PUC

Series: Monografias em Ciência da Computação,
No.30/89

ELICITATION OF APPLICATION LANGUAGES

Júlio C. S. P. Leite

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 – CEP-22453
RIO DE JANEIRO – BRASIL

PUC/RJ - DEPARTAMENTO DE INFORMÁTICA

Série: Monografias em Ciência da Computação, 30/89

Editor: Paulo Augusto Silva Veloso November, 1989

ELICITATION OF APPLICATION LANGUAGES

Júlio C. S. P. Leite

This work has been partially sponsored by FINEP

In charge of publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC RIO, Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453 - Rio de Janeiro, RJ
BRASIL

Tel.:(021)529-9386 TELEX:31078 FAX:(021)274-4546
BITNET:userrtlc@lncc.bitnet

Elicitation of Application Languages

Julio Cesar S. P. Leite
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de S. Vicente 225 Rio de Janeiro 22453
Brasil

July 1989

Abstract

Domain modeling demands that the knowledge of what should be modeled be available. Acquiring this knowledge or eliciting the domain is recognised to be a very hard problem. Using the idea that a domain will be represented by a special language, and using insights from the field of semiotics, we are investigating the elicitation of application languages. An application language is a special language aiming to capture the culture of a given social setting of an application. An application language could be viewed as a restricted domain language, since the main focus is to capture the observable knowledge in a given social setting, instead of the knowledge that is valid across social settings. Our approach to the elicitation of application languages is to empirically search for a method of acquiring these languages. The objective of this report is to give an idea of our ongoing research. First results and the problems encoutered are reported.

1 Introduction

Neighbors [Neighbors 84] believes that domains should be characterised by languages, with well-defined syntax and semantics. The compositional paradigm imagined by Neighbors is based on the fact that the semantics of one domain language would be described using previously encoded domain languages. This composition would be instantiated by what is called a network of domain languages, creating a powerful mechanism for the reusability of analysis and design. The compositional idea not only applies to the description of the semantics of a language, but also to the specifications of a particular problem as well. In this case one specification could be writtten in several available domain languages. We do not explore in this report the idea of multi-language specifications. Instead we explore only single language specifications.

For Neighbors, the production of a domain language has two processes: the process of domain analysis and the process of domain design. Domain analysis is concerned with what actions and objects occur in all systems in an application area, i.e., problem domain.

Domain design is how one specify the objects and actions, i.e., operations in the context of the Draco [Neighbors 84] system. This is done by the construction of a domain language parser, a domain language prettyprinter, source-to-source transformations for the domain, and components, i.e., semantics, for the domain.

Our approach attains what we consider fundamental in the Draco idea, that is, to use the language of the problem itself to express the problem. The approach, however, does not aim at collecting objects and operations for all the systems in an application area first and then designing the language. Instead the approach aims at eliciting the language used in the social setting of one application. As such, the process of domain analysis and of domain design are somehow interwined. We elicit the language using a semiotic framework, that we believe, naturally, leads itself to an implementation, which Neighbors calls domain design.

Using an anthropological or semiotical approach to the problem, we assume that an application area, in a social setting, has its own culture that is, "the laws of signification are the laws of culture" [Eco 79]. As such, there are two strategies that should be used during the process of acquiring the application language. First, the actors involved in the process of acquiring a language should, as much as possible, act as anthropologists. That is, the software engineers should, as much as possible, live or participate in the application arena to really capture the application culture. Second, since *language* is a reflection of culture, the knowledge about the application should be reflected in a language.

At the moment we have been dealing with just one example of the social setting and it remains to be seen how the method applied to it is extensible to other systems in the same application area but with a distinct social setting.

In the next sections we summarize some of the basic ideas of semiotics, use some of those ideas in proposing an overall schema for the production of application languages, and report on our ongoing research.

2 Application Languages

Eco [Eco 79] believes that it is necessary to distinguish between *signification* and *communication*, in which the process of communication needs a system of significations. Although recognizing that they are strictly intertwined, Eco proposes a theory of codes and a theory of sign production. If a system of codes should reflect a previously existing social convention, then it is obvious that one should capture this social convention before producing the system of codes, but this system of codes must follow the social convention if it wishes to be a medium of communication in the same social setting.

Using Eco's theory of codes, signs are seen as *sign-functions*. "A sign-function arises when an expression is correlated to a content, both the correlated elements being the functives of such a correlation". A system of codes is the underlining representation used by sign-functions. Our approach to language acquisition is based on the elicitation of sign-functions.

We use as our system of codes, a very simple one, proposed by Eco [Eco 79]. It has four different entities:

- signs,[1]
- notions,
- behavioral responses, and
- rules.

Signs are the syntatic representations, i.e., expressions. The notions are the set of contents that could be related to a sign depending on different viewpoints. Expressions or signs could be related to different contents or notions. Also contents or notions could be related to different signs. A behavioral responce is the effect of a sign-function on the universe of discourse[2]. A rule is the set of relationships linking signs to notions and to behavioral responses.

The notions and the responses assigned to a given sign, although using just natural language descriptions, should be enough as semantics, given an existing social convention. This is the point of using the semiotic approach.

3 Experiment

Given our objective of empirically studying ways of producing application languages we have been conducting an experiment with the following purpose and viewpoint.

- The purpose is to elicit the language used by a set of actors in a given social setting.
- The viewpoint is that of a software engineer, a student, using a preliminary version of a language elicitation method.

The experiment was conducted in a seminar on the Draco paradigm. There were three case studies; the PUC library, a small doctor's clinic, and an inventory for auto parts. The university library case was performed by two graduate students, the clinic by one graduate student and the inventory by one graduate student. The method used by these students is described next.

3.1 The Method

The method is actually a series of heuristics to capture the language of a given application. We divide the method into two set of heuristics. One is the general sequence and basis of the method, and the other is a set of more detailed heuristics for each of the steps sketched in the first set of heuristics. The first set of heuristics is as follows:

[1]Eco distinguishes between signal and sign. Something is a signal if there is no concern with its content. Since we use "signal" with a semiotic purpose, we use "sign" for meaning both sign and signal.

[2]Universe of discourse is the overall context in which the software will be developed. The universe of discourse includes all the sources of information and all the people related to the software. These people are referred to as the actors in this universe of discourse. It is the reality trimmed by the set of objectives established by the ones demanding a software.

1. The software engineer interviews and observers the users, keeping track of words or phrases that seem to have a special meaning in the application.

2. The software engineer makes a list of words and phrases, that *seem* to have a special meaning in the application, and checks the meaning with the users at a special interview centered on the words or phrases grasped in the first place. In this process, other words or phrases that had not been listed before, may appear. Repeat the process to find these.

3. Represent the words and phrases elicited by means of the simple semiotic system of codes. Use the principles of *circularity*[3] and *minimal vocabulary*[4] to describe the signs.

4. Check with the users the descriptions produced in the proposed representation.

5. Use verbalization to validate the system of codes elicited and to help the process of building a grammar out of the elicited sign-functions.

6. Build a BNF description for the syntax of the language.

7. Write sample programs in the language, i.e., specifications, in order to validate the BNF description.

8. Identify possible heuristics for optimizing the programs in the language.

9. Describe the semantics using an operational approach, that is using the Draco idea of components.

The detailed set of heuristics is shown in [Leite 89], in which the method and the case studies are described.

3.2 Results and Observations

Using the three domains we applied the method up to the step 7. That is, we managed to get third parties to produce application languages up to the stage of a first validation of a BNF description.

The students were presented with a very rough draft of a description of the method, and were guided by a series of interviews with the author. The detailed guidelines were provided as the work progressed.

The presentation of the system of codes was done as a list of signs, and at each sign the description of its notions and its behavioral responses. Each language sign used in

[3]The principle of circularity is as follows: in the description of notions and behavioral responses maximize the use of signs, i.e., words and phrases of the language being elicited. That is, use the language to describe itself.

[4]The principle of minimal vocabulary is as follows. In the description of notions and behavioral responses minimize the use of signs exterior to the language being elicited, and when using these words and phrases, make sure they belong to the basic vocabulary of the natural language in use. Moreover, as much as possible, have a clear mathematical representation, eg. set, belongs, union, intersection, function.

the description of notion or behavioral response was underlined, thus making the reference between symbols explicit in the form of a rule.

The university library case elicited a system of codes comprised of 56 signs, 118 notions, and 27 behavioral responses. The rules was composed of 297 cross references. The BNF grammar produced has 60 rules.

The clinic case elicited a system of codes with 41 signs, 48 notions, and 55 behavioral responses. The rules was composed of 215 cross references. The BNF grammar produced has 32 rules.

The inventory case elicited a system of codes with 34 signs, 34 notions, and 15 behavioral responses. The rules was composed of 36 cross references. The BNF grammar produced has 28 rules.

The first difficulty found by the students was to really understand what was a sign in the language. The vague notion given in the first set of heuristics made itself clear only by a trial and error approach. Once the students grasped the idea of words and phrases, that *seem* to have a special meaning the work really progressed. The general guideline given here is that their objective was to elicit a language, and not the problem to be solved by a piece of software. This difference is important since we want to capture symbols as well as their denotations and conotations, and not *concepts* to be mapped into functions and data as the usual software engineer does.

Another difficulty found by the students was to really differenciate between notion and behavioral response. The clear semiotics idea of denotation and connotation was not very easily grasped by the students. The general guideline provided by the author was to identify behavioral response as an impact, a change of state. That helped, but we recognize that further research is necessary to provide better heuristics. It was also difficult at first to classify constraints or restrictions. We opted for describing them as behavioral responses.

It is not clear that the students checked the system of codes with the actors of the social setting eventhough doing so was prescribed time and time again by the author. Not much can be said about that.

The process of verbalisation was, according to the students, the one that helped them best in the cleaning of the system of codes as well as in figuring out how to express situations with the language they elicited. The general guideline provided by the author was to form sentences describing processes in the application, using, if necessary, control structures borrowed from programming languages.

For writing the BNF grammar, we oriented the students to restrict themselves to the part of the system of codes that would be used in a controlling system for the application area. That is, not all the signs would be present in the syntax, but only those that could be used for producing a system to control the library or the clinic or the inventory. Because of that, not all the signs elicited were used in the grammar. The grammar also had control structures not present in the system of codes. An interesting outcome of this exercise was the choice of paradigm used to underline the control structure of the grammar. The library and the clinic used an imperative paradigm, but the inventory BNF was formulated using a declarative paradigm. The use of the declarative paradigm was a personal choice of the student, because of his familiarity with Prolog. Further research on the implication of the underlining paradigm needs to be performed.

Step 7, writing programs using the language of the grammars helped, as means to validate the grammar, as well as a mean to give the students a real feeling of the possibility of specifying systems in their own language. The parsing of the programs were done by hand, since there was no automatic support.

Our experiments have been restricted to one social setting, and we have not yet fully applied all the steps of our method. There are several aspects still unclear, but we believe that the continuation of this empirical search for a method will produce useful information about acquiring domain knowledge.

In this proposition we used semiotics as to provide a system of codes for representing the *signification* part. It is obvious that our scheme has to be further developed to support the *communication* part as well. The communication part would be the expression of the requirements or the specification of the control system of an application area itself. The application language is the underlying representation for that expresion.

From our experience so far, it is clear that the system of codes needs automated support. It is our plan to start developing an hypertext system capable of implementing the semantic net[5] expressed by our system of codes.

With respect to the validation of a system of codes, we should start using the viewpoint approach to elicitation [Leite 88] as another way, besides the verbalisation and check-with-the-user, of providing validation capabilities.

An explicit guideline of our method is to avoid abstraction when eliciting the language. This is, somehow, contradictory with the belief that the right abstraction should lead to better reusability. In some sense this may bring to bear the idea of language extensibility, but we do not have much to say about at that at this point.

References

[Eco 79] Eco U, *A Theory of Semiotics*, Indiana University Press, Bloomington, 1979.

[Leite 88] Leite, J.C.S.P., *Viewpoint Resolution in Requirements Elicitation*. PHD thesis, Dept. of Comp. Science, Univ. of Calif., Irvine, 1988.

[Leite 89] Leite, J.C.S.P., Produção de Linguages da Aplicação, Primeiros Resultados; *Departamento de Informática PUC/RJ*, Rio de Janeiro - Brasil, 1989.

[Neighbors 84] Neighbors, J., The Draco Approach to Constructing Software from Reusable Components. IEEE Trans. on Software Engineering, SE-10 (Sep. 1984), 564-573.

[5]In the process of "formalizing" the proposed system of codes, it is clear to us that extensions to our model would have to be made.