Formal specification of a scanning tunnelling microscope [1] [2]

Joaquín Nicolás (*), Juan Alcalde (*), Ambrosio Toval (*), Aurelio Arenas (**)

University of Murcia. C/ Santo Cristo, 1 (30001) Murcia (Spain)

Fax. +34-68-364151; Tlf. +34-68-307100; e-mail {jnicolas, atoval}@dif.um.es

(*) *Department of Informatics and Systems. Software Engineering Research Group*[3].

(**) *Physics Department*

Abstract

This paper presents a real case study of a formal specification for both the control software and the activities involved in topographies realisation of a scanning tunnelling microscope. The microscope was constructed in the Physics Department of the University of Murcia. In order to obtain an executable formal specification, the authors followed a methodology that implies the following tasks: obtaining a functional model of the selected experiences, the choice of a specification formalism and language, the establishment of certain rules to convert the functional model to a formal one, and the construction of the formal specification. The final executable formal specification was used both to validate the requirements and to check some properties and behaviour of the system described. Trough this project much has been learned about the use of formal techniques as well as its potential for being accepted as a viable way to improve requirements analysis in electronic instrument systems and related software development.

Keywords: Requirements Analysis, Algebraic Specifications, Formal Methods, OBJ, Evolutionary Prototyping, Feedback Control Systems, STM.

1. Introduction

Traditionally, the electronic instrument systems, such as oscilloscopes or spectrum analysers, have been built with devices controlled by hardware. However, during the eighties decade, with the improvement of computers performance and the reduction of their cost, the use of software in electronic instruments implementation has been fastly growing. Thus, so far, the introduction of Software Engineering techniques and methods in this area has been relatively new. Seldom, software developers and designers involved in these projects had enough theoretical knowledge about which techniques to use to analyse and design this type of applica-

---

[1]Partially granted by the PASO (*SOftware Action Plan*) programme OOAP, PC310 of the DGIII (EEC, ESPRIT Special Action P7506- PASO) and the Spanish Ministry of Industry.

[2]An early version of this paper was presented at the *II Jornadas de Informática*, organized by the AIEA and the Department of Informatics and Systems of the University of Granada, in Almuñécar (Spain), from 15th to 19th July 1996.

[3]Member of RENOIR (European Requirements Engineering Network of Excellence, approved by the EEC).

tions, in contrast with other computing fields, as for instance, information systems, compilers construction or relational databases design [5].

In this context, a variety of experiences are being developed at the present in order to apply formal methods to the analysis and design of embedded systems and electronic instrumentation systems. Among the benefits, we can find: obtaining reusable specifications, formalising the requirements and allowing designers to gain insight into its behaviour, so making possible to reason in a rigorous way about some critical features of such systems ([12]).

An algebraic formalization has been done in this case study, by using the functional language OBJ3 [8] [10], of a scanning tunnelling microscope, which was developed in the Physics Department of the University of Murcia. The algebraic specification of the microscope in OBJ3 allows not only to capture and record the requirements of the system to be designed, but also to execute the specification immediately in a formal environment, in which these requirements can be validated and the internal consistency of the specification verified.

The method followed to obtain the formal model of the microscope consists in the production of a preliminary functional model [5], after the system has been understood sufficiently and once defined the limits of the domain [14]. Later, once the formalism to be used has been chosen (Algebraic Specifications in this case study), the formal model from the functional one is built, by applying some conversion rules (developed exactly for that purpose) between both models.

The evolutionary prototyping strategy has been used, as opposite to throwaway prototyping, in order to produce and manage a well-defined set of requirements. The last OBJ3 prototype was used both to validate the requirements on behalf of final users and to formally verify some interesting properties related to the system operations.

2. The scanning tunnelling microscope

The system object of the specification is an atomic resolution microscope called scanning tunnelling microscope (described for the first time in [2], later by Lynding [13] and widely in [3]). The main goal of this system is, given a sample, to obtain the three-dimensional, real-space images of the sample surface at high spatial resolution. When the sample is clean and flat, even atoms can be imaged. Nevertheless, there are also other applications of this kind of microscopes as Tunnelling Spectroscopy, to characterise electrically the sample, Tunnelling Nanolitography and Electrochemistry as well as Biology applications.

The theoretical fundamentals of the microscope are based in a result of the Quantic Mechanics (called the *tunnel-effect* [17]), that says that when two surfaces are close enough that their wave functions overlap, a finite probability exists that electrons will cross the barrier between the surfaces when a voltage is applied between the surfaces. The resulting current, the so-called *tunnel current,* will be higher or lower relying on the conductors to be "nearer" or "further", also depending on the value of the potential difference and the work function that is characteristic of the surface electron densities.

The most important parts of this system are the microscope tip and the sample (see Figure 1). These are the two conductors to whom the potential difference is applied, called bias-voltage (Vp).

The physical system we are treating with is a feedback control system: the tunnel current value is read and the actuation value is calculated. This actuation will be a potential difference that will be applied on the piezoelectric element[4] "*z*" in order to bring nearer or further the tip from the sample. This approaching or receding will have an effect on next tunnel current data read.

The motion in all three directions (*x*, *y*, and *z*) is controlled by piezoelectic elements. Simple linear voltage ramps applied to the *x* and y piezo cause the tip to scan the surface and the voltage signal from a comparison circuit is directed to the *z* piezo.

During the performing of a topographic image, by altering the vertical position of the tip, the system tries to keep a constant distance between tip and sample while the tip is scanned over the sample surface. In other words it maintains a constant tunnel current (called reference current). When the tip moves along the surface, the system will react against the changes in the tunnel current produced by surface roughness (hills and valleys), by bringing near or receding the tip from the sample.

The topographic image is recorded as a two-dimensional array of integers representing heights (actuation values) at a specific *x*, *y* positions.

---

[4]The piezoelectric elements are lead zirconate titanate ceramic transducers (PZT). When an electric potential is applied across the piezo its length varies.

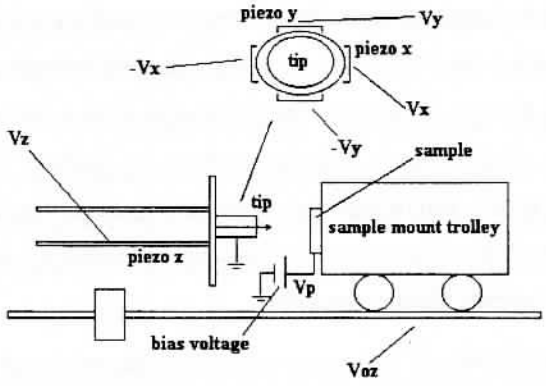

Figure 1. Schematic diagram of the scanning tunnelling microscope

3. The functional model

The study of the system through functional diagrams [5], that is to say, the system modelling through composition of functions, let us have a preliminary whole and clear view of it.

In Figure 2 it is shown how the microscope takes values of real current (current between tip and sample), denoted by the entry *RCurrent*, and obtains a graphic representation of the collected actuations (topography of the sample surface), denoted by the output *GraphRepres*.

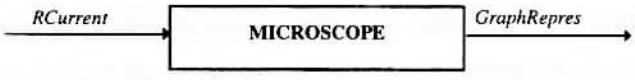

Figure 2. Functional diagram of the microscope

The system can be divided into two main functional parts: one is called the control part and the other one is the topography. This can be seen in Figure 3, which is the refinement of Figure 2. As it is shown, the control part takes the real current and calculates the successive actuations that will be applied upon the piezo *z* that supports the tip of the microscope. These actuations are denoted by the information flow *Actuation*. The part of topography takes these actuations and prepares them to be graphically represented, obtaining a graphical representation of the sample surface.

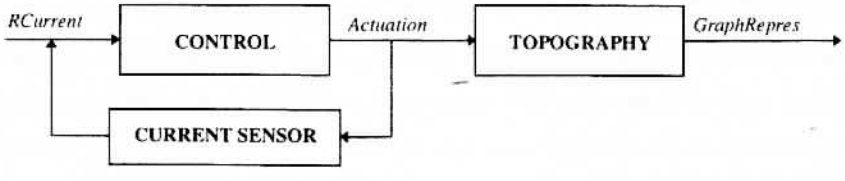

Figure 3. Functional diagram of the control system and topography

3.1 Control function

The activity of control (see Figure 4) itself consists of the following activities:

1. With the real current data (represented by the flow *RCurrent*) it is calculated the average (flow *Current*) of a certain number of values according to a parameter introduced by users (*triggering*). This is the function *AVERAGE*.

2. The differences (errors) between these averages and the reference current (denoted by *r*) are calculated. This function, called *ERROR*, has as output the flow *Error*.

3. Taking the errors, then the consecutive actuations are also calculated, according to the PI (Proportional+Integral) control algorithm (u(k)=u(k-1)+(q1 * (r - i(k-1)))+(q0 * (r - i(k))), where *u(k)* is the actuation at the instant *k*). The function *ACT* with the control parameters (*q0* and *q1*) and the initial actuation (*u0*) will produce the flow that represents the actuations, denoted by *Actuation*.

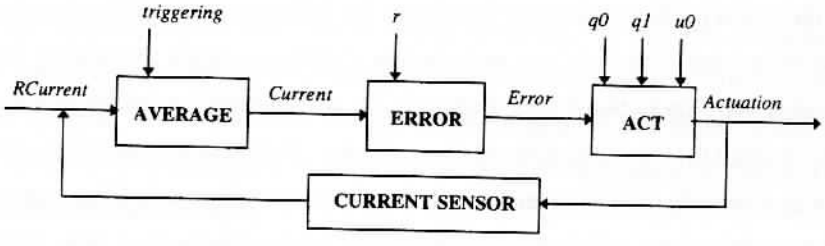

Figure 4. Functional diagram of control

3.2 Topography function

The function of topography, as Figure 5 shows, can be divided into the following activities:

1. Function *ACQUIRE* (uses function *SWEEP*): the actuations (depending on the time) are expressed depending on the situation of the tip at each instant of the sweep. That way of stating the actuations is represented by the data flow *UGraph*.

2. The compensation of the sample slope: if the sample is sloped with regard to the plane "x" or "y" the topographies will not reflect accurately their relief. Because of that, it is necessary to calculate the offset of the actuations (function *CALC-OFFSET*) both for the "x" axis and the "y" axis. The calculation of the offset is performed with certain regression lines (function *REGRESSION*). Once they are obtained it is necessary to modify all the actuation subtracting them both offsets (function *SUBT-OFFSET*).

3. Finally we must take the new actuations and convert them into pixels for the representation on a screen (function *PAINT*).

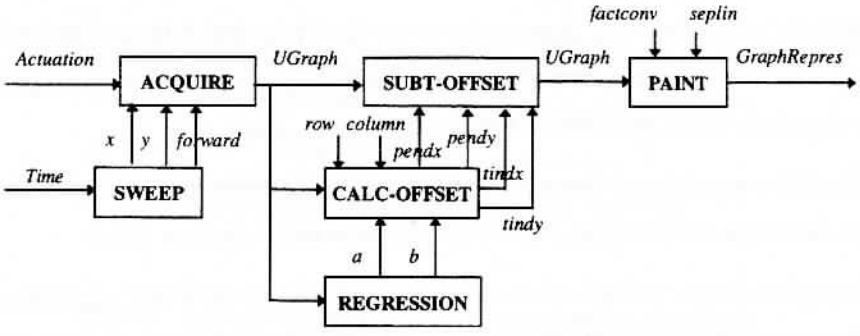

Figure 5. Functional diagram of topography

4. The formal model

4.1 Which formalism is more suited for this case study?

Once the functional model has been developed, the following matter is the choice of the formalism to be used. There are two main classes of formal notation for defining functionality (see Cohen [4]): model-based (like VDM and Z), and algebraic or equational (like OBJ and Larch). Z and OBJ3 were considered as candidates. Both formal approaches were already known and formerly used by several members of the team. Other possibilities were discarded because of the training and time costs implications for the development team. First attention was paid to Z [15], a formal specification language based on set theory and first order logic. An example of Z expressive power can be found in [5], where an oscilloscope is described. Nevertheless, its high abstraction level involves the non availability of prototypes automatic generators efficient enough to execute directly the specification without restrictions. Furthermore, Z does not support properly the concept of modularity [1], what involves problems in maintaining the specifications.

The OBJ3 language [8][10] is based upon *order-sorted* equational algebra (the interested reader can find an effective presentation of these notions in [6] or [7]), and it has not the limitations mentioned above. An OBJ3 specification usually consists of a hierarchy of Abstract Data Types algebraic specifications. The specifications are executable by means of terms rewriting rules, maintaining at the same time a high abstraction level. However, we must recognise that Z expressive power is higher: besides a wide set of operators, Z can express properties over the data elements of a sort and not only over their operations (as it happens in OBJ).

Anyway, the fact that the behaviour of objects of this particular domain can be naturally represented with equational expressions and the availability of its prototyping tool, OBJ3, lead us to choose the algebraic formalism together with the OBJ3 specification language, as the basis for the specification of this case. In previous work, the team successfully used this formalism to validate and verify reactive systems, from a logic perspective [11], as well as to formalise an object-oriented operational prototyping environment for computing-aided software engineering [16].

4.2 Transformation from the functional to the algebraic model

Before translating the functional specification to an algebraic one some criteria should be stated. In this case, for each function that appears in the diagrams one OBJ module with the same name should be included. Within each module the related elements will be modelled. It could also exist OBJ auxiliary modules that do not match with a function of the diagram. The information flows are algebraically modelled as OBJ sorts with the same name, or simply as OBJ functions, depending on its relevance within the model. For example, the information flows that represent the data transformation performed by the system, turn into the sorts *RCurrent*, *Current*, *Error*, *Actuation*, *UGraph* and *GraphRepres* (see figures 4 and 5). The auxiliary information flows, like as x and y (Figure 5), do not produce new sorts and remain simply as OBJ auxiliary functions.

4.3 Brief description of the OBJ3 model

OBJ3 rigorously supports multiple inheritance, exception handling and overloading, together with parameterised programming, a technique which provides powerful support for design, verification, reuse and maintenance. OBJ3 specifications are based on using two kinds of module: *objects*, to encapsulate code and in particular to define abstract data types by initial algebra semantics[5]; and *theories*, to specify both syntactic structure and semantic properties for modules and module interfaces. OBJ also uses *views*, which map modules into theories to prove that a module satisfies a certain theory. In Figure 6 it is presented, in a simplified way, a diagram showing the relationships among the main OBJ modules of the specification. The as-

[5]Roughly speaking the term *initial semantics* refers to the intended meaning of object modules. More precisely, this concept refers to the fact that always we have an algebraic specification AS (as the one denoted by an OBJ3 "program") then there is a model A -the so called *algebra*- such as every element of A can be represented by some term built from the AS, and every ground equation true of A can be proved from the equations in AS.

terisk denotes that the corresponding module is not completely defined, in the sense that it imports another modules which do not appear in this diagram (in particular, the built-in OBJ3 modules: INT, FLOAT, BOOL, etc.).

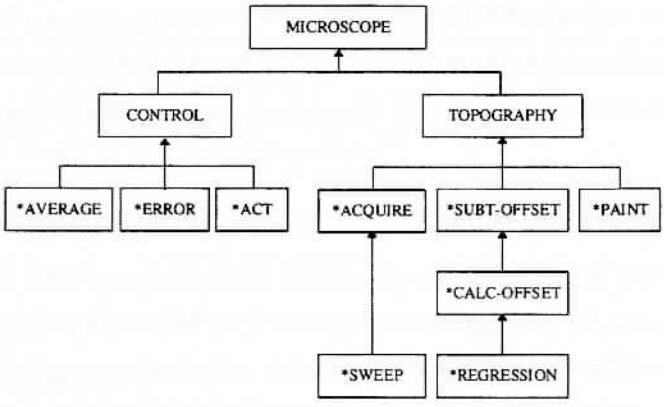

Figure 6. Block Diagram of the main modules

As this figure shows, the module *MICROSCOPE* (which models the basic behaviour of the system) imports the *CONTROL* and *TOPOGRAPHY* ones. The operation defined in *MICROSCOPE* only consists of putting together the operations of the other two modules, that is to say:

```
op microscope : RCurrent -> GraphRepres .
var C : RCurrent .
eq microscope (C) = topography(control(C)) .
```

having defined the sorts *RCurrent* and *GraphRepres* as follows:

```
define RCurrent is (LIST *(op nil to nilcr))[SUB12B] .
define GraphRepres is (LIST *(op nil to nilrg, op (_) to (_::_)))
                           [2TUPLE [SUBPANTX, SUBPANTY] ] .
```

where sort *RCurrent* represents a list of *SUB12B*, which is the module where the sort *Subran12b* (subrange of 12 bits) is defined. This sort characterises all the natural numbers that can be expressed with 12 bits, which is the range of the acquisition and actuation microscope channels. Analogously, the sort *GraphRepres* is defined as a list of pairs of values <x,y>, that have to be within the screen co-ordinates range permitted in abscissas and ordinates (*SUBPANTX*, *SUBPANTY*). Note that the constant operation *nil* is renamed by *nilcr* and *nilrg*, respectively.

The module named *CONTROL* models the control part of the microscope and imports the modules *AVERAGE*, *ERROR* and *ACT*, which specify the functions of the same name. The only operation defined in *CONTROL* is:

```
op control : RCurrent -> Actuation .
var C : RCurrent .
eq control (C) = act (error (average (C))) .
```

where the sort Actuation is defined as follows:

```
define Actuation is (LIST *(op nil to nila, op (_) to (_._)))[SUB12B] .
```

thus, the microscope control system actuations upon the piezoelectric that moves the tip along direction z are also voltage values between 0 and 2^{12}, that is to say, subranges of 12 bits. Both in currents representation (*sorts RCurrent* y *Current*) and in the actuation one, time concept is expressed in a implicit way, by the position of the values in the list.

The module *TOPOGRAPHY* models the necessary treatment of the actuations to be graphically represented. Its only defined operation is:

```
op topography : Actuation -> GraphRepres .
var L : Actuation .
eq topography (L) = paint (subtoff (acquire (L))) .
```

The module *ACQUIRE* needs to use the operations that calculate the position and moving sense of the tip in each instant of the sweep (defined in the module *SWEEP*). The module *SUBT-OFFSET* which offsets the actuation slope imports *CALC-OFFSET* to calculate the offset and this last module itself uses a module that calculates a regression straight line from a list of points.

5. Validation and verification of the model

In this phase, an intensive use of the system prototype is achieved; the prototype is automatically generated from the OBJ3 language (in fact, the OBJ3 specification itself is already an executable prototype). Both activities (validation & verification) are feasible with this kind of prototypes because of the theoretical foundations of OBJ3 and the Algebraic Specifications theory in which OBJ3 is based upon. This is a well founded logic theory where checking a property on a given specification internally lies in formally proving a theorem. In particular, OBJ3 could be seen as a kind of theorem prover; in fact a specific theorem prover has been built on OBJ3, called 2OBJ [9]. A deep formal discussion on the formal features of OBJ3 and 2OBJ is beyond this paper, but it is valuable to know that both the prototype and the activity of reasoning with it are rigorously established.

The process of checking with the model on properties that the system fulfils (or not) according to the users requirements, is called *validation* of the model. As an example, observe that given the PI control equation, something about the knowledge of the actuation trends can be inferred. We mean with trend the increase or decrease in two consecutive actuations. The

property is stated as follows: supposing certain conditions in the consecutive values of current tunnel (as regards they are above or under the reference current), there are two cases in which we know how it will be the trend of the actuation (positive or negative). There are also other two cases in which we cannot know anything a priori about this trend. The fulfilment of the two first cases has been checked with the help of the prototype and it has also been possible to check the truthfulness of a hypothesis about the other two cases.

For example, taking into account that in the control equation it is stated that $q0 > 0$ and $q1 < 0$, if the current in the preceding instant is higher than reference current but in the following instant is lower, we could state that the preceding actuation will be lower than the following one. This feature can be proved by making use of the following operation *val-control1* ("validate control in case1"):

```
op val-control1 : Current Actuation -> Bool .
eq val-control1 (C1,U) = true .
ceq val-control1 (C1 : C2 : C, U1..U2..U) = val-control1 (C2:C, U2..U)
  if (C1 > r) and (C2 < r) and (U1 < U2) .
ceq val-control1 (C1 : C2 : C, U1 .. U2 .. U) = false
  if (C1 > r) and (C2 < r) and (U1 >= U2) .
ceq val-control1 (C1 : C2 : C, U1 .. U) = val-control1 (C2 : C, U)
  if (C1 <= r) or (C2 >= r) .
```

Effectively, if we test the following reduction, using as *average_current* and *Actuations* values calculated by the prototype, we will obtain the expected result, *true:*

```
test reduction val-control1 (average-current, actuations) expect: true.
```

The *verification* of the model is basically the checking of its internal consistency, in this case mainly of its operations. An example of consistency proof is checking that the values of a certain variable are always within its allowed range [12]. This was done to check that during the sweep the microscope tip does not go out of its fixed limits, and that in ordinary conditions the topographies do not exceed the stated screen bounds. Another example of verification is to confirm that a data structure is properly formed, as it was done with the structure *UGraph*. Finally, the last example consists in checking if the effects of an operation are as expected, as it was achieved with the operation that subtracts the sample slope (the *offset*).

6. Conclusions and further work

It has been shown how a functional view of system operations and their representation by means of functional diagrams, may be a very suitable starting point to capture the system requirements. These functional diagrams can be an essential guide to the modelling process.

In relation with this case study, the hardest work has been to convert the functions obtained to OBJ statements. With this formalism, it is possible to specify properties of the system in an abstract way, without the need to express their implementation. However, it is also true that when anyone tries to build an executable prototype, it is necessary to "code" (in a certain sense) the specification. This task may imply a great effort in large models.

Using other formal notations, as Z, it is possible to build more powerful models from the point of view of the expressiveness. However, the important advantage of obtaining executable prototypes is not easily achieved. In spite of that difficulty, the authors finally concluded that probably OBJ3 is more balanced in relation to the trade-off between executability and abstraction, for this case study.

In every case, if the formal method permits the executability of the model, besides observing the results produced by the prototype, it is possible to carry out validations and verifications to check that the model properly captures the system requirements. It also helps to ensure that the system has been built in a consistent way, that is to say, that developers have not done design mistakes.

It is very interesting to have built a "logical model" (as an OBJ3 specification is) where making changes and hypothesis is easier, cheaper and faster than making them on the physical system. As a result of the work presented in this paper, a basic formal model has been provided, in which some interesting system properties of the scanning tunnelling microscope can be analysed and proven.

In future extensions, the formalization of other experiences made with the microscope, such as the approaching of the tip to the sample or the nanolitographic marks upon passive silicon surface, should be embodied. The potential extensions related to formal proofs are numerous, for example the proofs of new properties, making use of the theorem demonstrator 2OBJ, and collecting a "logic samples bank" of the scanning tunnelling microscope properties.

7. References

[1] A. Alencar, J.A. Goguen. "OOZE: An Object Oriented Z Environment". ECOOP '91 Proceedings, vol. 512, pp.180-199.

[2] G. Binning, H. Rohrer, Ch. Gerber, E. Weibel. "Surface Studies by Scanning Tunneling Microscopy". Phys. Rey Left. 49, 57-64. 1982.

[3] Dawn A. Bonnell *et al.* "Scanning Tunneling Microscopy and Spectroscopy, Theory Techniques, and Applications". (Chapter 2 contains the description of microscope design and operation. Chapter

3 and 4 contains fundamentals of topography and spectroscopy with STM). VCH Publishers Inc, 1993.

[4] B. Cohen, W. J. Harwood, M. I. Jackson. "The Specification of Complex Systems". Addison-Wesley, 1986.

[5] N. Delisle, D. Garlan. "A Formal Specification of an Oscilloscope". IEEE Software. September de 1990.

[6] J. A. Goguen. "Parameterized Programming", IEEE Trans. Software Eng., vol. 10, no.5, pp. 528-543, 1.984

[7] J. Goguen, J. Meseguer. "Order-Sorted Algebra I". Technical Report, SRI International, Stanford University, 1988.

[8] J.A.Goguen, T. Winkler, J. Meseguer, K. Futatsugi, J.P. Jouannaud. "Introducing OBJ". SRI-CSL Report (Draft of January 1992).

[9] J.Goguen; A.Stevens; K.Hobley; H.Hilberdink. "2OBJ, a metalogical framework based on equational logic" Philosophical Trans. of the Royal Society, Series A, 339:69-86, 1992

[10] J. Goguen; G. Malcolm. "Algebraic Semantics of Imperative Programs", MIT Press, 1.996 (Chapter 1 contains a good and updated introduction to order-sorted algebra and OBJ3)

[11] B.Grima, A.Toval. "An Algebraic Formalization of the Objectcharts Notation: Validation & Verification of Object-Oriented Reactive Systems" (in Spanish). Proc. of the GULP-PRODE'94 1994. Joint Conference on Declarative Programming. Universidad Politécnica de Valencia, Peñíscola (Valencia). September 1994.

[12] W. A. Halang, B. J. Krämer. "Safety Assurance in Process Control". IEEE Software. January 1994.

[13] J.W. Lynding, S. Skala, J.S. Hubacek, R. Brockenbrough, G. Gammie. "Variable-Temperature Scanning Tunneling Microscope" Rev. Sci. Instrum. Vol. 59 (9), pp.1997-1902. September 1988.

[14] T.L.McCluskey, J.M.Porteous, Y.Naik, C.N.Taylor, S.Jones. "A Requirements Capture Method and its use in an Air Traffic Control Application". Software-Practice and Experience. Vol. 25(1), pp.47-71. January 1995.

[15] J. M. Spivey. "The Z Notation, A Reference Manual". International Series in Computer Science. Prentice Hall International, 1989.

[16] A.Toval, I.Ramos, O.Pastor. "Prototyping Object Oriented Specifications in an Algebraic Environment". In "Database and Expert Systems Applications" (D.Karagianis, Ed.). LNCS nº 856. Springer-Verlag 1994.

[17] D.G.Walmsley. "Pre-microscope Tunnelling -Inspiration or Constraint?" Surface Science 181 pp.1-26. January 1987.