*Közlemények 31/1984. 51-61*

# ON THE SEMANTICS OF DESCRIPTION LANGUAGES

*P. RADÓ*

Computer and Automation Institute,
Hungarian Academy of Sciences

The dynamic growth (and rate of growt) of the amount and global cost of software products has been a tendency for the past 20 years all over the world. This process has brought about not only quantitative changes. The problems to be solved have been becoming more and more complex, the solutions - software systems - larger and larger, their construction more and more complicated. The efforts to solve the arising new problems have resulted in the birth of a new branch of the computer sciences: software engineering.

As experience has shown, the problems capable of making a software project unsuccessful, arise mainly at the early phases of the project [1]. After many years of research and numerous failed software projects the importance of requirement specification and design is generally recognized, although the available tools and methods provide no unique and universal solution for the practicioner.

One of the most promising approaches to the problem of requirements specification and design description is that of computer aided systems. It is quite natural, that the computer can help, storing the text, producing listings of different formats, answering interactive queries e.t.c. Of course this much can be achieved using any text editor. The text editors and the computer aided specification systems differ in the latter's ability to understand the meaning (strictly defined syntax and semantics) of the stored data. This capability adds the valuable facility of automatic consistency checking to the features of the system.

In this paper we propose a general computer aided specification system model. A semantics definition valid for a wide class of specification languages is given based on this model.

## I. THE ARCHITECTURE OF THE COMPUTER AIDED SPECIFICATION SYSTEM

Considering the text editor as a simple computer aided specification system its functioning can be described relatively simply (see *Figure 1*): the a specification is accepted in interactive mode and stored in a data file. There is possibility to change the stored data, to obtain different listings. The text editors support some consistency checking of the specification as well, providing different search facilities based upon formal criteria (all occurences of a given character string, e.t.c.).

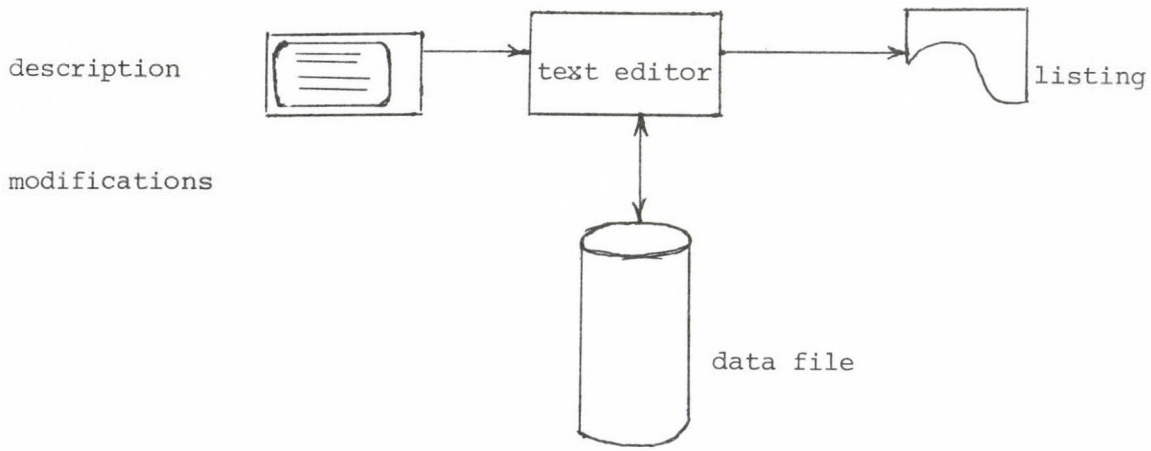


*Figure 1.*

*The text editor*

There is a principal difference between text editors and the specific computer aided specification systems: the former accepts any informal text, while the latter checks its input,

and accepts only correct sentences of a precisely defined machine analyzable language. This description language should be extremely user friendly, easy to read, self explaining (its users in general are not computer specialists) and at the same time should possess sufficient descriptive power.

These points are demonstrated by a tiny PSL [2] description fragment in *Figure 2*. The language is structured into sections (indicated by indentation in Figure 2). Each section consists of a section header (the first line of the section), which specifies the object we want to tell things about, and a section body containing statements referring to the section header. The meaning of the test can be easily understood, and it is quite clear, that is structure is simple enough to be computer analyzable.

```
process payroll system;
   uses payroll input to derive payroll output;
interface administration;
   generates payroll input;
   receives payroll summaries;
process syntax check;
   uses payroll input;
   derives payroll file;
```

*Figure 2.*

*A PSL description fragment*

The strict description language is advantageous from the point of view of the communication. It is not only the man-machine interface, which needs a formal language. In the communication among people some defense is required against misunderstanding each other's thought expressed in human language. We refer to the well known teamwork problems, or to the apparent inability of the user to explain the problem to be solved and

to understand and check the proposed solution without misinterpretation.

The formal languages - besides the offered advantages - have their own weak points. In applying a particular language to the solution of some real world problem, it is often hard to match the predefined concepts of the language with those naturally arising from the analysis of the problem. The description language must be general to be widely applicable, and at the same time should provide concepts, which are suitable to describe the essential special characteristics of any problem within the - preferably wide - range of applicability. This contradiction leads to the development of the two-level or meta specification systems [4].

This approach - recognising the above mentioned advantages and disadvantages - provides software, which instead of giving supply to only one predefined language, encourages the user to define his own language. After the definition a two-level specification system will provide all the usual advantages of the computer aided specification systems with the user defined - and therefore problem oriented - concepts.

This is similar to the abstract data type definition mechanism of some programming languages, where a type is defined by its name and characterized by interrelationships with other types (abstract operations) [3]. An object of a given type is capable of possessing those and only those relationships which are permitted for the type according to its specification. In this regard the fixed language specification systems can be compared with the programming languages without type definition facility, while the two-level systems with those programming languages, which provide it.

The architecture of the two-level specification system is shown in *Figure 3*. The meta level system provides facilites to define the specification language to be accepted by the description level system later on. This connection is realized by tables created by the definition interpreter and used by all the

modules of the description system. We note, that the description level in itself can be considered as one-level self-contained specification system, once the description language has been fixed.

As a concrete example we mention the SDLA system [5]. At the meta level the SDLA user can define:

- the conceptual framework,
- the syntax of statements used throughout the specifications,
- semantical constraints associated with the concepts.

*Figure 4* shows a definition fragment. The defined language coincides with a subset of the PSL fragment of Figure 2.

```
concept process;
concept input;
concept output;
concept uses to derive (input, process, output);
        form process; uses input to derive output;
concept usage (input, process);
        form process: uses input;
```

*Figure 4.*
*SDLA definition fragment*

## 2. SPECIFICATION SYSTEM MODEL

First of all we note, that each specification system is a model in itself. A description language contains predefined concepts (for example in Figure 2 appear "process", "uses", e.t.c.), which are abstractions of real worls entities. Their existence is based upon the fact, that the different real world systems have enough in common to make possible a classification of their constituent parts into general types (concepts). The statements (concepts) of the PSL for example can

be used to describe any concrete information system, so they can be considered as general information system model.

The two-level specification system is an even higher level abstraction. It integrates the different specification systems into unique framework. The meta level can be considered as a specification system, modeling specification systems. Indeed, the "concept" concept of the SDLA is applicable to a wide variety of specification systems [6], as it graphs their essence - the use of abstract concepts (types) to describe the properties and the relationships of real world entities by a proper classification scheme.

The specification system can be considered as a special kind of database management system as well. It allows the user to classify, store, list and modify data via proper interface, while the software provides the technical details of storage and retrieval. A general database management system model was proposed by the ANSI/X3/SPARC committee in 1975 [7]. According to this model a database management system consists of the user interface (external scheme), details of storage (internal scheme), and the conceptual scheme, invisible from outside, but playing a most significant - if not allways explicitly stated - role in the functioning of the database management system. The conceptual scheme is an abstraction of the outside world to be modeled. The mapping between the elements of the external and the internal scheme is not direct, it is materialized by their referring to the objects of the conceptual scheme. Although the conceptual scheme may be invisible for the user, it is the very model, which determines the external and internal schemes. The conceptual scheme is the central element of the whole conception.

All three schemes have their corresponding counterpart in the computer aided specification systems. The external scheme corresponds to the specification language, the internal to the data management. The counterpart of the conceptual scheme is the method of modeling provided, the concepts available to describe the real world. In the case of the PSL these concepts are concrete object and relationship types and some semantical

constraints (see *Figure 5*).

```
type input;
type output;
type process;
type interface;
    :
relation generates (interfaces, input);
relation uses to derive (input, process, output);
relation uses (input, process);
    :
constraint uses to derive (input, process, output) implies
     uses (process, input) and derives (process, output);
```

*Figure 5.*

*PSL conceptual scheme fragment*

The definition of types is obvious (Figure 2 shows examples of their usage), but the formalization of the semantics is a harder task. For example the

```
process  P;
   uses I to derive O;
PSL description fragment, that is the
   uses to derive (I,P,O);
relationship automatically implies the
   process P;
      uses I;
      derives O;
fragment, that is the
      uses(P,I); derives (P,O);
```

relationships for any P,I and O objects. The "constraint" statement of Figure 5 describes this property of the PSL.

Another example of conceptual scheme is that of the SDLA.

On the meta level there is only one conceptual object type. Everything is expressed using

*type* concept (attribute 1,...,attribute n);

where all attributes are concepts as well. Without going into details we remark the proximity of this model to the relational data model [8].

It is also possible to define semantical constraints on meta level. The statement

*concept* $A(X_1,X_2,...,X_n)$ *implies* $B(X_i,...,X_i)$; $(k < n)$

for example is a generalization of the above mentioned PSL semantic constraint. Whenever on object of type A is created on the description level, the system automatically creates another object of type B. Semantic constraints of this type are called "implication" in the SDLA. While in the PSL it works only for a fixed relationship ("uses to derive"), on the SDLA meta level implication constraint can be defined between any two concepts.

## 3. SPECIFICATION LANGUAGE SEMANTICS

We can use the ANSI/SPARC model to describe the functioning of the computer aided specification system. According to the model, the processing of the arriving description may be considered as realization of two mappings - from the external scheme language (specification language) into the elements of the conceptual language (occurrences of conceptual objects), and then from the conceptual scheme language into data manipulation commands of the internal scheme. The query is the inverse of these two mappings, and the modification also can be described by them.

We define the semantics of the specification languages as a set of relationships between conceptual scheme objects. According to this definition only those relationships are

semantic, which involve actions executed between the realization of the two above mentioned mappings. For example, when a new description fragment arrives, the semantical constraints result in actions realized after mapping specification language statements into concept occurances, and before mapping these into data manipulation commands.

This definition implies the way of semantics specification. As any semantic constraint is a relationship between conceptual scheme objects, it should be defined in terms of the conceptual scheme.

There is no problem in the case of one-level specification systems. For example, the conceptual scheme of the PSL is a set of fixed types, therefore a routine executing the necessary actions may serve as definition (or realization of the definition) of a semantic constraint. The processing algorithm of the semantic constraint from Figure 5 is quite clear by itself, and the routines realizing it can be added to that part of the input analyzer which deals with the "uses to derive" relationship.

If we want to define semantics in a two-level specification system, we face additional difficulties. A semantics constraint statement can not be attached to a concrete object type, as it should operate with conceptual scheme objects (in case of the SDLA these are concepts). A formal tool, capable of defining semantics in general, not only for a given specification language, but for a wide family of specification languages is needed. As we have seen the SDLA's "implies" constraint is a generalization of a semantic property of the PSL's "uses to derive" statement. Its definition is correct - relationship between any two concepts - and the facility can be applied on meta level to specify semantic constraints in any defined description language.

There is a number of mathematically exact, but rather complicated, inpractical methods to describe general semantic constraints for the procedural (programming) language [9].

These methods don't seem to be applicable in case of description languages. It is a more realistic approach to generalize some of the semantic constraints of the existing one level specification systems based on the needs of the applications. We refer to the numerous SDLA publications ([4], [5], [6], [10], e.t.c.), which contain several examples of semantic constraint specification in concrete systems.

REFERENCES

[1] Boehm, B.W., Software Engineering. *IEEE Transactions on Computers*, C-25 (1976) 12, 1226-1241.

[2] Teichroew, D., E.A. Hershey; PSL/PSA: a Computer-aided Technique for Structure Documentation and Analysis of Information Processing Systems. *IEEE Transactions on Software Engineering*, SE-3 (1977) 1, 41-48.

[3] Liskov, B.H., A. Snyder, R. Atkinson and C. Shaffert; Abstraction mechanisms in CLU. *Communications of ACM*, August 1977.

[4] Demetrovics, J., E. Knuth, P. Rado; Specification Meta Systems. *Computer*, 15 (1982) 5, 29-35.

[5] Knuth, P. Rado, A. Toth; Preliminary Description of SDLA. *MTA SZTAKI Tanulmányok*, Budapest, 105/1980, 1-62.

[6] Knuth, E., P. Rado; Principles of Computer Aided System Description. *MTA SZTAKI Tanulmányok*, Budapest, 117/1981, 1-46.

[7] ANSI/X3/SPARC Study Group on Database Management Systems. *Interim Report*, 1975.

[8] Codd, E.F., Extending the Database Relational Model to Capture More Meaning. *ACM Transactions on Database Systems*, 4(1979) 4, 397-434.

[9] Donahue, J.E.; Complementary Definitions of Programming Language Semantics. *Springer Lecture Notes in Computer Science*,

[10] Knuth, E., F. Halasz, P. Rado; SDLA System Descriptor and Logical Analyzer. in Information System Design Methodologies: a Comparative Review (ed. T.W. Olle, et al.), *North-Holland*, Amstardam, 1982.

## Ö S S Z E F O G L A L Á S

### A LEIRÓ NYELVEK SZEMANTIKÁJÁRÓL

*Radó P.*

A cikk a számitógépes specifikációs rendszerek felépitését elemezve jut el a rendszerek egy széles osztályára alkalmazható modellhez. A modell alapján általános definiciót javasol a leiró nyelvek szemantikájára.

## О СЕМАНТИКЕ ЯЗЫКОВ ОПИСАНИЯ

П. Радо

Анализ архитектуры систем спецификации приводит к общей модели применяемой на широкий класс таких систем. На базе этой модели предлагается общее описание семантики для языков описания.