THE USE OF A SEMI-FORMAL SPECIFICATION LANGUAGE TO BRIDGE THE GAP BETWEEN INFORMAL AND FORMAL SPECIFICATIONS OF SOFTWARE SYSTEMS: A PROPOSAL *

Stanley **LOH** (1)
José Mauro **CASTILHO** (2)

UFRGS, Instituto de Informática
Departamento de Informática Aplicada
Caixa Postal 15064
CEP 91.501 - Porto Alegre - RS, BRAZIL
E-mail: castilho@inf.ufrgs.br

(1) Prof. of Universidade Católica de Pelotas and Universidade Luterana do Brasil
(2) Prof. of Universidade Federal do Rio Grande do Sul

* This work is partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil.

ABSTRACT

The use of a semi-formal specification language in the process of constructing a formal specification for a software system is discussed. The process is executed in two separate steps, and the whole formalization activity could be performed with less effort, thus achieving better final results.

A set of heuristic rules is proposed, to guide the two translation steps (from informal to semi-formal specification, and from semi-formal to formal specification).

A dictionary of terms that appear in the specification plays an important role in the process, helping in the identification of "synonymous" parts of the specification.

Keywords: database design, conceptual modelling, software specification in logic, formal specifications.

---

1. Introduction

In the development of software systems, the phase of Requirements Analysis and System Specification is perhaps the most important one, due to its influence on the quality of the final product of the development process.

Once the specification is constructed, the results of the following design phases are in some sense consequences of this specification. If the specification is formal, we may have some advantages: non-ambiguous understanding of requirements, formal verification of properties of the specification, final user validation through the execution of the specification (with executable specifications). However, formal specifications are not so natural and legible as **informal specifications**, written

with the use of natural textual languages, or graphical and diagramatical languages of some specific application domain.

Even considering the advantages of formal languages over informal ones, some authors ([6], [1], [2], [3], [5]) advise the use of the two types of languages. In the approach taken in this work, each type of language plays a different role: informal languages are used in the requirements elicitation phase of the systems development cycle, and in the users' "visual checking" of the informations given; formal languages are useful for precisely describing requirements collected and for the unambiguous communication between the systems development agents.

Nevertheless, no matter how critical is the construction of (formal) specifications for software systems, this activity is more art then science, and depends much on the specifier's experience and personal insight on the system.

This paper introduces a two-step procedure for the construction of formal specifications for application software systems, using an intermediate, or "semi-formal", language to express the output of the first step (and input for the second). It is intended that this intermediate language has some characteristics of formal languages (it is more precise and hopefully less ambiguous than natural languages), and some characteristics of informal languages (it is more userfriendly and less complicated to use than ordinary formal languages).

Together with the semi-formal language, another essential part of the formalization procedure is a set of heuristic rules, used in the transformation steps of informal to semi-formal, and semi-formal to formal, specifications. Some of those rules (described here in natural language, to simplify the explanations), and their use, will be presented through examples, in section 3.

The approach proposed here for the construction of formal specifications of software systems is adopted in the SILOG project, an academic research project of the Information Systems Group, in Instituto de Informatica, UFRGS. The project's goal is the construction of a software environment prototype to aid database designers in their task. The Requirement Analysis and Data Modelling module of this environment accepts a set of (portuguese) natural language sentences, collected previously in interviews with users of the future database system, that form the initial informal specification.

The formal specification language used in SILOG is based on first order mathematical logic, and contains three sublanguages, one for each of the three parts that compose SILOG's formal specifications for database systems:

a. data structures definition sublanguage;
b. update and query operations definition sublanguage;
c. static and temporal integrity constraints definition

sublanguage.

The first sublanguage supports relational-like data structure definitions. The second one defines the behaviour of encapsulated operations over the database, in terms of their pre- and post-conditions. The third sublanguage is a logic language that allows the description of constraints on the data stored in the database, and of constraints on the database state transitions.

Some examples (of a library database system):

a. user(x), book(y), borrow(x,y)
b. borrowing(x,y):
   Pre-conditions: user(x) and book(y)
   Post-conditions: borrow(x,y)
c. borrow(x,y) implies (user(x) and book(y)).

In the next section we introduce the semi-formal language used in the formalization process.

2. The semi-formal language.

In the definition of SILOG's semi-formal language, some criteria served as guiding rules: sentences in the language should have simple structures. Necessarily, using such a language, one should write or speak several sentences to communicate the same idea contained in an ellaborate natural language sentence, and is induced to be more explicit and to eliminate irrelevant informations.

Another criterium used was some similarity with the formal language used in the SILOG project, with the intent to facilitate transformations from the semi-formal to formal specifications.

Corresponding heuristic rules, to guide the translation of sentences between languages, were identified. It was found necessary to have a Term Dictionary, whose main role was to ease the identification, in one specification, of those parts that had the same (or similar) meaning.

Examples:
to consult = consultation;
students = pupils;
to consult = to be consulted;

The software specification in Semi-Formal Language follows the three-part division like the formal specification, and is detailed below.

2.1 Part of Data

Informations are described with simple sentences (one verb only) in the following format:

SUBJECT VERB VERBAL_COMPLEMENT,

where SUBJECT and VERBAL_COMPLEMENT may be substantives linked

by prepositions and each substantive may be qualified by adjectives.

Examples:

users borrow books;
books have long titles.

In some cases, adverbs may be used, like in format:
SUBJECT INTRANSITIVE_VERB ADVERB.

Example: borrowings are made everyday.

2.2 Part of Operations

Each update or query operation has a name, followed by a sequence of actions. Formats are like the following:

Operation_name = VERB VERBAL_COMPLEMENT (or SUBSTANTIVE + PREPOSITION replaciong the verb);

Action = VERB VERBAL_COMPLEMENT.

Actions must be of one of the 5 types below:

- solicitation or receipt of information;
- verification of condition (in this case, an "if" could be used after the verb);
- calculation of information;
- modification of information set;
- supply of information.

Example:
Hiring employee:
Receive name and address of employee
Check if employee is not registered
Register employee.

2.3 Part of Integrity Constraints

Integrity constraints must be written according to one of the standard formats identified in [4] (total of 15 approximately).

Some examples of standard formats:
ALL/ANY subject MUST verb verbal_complement;
subject WHO/THAT/WHICH verb_1 verbal_complement_1 MAY NOT verb_2 verbal_complement_2;

Examples:
Any user must have 1 name, at most.
Users who have debts may not borrow books.

3. The Two-step Formalization Procedure, and a Sample of the Set of Heuristic Rules for Language Translation.

There are two sets of heuristic rules, each one associated to a particular translation step: (i) informal to semi-formal

translation, and (ii) semi-formal to formal translation. Each set has rules for data, operations and integrity constraints parts.

Some examples of transformations, and the corresponding rules used, are presented below in an abreviated mode (these examples were taken from a specification study of a library database system; the complete set of heuristics can be found in [4]).

3.1 Transformation from Informal Language to Semi-Formal Language

Part of Data

Informal Language:

Users borrow bibliographic materials from the library. Users must give their names when they use library services.

Semi-formal Language:

A) users borrow materials
B) users have names

Rules used:

A) - each sentence in the informal language must be analysed separately; this is a good rule!);
- verbs must be checked (some of them may have special roles, like "have", for instance);
- only transitive verbs are allowed.

B) - pronouns must be replaced by corresponding names;
- prepositions must be analysed, as well as the relationship defined by them.

Part of Operations

Informal Language:

When executing the borrowing activity, the library employee requests the user's name and the title of the desired material. After that, the employee checks if the user is registered in the library and if the material is available. If the answers are positive, the employee annotates in a special form that the user has borrowed that material.

Semi-formal Language:

Borrowing material:
Request name of the user
Request title of the material
Check if user is registered
Check if material is available
Annotate that user has borrowed material.

Rules used:

- each verb may represent an action;
- each action must be one of the 5 types established previously ;
- the order of the actions must be respected;

- physical storage information, and informations about who does the action must be disregarded;
- redundant information must be eliminated.

Part of Constraints

Informal Language:
Only faculty personnel (teachers) may borrow journals. And users may only borrow available materials.

Semi-formal Language:
A) User may only borrow journals if he/she is a teacher.
B) Only available materials may be borrowed by users.

Rules used:
A) - expressions like "must", "may not", "only", "if", "never", "always", etc, must be checked; they may identify a constraint;
- try matching the constraint with one of the defined formats.

B) - the same first rule above;
- "to be borrowed by" is the passive mode of "borrow".

3.2 Transformation from Semi-Formal Language to Formal Language

Part of Data

Semi-formal Language:
A) users borrow materials
B) users have names

Formal Language:
A) user(x), borrows(x,y), material(y)
B) user(x), have(x,y), names(y)

Heuristic rules used:
A) and B)
- transform the format "subject verb complement" to "subject(x), verb(x,y), complement(y)";
- use singular names.

Part of Operations

Semi-formal Language:
Borrowing material:
Request name of the user
Request title of the material
Check if user is registered
Check if material is available
Annotate user has borrowed material.

Formal Language:
borrowing_material(n,t):
∃ u,m (
Pre-conditions:
name(n) and title(t) and user(u) and has(u,n),
material(m) and has(m,t) and available(m).
Post-conditions:

borrows(u,m)).

Rules used:
- analyse separately each action of the operation;
- every variable should appear as argument in a "type defining" predicate;
- "input" words (words acting as objects to verbs like "to request", "to receive"), should appear as variables that are input arguments of the operation;
- preposition "of" may indicate the relation "has(x,y)";
- facts after "check if" are a pre-condition;
- expressions like "subject IS REGISTERED" must be translated to "subject(x)";
- the verb "to be" may indicate a specialization (subtype) of a concept;
- the expression "ANNOTATE fact" indicates that "fact" must be a post-condition;
- redundant occurrences must be eliminated;
- variables which appear in pre-conditions without a preceding negation symbol must be associated with an existencial quantifier ("∃").

Part of Constraints

Semi-formal Language:
A) User may only borrow journals if she/he is a teacher.
B) Only available materials may be borrowed by users.

Formal Language:
A) ∀ x,y
user(x) and journal(y) and borrows(x,y) --> teacher(x).
B) ∀ x,y
user(y) and material(x) and borrows(y,x) --> available(x).

Rules used:
A) - the format "subject MAY ONLY verb complement_1 IF HE/SHE/IT IS complement_2" is directly translated to "∀ x,y subject(x) and complement_1(y) and verb(x,y) --> complement_2(x)".
B) - the format "ONLY subject MAY verb complement" is translated to "∀ x,y complement(y) and verb(x,y) --> subject(x)";
- expressions with adjectives, like "adjective name", must be translated to "name(x) and adjective(x)";
- standard terms, defined in the Terms Dictionary, must replace their synonyms.

4. Conclusion

A semi-formal language that could be used to establish an intermediate "stepping stone" between informal and formal specifications of software systems, and a corresponding formalization procedure (based on the use of a set of heuristic language-transforming rules), were introduced in this paper.

Some small experiments were made with the use of this semi-formal language and with the set of heuristic rules, and it was noticed that the formalization process, at least for the class of applications chosen (mainly normal or "conventional" data management applications, like library management) is facilitated by the easy and simple identification of specification elements for the formal language. A bigger experiment is going on, to test more deeply our approach, in the specification of a database system for the management of the laboratories of our Instituto de Informatica.

The set of heuristic rules identified has proven to be useful for the discovery of relevant information patterns in sentences of the languages, and for the identification of transformation actions. Also, this set may be used to build a "knowledge base" for some expert system that could perform some tasks in the formalization process. However, the set is not complete yet. More experiments need to be made, in order to improve it.

Some other formalization problems remain. The approach introduced here still needs intensive participation of software specialists. Also, there is still no way of guaranteeing the "completeness" of an informal specification, or to tell that it contains all relevant information for the implementation of the system. Only end-user validation can detect missing requirements.

As a final comment, it is necessary to remember that the semi-formal language presented here was only tested with the formal language used in the SILOG Project. It is the intention of the authors to evaluate its effectiveness in the formalization process with other formal languages (like VDM, OBJ, Z, etc).

REFERENCES

[1] R. BALZER et al. Informality in program specifications. IEEE Transactions on Software Engineering, New York, v.SE-4, n.2, Mar. 1978.

[2] R. BALZER. A 15 year perspective on automatic programming. IEEE Transactions on Software Engineering, New York, v.SE-11, n.11, Nov. 1985.

[3] M. D. FRASER et al. Informal and formal requirements specification languages: bridging the gap. IEEE Trans. on Software Engineering, v.17, n.5, May 1991.

[4] S. LOH. Uma Linguagem comum entre usuários e analistas para definição de requisitos de sistemas de informação. Porto Alegre, CPGCC/UFRGS, 1991. MSc Dissertation (in portuguese).

[5] K. MIRIYALA & M. T. HARANDI. Automatic derivation of formal software specifications from informal descriptions. IEEE Trans. on Software Engineering, v.17, n.10, October 1991.

[6] T. J. TEOREY & J. P. FRY. Design of database structures. Englewood Cliffs, Prentice-Hall, 1982.