HyperADD: an Incremental and Hybrid Approach for Knowledge Acquisition on Active Design Documents

Sandro Coimbra Assis  
CDSV  
Xerox do Brasil  
Vitória, Espírito Santo, Brasil,  
29000  
sandro.assis@bra.xerox.com

Flávio Miguel Varejão  
Departamento de Informática  
Universidade Federal do Espírito Santo  
Vitória, Espírito Santo, Brasil,  
29000  
fvarejao@inf.ufes.br

Ana Cristina Garcia  
Departamento de Ciência da Computação  
Universidade Federal Fluminense  
Niterói, Rio de Janeiro, Brasil,  
22000  
bicharra@dcc.uff.br

Abstract

Many knowledge acquisition approaches have been proposed to minimize the cognitive effort in knowledge-based system construction. The knowledge acquisition approach proposed here put together the user interaction benefits of semiformal modeling and the computational benefits of formal modeling. Our approach defines a method where the designer first constructs a semiformal model of the design process and then formalizes it to allow computational interpretation.

Introduction

Anyone who has ever developed a knowledge-based system (KBS) agrees that the knowledge acquisition and formalization process is not an easy activity. Usually, acquiring knowledge is hard because the agent who possesses the knowledge is not able to express it in an accurate and complete way. Formalizing knowledge is also hard because it requires a great cognitive effort by the agent responsible for the formalization, since there is an expressive gap between the abstract language used by the knowledge owner and the formal language required by the computational agent.

For avoiding or reducing these problems, some knowledge acquisition approaches have been proposed to minimize the cognitive effort in the KBS construction, such as KADS (Wielinga, Schreiber and Breuker 1992). They are based on the construction and refinement of knowledge models at many abstraction levels using graphical, semiformal and computational representations. However, as these approaches have general purposes, they cannot make use of particularities found at some problem solving methods.

Active Design Documents (ADD) are computational systems that follow, support and document the designer’s work (Garcia 1992). These systems use a computational model of the design process to execute their tasks. This model is expressed in a knowledge representation language appropriate for parametric design problems.

Knowledge Acquisition Techniques

Manual techniques of knowledge acquisition, such as interviews [McGraw and Harbison-Briggs 1989] and protocol analysis [Ericsson and Simon 1984], allow the designer to express the knowledge in natural language. The designer describes the design process without being forced to adjust the knowledge for any type of formalism. However, these techniques frequently produce an incomplete, inaccurate and ambiguous knowledge representation language is simple, model acquisition and formalization are still hard and complex activities. Usually, the knowledge acquisition process consists of a knowledge engineer interviewing the designers and formulating a graphical model of a parametric dependency network. After validating the model, the system is implemented using ADD representation language. During this phase, incompleteness and inconsistencies often arise on the graphical model. At the last phase, a prototype is presented to the designer who rejects, suggests changes or approves it.

The main problem on this process is the appearance of semantic noises caused by difficulties on communication between designers, knowledge engineers and programmers. As a result, many iterative development cycles are necessary until an acceptable system is achieved. Additionally, the knowledge acquisition process does not consider that changes in the computational model of design may be required during problem solving. Therefore, it is difficult for active design documents to deal with the exploratory nature of design problems.

This paper presents a new approach for acquiring and formalizing the knowledge used by active design documents. In our approach, the designers themselves become responsible for constructing and changing the knowledge base. The approach is based on QOC, a well-known argumentative approach for capturing design knowledge (MacLean et al. 1991).

Following, we briefly discuss some techniques of knowledge acquisition. Then, we present our approach and illustrate it by showing a prototype. In the end, we draw our conclusions.
description. Thus, the reduced effort made by the expert makes knowledge formalization hard, since the knowledge engineer must interpret, complete and refine the knowledge expressed by the expert.

Design knowledge acquisition techniques based on hypertext (Conklin and Begeman 1989) assist the designer to produce organized knowledge descriptions because nodes and links represent categories of design concepts. Therefore, it induces the designer to provide more accurate and complete design knowledge. For instance, QOC (MacLean et al. 1991) uses a semiformal representation for guiding the designer in a systematic exploration of design questions. However, QOC does not provide an operational knowledge model for the design process. Therefore, it still requires the knowledge engineer services. Moreover, QOC does not provide active computational services for supporting the knowledge engineer. For instance, conflicting design decisions can only be identified by browsing the hypertext documentation.

Knowledge based shells, such as Kappa¹, require that the designer express the design knowledge directly in a formal language. This may restrict the deliberation process and obstruct the knowledge acquisition. Some important aspects of the design may be discarded just because it is difficult for the designer to express them using the formal language (Shum 1997).

The HyperADD Approach for Knowledge Acquisition on Active Design Documents

In this section we present an incremental and hybrid approach for knowledge acquisition on active design documents. Our approach is incremental because the designer first describes the knowledge (by using a QOC based hypertext system), then formalizes it (by using a high level interface for the ADD representation language) and finally tries the new design process model. Our approach is hybrid because it integrates semiformal and formal representations. The integration of these representations allows the designer to express the knowledge using natural language and formalize it as a computational model without requiring the intervention of a knowledge engineer.

Computational models cannot anticipate all aspects involved in the design process due to the design exploratory nature. Opportunistic situations will demand improvements at the computational model during design problem solving. The designer, who detects these situations, should make the adjustments in an interactive way, as suggested by (Cartwright 1997). This interactivity is achieved in HyperADD approach because the designer can modify the computational model during the design process. The ADD interface is automatically adapted for the new model.

The Argumentative Process

In our approach, before formalizing the design knowledge, the designer semiformaly describes what he/she wants to formalize. The QOC based semiformal structure induces the designer to reflect about the design issues. We believe that this argumentative process will subsidize the formalization process. For semiformaly describing the design process, the designer uses some categories of design concepts and relationships. Figure 1 shows a UML graphical representation for the conceptual and navigational scheme of our approach. These concepts and relationships organize the designer’s argumentation. The designer may describe the concepts in natural language by specifying their "description" attribute.

The question class represents the design parameters. There are three specializations of this class: requirement specification question, calculus question and decision question. Each specialization has a different way of being resolved. A requirement specification question does not depend on any other design question. It is defined by the external environment, by the user or by the own designer. A calculus question is determined by the application of a mathematical function or a heuristic rule to the values of the questions upon which it is directly dependent. A decision question evaluates a set of options by analyzing restrictions and criteria. It indirectly depends on other design questions. Typically, the option that satisfies all the restrictions and optimizes the criteria will be chosen.

Figure 1 - Conceptual Scheme

The order in which the questions are solved is established by the "argument" and "condition" relationships. The "argument" relationship determines which are the arguments of a calculus question. The "condition" relationship determines which are the questions used by the set of decision criteria and restrictions. A dependency may also be established by the "consequent" relationship between an option and a question. This relationship means that choosing one option implies the need for solving a consequent question.

¹ http://www.intellicorp.com
Criteria and restrictions assess the set of options of the decision question. The "evaluates" relationship between a criterion and a set of options establishes that they will be evaluated according to this criterion. The "constrains" relationship between a restriction and a set of options qualifies each option as viable or not, according to the restriction.

Models QOC may be represented graphically by a graph, which makes easier the visualization but brings an additional effort to organize the layout. In our approach we chose to represent the argumentation as a hypertext. For supplying the lack of visualization, we defined navigational structures that aid the argumentation construction and reading. These structures are represented by the nodes and links on the conceptual scheme (figure 1). The nodes represent the designer's possible visions and the links represent the possible directions of navigation.

The conceptual scheme is navigated to assist the needs of the knowledge acquisition. One user of the navigational structures is the design process model constructor. This designer needs to navigate on the computational model as well as refine it incrementally. Another user is the designer that uses the ADD model for designing. This designer needs to input parameters values, ask for suggestions and explanations, and navigate the design process model.

The "question" node presents information about the question as well as the links tied up to the node. These links connect the question with local information such as, its options, criteria, restrictions and the questions that depend on it. The user may also consult how the set of options is evaluated.

The user is free to navigate through the questions network by the "dependent", "argument" and "condition" links. These links produce a local movement on the network. Global movements are made through the main menu or using the navigated questions history.

Figure 2 shows the hypertext page of a question that selects a case from a database. The right frame shows the contents of the visited question and the navigational links classes. The left frame shows the navigational links instances. One of the bottom frames shows the history of the visited questions.

The Formalization Process

By defining the set of design questions and the relations between them, the designer determines the order in which the design parameters are evaluated. However, the designer still needs to specify how each question should be computed. The HyperADD interface allows the designer to formalize the knowledge in a higher level representation than the ADD language formalism. In this section, we show how the design process model is formalized.

In a calculus question, the designer writes a formula that defines the question using its arguments. A formula is an expression written in the symbolic processing language Lua (Ierusalimschy and Celes 1995). The expression operands are the questions related to the calculus question by the "argument" relationship. The operators are arithmetic, relational, logical or catenated. The expression may also contain calls to functions encoded in Lua.
Values must be assigned to the options properties to evaluate the options. The user browses the set of options and chooses a property. For each property, a value matrix is edited. Alternatively, the acquisition of options may be automated through the use of an external source, such as a database, or by defining the interval of values that certain attributes may have. In both cases, the number of generated options may be large. Such situations demand heuristic search to find the best option.

Criteria and restrictions may be related to a pattern. In our example, both criteria require that the options are close to the Temperature Difference and Pressure Difference patterns (figure 3).

The designer should write a utility function to specify a decision question. A criterion weight defines its degree of importance. The utility function uses them to balance the multiple criteria. The utility function is also an expression written in Lua. Figure 3 also shows how to acquire a utility function for evaluating the "DesignCase".

**Experimentation Process**

Figure 4 shows the design interface automatically generated by our knowledge acquisition tool. We offer this capability to eliminate the need of intervention in the generation of the ADD system by a knowledge engineer or a programmer. Thus, the designer can experiment the computational model and refine it until it attends the designer's needs. In the interface, two button exists for each question: the button "Calculate" asks the ADD system to calculate and suggest an answer to the question and the button "Explain" provides an explanation about the way the question was resolved.

**Conclusions**

In this paper, we have proposed an argumentative approach for knowledge acquisition on ADD systems. Designers are required to

- deliberate about how the design process should be executed. During the deliberation process they must construct an argumentation. An hypertext based approach instigates the designer's reflection and makes easier the navigation and revision of the knowledge description. The product of this activity is a semiformal model of the design process.
- formalize the design process using a high level interface for the ADD representation language. This interface reduces the need of codifying the knowledge directly on this formal language. The product from this activity is a computational model for the active design document.

Our approach lets the designer free for choosing the best moment to formalize the knowledge. In other words, the designer first builds the relations among the design questions, criteria, restrictions and options and later, details them by specifying formulas, heuristic rules and values of properties.

Our approach lets the designer on the control over every modeling phase: conceptualization, formalization and experimentation. As the designer uses the computational model and detects a situation that requires changes on the model, he/she can easily modify and adapt it. As a consequence, the resultant ADD system will be able to cope with the exploratory nature of design problems.
We expect that the integration of argumentation and formalization will produce the following benefits:

- Induce the designer's reflection about different design alternatives and processes. This is achieved by the use of the QOC argumentative structure.
- Make easier the modeling process since the design model evolves from the semiformal to the formal level. Therefore, our approach may avoid restricting and hampering the reasoning whereas these situations often arise when expressing the knowledge directly in a formal representation.
- Allow the formalization in a more abstract level than the one of ADD programming language. This is achieved by providing computational services that aid the generation of the ADD computational model.

We have not constructed a complete ADD system using HyperADD yet. Indeed, we have only constructed part of an active design document system for the design of process plants of oil platforms. Our evaluation is that HyperADD makes easier the construction of ADD systems than the old approach. However, it was a knowledge engineer that used HyperADD for constructing the system. It will be necessary that a designer use it for a better evaluation.

There are some topics for future research. We have implemented a blind search strategy to evaluate decision questions. We need to extend the prototype by including new heuristic search strategies. We have also based our argumentation process in QOC, which is a simple notation. But there are more expressive notations, for instance, SYBIL (Lee 1990).

References


