played a significant role. A mereology was obtained from such process so filling in the current gap which lacks ontologies different from taxonomies exclusively. The mereology reflects the structure extracted in the process of abstraction of several sets of *cases* [3]. The part-of relation is associated to the unique identification of each compound object, which is displayed in each component part. Moreover, properties such as transitivity and non-symmetry were considered in the implementation. The taxonomic relation plays a secondary role in the domain identifying different classes of rocks and minerals recognized by the system. Other particular kinds of relationships can be recognized in the domain ontology: (1)

restrict, defining which concepts can be seen as values of attributes of other concepts; and (2) i*mplicate*, stating logical relations of evidences and conclusions in the domain, it being useful to support the inference process. Relations are defined through its name and its cardinality [1]. Conceptual attributes have types defined in the knowledge model, reflecting their behaviour in the domain. Some default format is also included in the model (i.e., how attributes can be viewed). Typical attribute types are: *real*, a numerical value, such as the depth of well log; *numerical*, such as the percentages of minerals of a rock; ; *strings* of characters with defined sizes, for names of units, places, etc; *symbolic*, values previously defined in the model nomenclature. The symbolic values of attributes can be defined as being unique (value domain defined as one-of) or multi-valued (defined as list-of). Also, the value can be informal, meaning that it does not receive any semantic or inference treatment.

3.2 The Communication Model

The communication model specifies the information exchange between agents (systems or people) carrying out tasks in the domain. The transaction expresses the communication model and it is defined in terms of *agents* involved in the task, *information items*, *message specifications* and *control* over messages. While the agent refers to the KBS and user, this transaction defines the interface communication plan with the user. The communication model proposed here can also deal with aspects of user interface design, besides representing the information about which information is exchanged among agents and control aspects. We propose how knowledge and communication models need to be improved in order to achieve this goal, and to demonstrate the use of the new conception through a real KBS application in the Petrography of oil-reservoir rocks domain.
In Table 1, a modification over the CommonKADS Communication Model, described in [14],to deal with interaction issues, is introduced. The extension of the model follows some premises about the KBS behavior and domain ontology:

1. Any concept, attribute and relation that is considered in the system, even if it is generated by inference, are specified in the ontology. Moreover, interaction messages are also specified as domain expressions or statements in the knowledge model.
2. Any system interaction will refer to some knowledge specified in the ontology or stored in the knowledge base.
3. Each knowledge type defined in the knowledge model has a particular set of properties which influences the choice of the interface object. These properties express the domain behaviour and should be explicitly defined along with the object.

From these principles, we can formalize the influence of the elements in the knowledge model over the definition of interface objects and their behaviour. Therefore, the selection of the interface object will be done according to: (1) the sender and receiver in the transaction definition; (2) the type of concept, concept-attribute or domain expression that is referred in the information item, and specified in the ontology; (3) the syntactic form of the information, included in the knowledge model; (4) the item's role (i.e., input, output or orientation message). Message specifications are defined according to the treatment provided by CommonKADS. However, as messages produced by the system are represented in the knowledge model, they can be treated as information items in the Communication Model. Table 1 presents a part of communication model in the reservoir rock application.

Table 1. An example of transaction in the PetroGrapher system.

Communication Model	Information Exchange Specification
Agents involved	1. Sender: User
	2. Receiver: KBS
Information Items	1. Object: Concept Microscopy
	2. Role: Core Object
	3. Medium: Presentation Unit, Microscopic Window
Message Specifications	1. Communication Type: Ask for the rock grain size of the sample.
	2. Content: "Enter the rock grain size "
	3. Reference: Siliciclastic knowledge Base
Control	Control: keep the message until the user provides information or change the PU.

4. The Mapping Between Ontology Elements and Interface Objects

In order to complete the information about transactions, it is necessary to define which medium needs to be associated with each information item in the information exchange. This is done according to the type of ontology category (concepts, attributes, expressions) that is associated with the transaction. Since transaction information items are defined according to the ontology content, the communication model does not need to define any syntactic format associated with the information items. The cardinality of the relationship influences the presentation of attributes [4]. A one-to-one relationship between two entities has no additional effect on the presentation, because they are already grouped according to their logical relation. In a one-to-many relationship, an aggregation view is needed to show more than one relationship instance. This is demonstrated in our model by allowing the user to navigate or visualise all interfaces with the same sample Unique Identification. The relationship between interface objects and primitives in the ontology is described in Table 2, in accordance with the syntactic characteristics of each concept and its attributes.
The first object, the Concept, which is defined through its attributes into the ontology, is mapped onto Presentation Units (PUs), namely, an input/display world (interface) decomposed into windows (which do not need to be all present at the same time). The PU is built on top of Concrete Interactive of Objects (CIO), which represent attributes, graphics and images of concepts. This mapping can be illustrated through the concept *Microscopic*, which describes some set of features recognisable in a thin section of a reservoir rock, and is represented in the knowledge model as the attributes of the Microscopic Concept. The way this concept and its attributes are materialized

into the user interface is described in Table 3, and the resulting interface is presented in Figure 1.

Table 2. Mapping between CommonKADS primitives and interface objects.

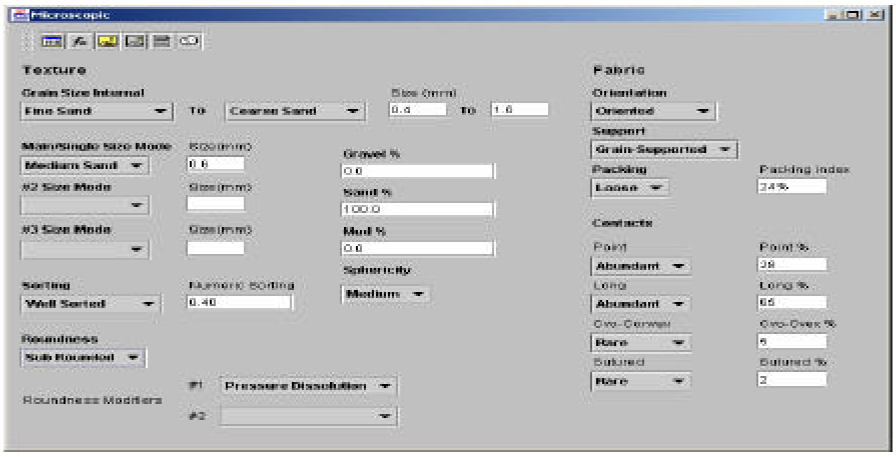
Ontology(Knowledge level)	Implementation Level
Concept	Presentation Unit
Attribute	Edit box
Value	Object
Graphic Objects (graphs, triangles)	Graphic Objects

Table 3. An example of mapping in the PetroGrapher system.

Ontology	Implementation Level
Concept: Microscopic	Presentation Unit MicroscopicWindow
Attributes: Gravel: real, ranged [0.0 – 100.00]	Edit Box
Values: GrainSize: string(20), one of [gravel, very coarse sand, coarse sand, medium sand, fine sand, very fine sand, silt, clay]	List of values option is represented in the Combination Box.

The concept "microscopic" (as a PU), and its associated attributes and values are shown in Figure 1.

Fig. 1. Presentation unit for the concept *Microscopic* and its attributes.


Microscopic

Texture
Grain Size Interval: Fine Sand TO Coarse Sand
Size (mm): 0.4 TO 1.6
Main/Single Size Mode: Medium Sand
Size(mm): 0.6
#2 Size Mode:
Size(mm):
#3 Size Mode:
Size(mm):
Gravel %: 0.0
Sand %: 100.0
Mud %: 0.0
Sphericity: Medium
Sorting: Well Sorted
Numeric Sorting: 0.40
Roundness: Sub Rounded
Roundness Modifiers
#1: Pressure Dissolution
#2:

Fabric
Orientation: Oriented
Support: Grain-Supported
Packing: Loose
Packing index: 24%
Contacts
Point: Abundant
Point %: 28
Long: Abundant
Long %: 65
Cvo-Convex: Rare
Cvo-Cvex %: 5
Sutured: Rare
Sutured %: 2

The positional association between a concept and its attributes is sometimes not so obvious. The module PointCounter, which is the most complex module of PetroGrapher system, shows some problems related with this. The PointCounter interface was developed to allow the user to select, visualise and qualify a large set of distinct (because they belong to different classes) objects of the ontology (minerals). Each mineral, as a concept, is displayed in a PU. However, the large amount of elements in the screen and the big number of properties to be defined made the interface design task complex to perform. Four full versions of the software were developed and systematically refused by the testing group. The software failed in being considered as having "natural usage" due to long paths to achieve the goal. Figure 2 is a screen snapshot of the current version of the system, which was approved by the testing group (including the reservoir rock interpretation expert, four of his assistants and five professional petrographers). In the user interface generation, visual elements are mapped onto appropriate interaction objects by means of selection rules which take into account properties of attributes such as type, range and number of values.
The user interface should support the fulfillment of a user's tasks by providing easy to use and to learn ways for system interactions. Producing an easy interface consists in mapping the domain knowledge structure (which is familiar and natural for the user) into interface elements. Moreover, it is also essential providing an appropriate grouping and distribution of elements in the PU, in such way that these elements can be coherently visualised. Attributes belonging to the same concept or relation are implicitly grouped to reflect their association and they are shown or not in a view depending on the context and the task goal.

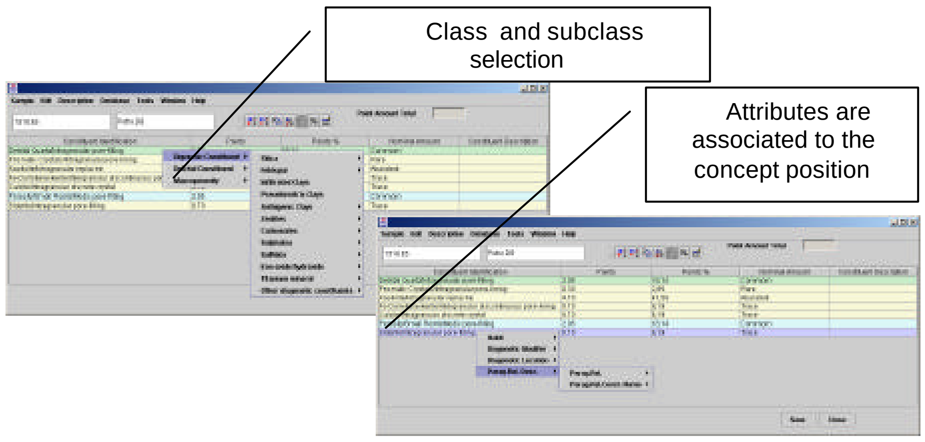

Class and subclass selection
Attributes are associated to the concept position

Fig. 2. The current version of the Point Counter interface.

5 Conclusion

Our experience in mapping knowledge models to interface issues, so that the system keeps user-orientation, lead us to the main objective of this paper, that is, to design a general model applicable to CommonKADS-like KBSs at least.

When producing an interface design from ontological definitions in the knowledge model, two main difficulties can be recognized. First, managing different kinds of concepts or relationships, with diverse semantic meaning that should be represented in the interface through distinct interface elements. Second, defining how to group and place these elements into the interface, since the spatial relation among attributes or concepts may not be clear enough (from users' perspectives) in the knowledge model.

The approach presented here for mapping ontology elements onto interface objects has shown to be a useful strategy to increase interface development efficiency. In our project, the strategy decreased the number of sections to approve a new interface model, and reduced the rework of defining new interface models. When developing an interface from scratch, it used to be necessary 4-5 interaction sessions with the test group to approve the new software interface module. Here, a module of medium complexity is considered, with less than 5 concepts and less than 30 attributes involved in the interface. Complex interfaces objects such as Point Counter, with 130 objects to be selected and qualified, can take 10 sessions and 3 interface versions to be approved or even never be fully approved. It became clear that developing the interface under the user point of view provides better familiarity and satisfaction to the user.

Designing interfaces by using the domain ontology reinforces the principle of structure-preserving design (i.e., the content and domain structure elicited in the acquisition phase are maintained and can be easily recognized through each level of KBS development, from the analysis stage in the knowledge level to the implementation level). The advantages in terms of modularity and maintenance are evident:

1. There is a standard way to develop and maintain user interfaces.
2. It is possible to reuse user interface design, minimising efforts in designing new interfaces in the same domain (or in any other domain with similar domain ontology) from scratch.
3. Any expansion in the knowledge base, even if it includes new system functionalities, is easily absorbed by the system designer.
4. The system has an homogeneous way of dealing with distinct kind of information: concepts in the ontology, interaction objects and inferred objects.
5. Including or excluding elements of the domain can be reflected in the interface without side effects.

The approach presented here can help KBS developers in the definition approval of some interface standards in the context of a software project. Provided that interface design is one of the most complex and time-consuming activities in the software development process, this approach can reduce the overall resource applied in a KBS project.

Acknowledgements

This project is supported by FINEP/CTPETRO Brazilian program. The fourth author is supported by the Séneca Foundation, Coordination Centre for Research, through the Séneca Program(FPI).

References

[1] Abel, M. (2001) Estudo da pericia em petrografia sedimentar e sua importância para a engenharia de conhecimento, Doctoral Thesis, in Programa de PG em Computação/UFRGS, 239 p., Porto Alegre

[2] Abel, M., Castilho, J.M.V., Campbell, J. (1998) Analysis of expertise for implementing geological expert systems. World Conference in Expert Systems, Mexico City, Mexico.

[3] Abel, M., Reategui, E.B., Castilho, J.M.V. (1996) Using case-based reasoning in a system that supports petrographic analysis. In B. Braunschweig and B.Bremdal (Eds), Artificial Intelligence in Petroleum Industry, 159-172, Paris, France.

[4] Bullinger, H.J., Fähnrich, K.P., Weisbecker, A. (1996) GENIUS: Generating Software ergonomic User Interfaces. International Journal of Human-Computer Studies, 115-144.

[5] Duineveld, A.J., Stoter, R., Weiden, M.R., Kenepa, B., & Benjamins, V.R. (2000). WonderTools? A comparative study of ontological engineering tools. *International Journal of Human-Computer Studies 52:1111-1133*.

[6] Fernández, M., Gómez-Pérez, A., Pazos, J., & Pazos, A. (1999) Building a chemical ontology using methontology and the ontology desing environment. *IEEE Intelligent Systems, 37-46*.

[7] Fernández-Breis, J.T., Martínez-Béjar, R. (2000) A Cooperative Tool for Facilitating Knowledge Management. Expert Systems with Applications 18(4)_315-330.

[8] Gómez-Pérez, A., Benjamins, V.R. (1999) Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods, in International Joint Conference on Artificial Intelligence, Workshop on Ontologies and Problem-Solving Methods, Stockholm, Sweden.

[9] Gruber, T.R. (1993) A translation approach to portable ontology specifications. Knowledge Acquisition 5:199-220.

[10] Martínez-Béjar, R., Ibañez-Cruz, F., Compton, P., Fernández-Breis, J.T., De las Heras-González, M. (2001) Integrating Ripple Down Rules with Ontologies in an Oncology Domain. Lecture Notes in Computer Science 2101:324-327.

[11] Martínez-Béjar, R., Ibañez-Cruz, F., Compton, P., Mihn Cao, T. (2001) An easy maintenance, reusable approach for building Knowledge-Based Systems. Expert Systems with Applications 20(2):153-162.

[12] Mizoguchi, R., Kozaki, K., Sano, T., Kitamura, Y. (2000). Construction and Deployment of a Plant Ontology. In Proceedings of European Knowledge Acquisition Workshop, 113-128.

[13] Pinto, H.S., & Martins, J.P. (2001). Ontology Integration: How to perform the Process. In *Proceedings of International Joint Conference on Artificial Intelligence*, Seattle, Washington, USA.

[14] Schreiber, G. (Editor) (1999) Knowledge Engineering and Management: The CommonKADS Methodology. Bradford Books, London, England.

[15] Van Heijst,G. Schreiber, A.T. Wielinga, B.J. (1997) Using explicit ontologies in KBS development'. International Journal of Human-Computer Studies, **45**, 183-292.