Specifying graphical interaction: Technique, application and correctness proof

Christian Attiogbé
IRIT-Université Paul Sabatier
118, Route de Narbonne
31062 Toulouse Cédex
e-mail: attiogbe@irit.fr

Abstract: We deal with interactive and graphical systems; that is, those allowing users to manipulate data through their graphical representations by using input output devices. Such a system integrates a lot of notions going from software and hardware components of the development environment to the interaction modes with users : selection and direct manipulation of graphical representation of data. The system reacts on external events due to user actions on graphical objects. The effect of these actions is to change the state of the system. Then, an interactive system evolves during a session by changing its state according to external events. Considering these characteristics, we elaborate a formal specification approach intended to allow the specification of interaction when specifying the entire system. The approach is based on the description of the system and its behaviour through the differents steps of evolution. As results, we get for a specification, a collection of structured operational rules offering proof possibilities. An example of specification is given with its correctness proof.

Keywords : Graphical interactive system – Graphical interaction – Formal specification – Visual formalism – Behaviour modelization

Topic : Graphical software specification

1. Introduction

Considering graphical aspects in the specification of interactive and graphical software is still a challenge. We mean by interactive and graphical software, those allowing users to manipulate data through their pictures by using input output devices. A graphical interactive system integrates a lot of notions going from software and hardware components of the development environment to the interaction modes with users (selection and direct manipulation of data pictures). These latter elements are responsible of additional difficulties.

If conception and development aspects have been subject of many works, [6], [4] [5], [11], the formalization aspect is less considered.

Our work can be taken as the research of a formal specification approach allowing to solve the difficulties. The approach is based on the description of the system and its behaviour through the differents steps of evolution. We consider here a system in its globality instead of the behaviour of isolated graphical components as proposed in [3].

Following the idea of structural operational semantics of Plotkin [9], we elaborate an approach which consists in describing the set of specifications of the system behaviour according to the interaction with the external world. The entire specification is then the set of rules associated with each external action. These considerations of rules enable formal reasoning.

The description of the system, its development environment and users's actions are handled. That needs appropriate notations and formalism that will be presented in the paper.

The paper is organized as follows. In section 2) we'll recall the main characteristics and difficulties of the specification. In section 3) we'll present the key concepts introduced and the specification technique. We'll give a specification example and correctness proof in section 4) and in the last section, related and future works.

2. Graphical interactive systems

2.1. Main characteristics

In order to fix the area of our study, we give the main notions characterizing graphical interactive system.

Concerning graphical aspects, note that the system manipulates (creates, displays, modifies) graphical objects which are the pictures of manipulated data or predefinied objects.

In all cases, all these objects can be defined only according to the graphical development environment. Their behaviours are also linked to the kind of behaviour predefined in the development environment.

Concerning interaction with users, they are on one hand, the interaction devices (mouse, keyboard, etc) and on the other hand the designation facilities offered by the environment: most of the time, the interaction is done by selecting the displayed pictures by means of mouse, then the selection and activation of the desired operations through buttons, menus, etc.

About the system behaviour itself, it turns to be, when in a given state, the application of an operation to one or more objects, producing results and then entering into another state in which another operation can be handled.

In addition, we can summarize the characteristics of the system as follows :
– its graphical development environment in physical level in order to take account of interaction devices and logical level in order to exploit defined objects, their behaviour and interaction modes,
– its behaviour according to manipulated objects, its current state and external events indicating the operations of the user.

2.2. Difficulties of formal specification

According to the considered characteristics, what is fundamental is the management of the graphical environment in all aspects. The difficulties come from the description of objects in both internal and external aspects, the description of their behaviour and the impact of external events on the system.` In the opposite of classical transformational systems where the specification technique can just manages how inputs are transformed into outputs as the result of execution, the specification of graphical interactive systems must deal with the formalization of graphical aspects, the control of the evolution of the system and consequently the description of a dynamic semantics expressing the system behaviour.

A specification technique for such a system must deal with all these specificities. How to manage these problems with a convenient abstraction level and without getting into technical considerations ?

We attempt to bring some simple solutions to these interrogations through the concepts presented in the following.

3. The specification approach

3.1. Main concepts introduced

The aim is to find the suitable way to describe the behaviour of the system during its execution. Considering the above defined characterictics, it seems that any system can be described through the set of states by which it transits during its execution. This state can be characterized by:
– the description of objects manipulated by the system and their current characteristics,
– the description of graphical representations of the latter objects,
– the description of graphical objects of the environment and
– the description of interaction devices.

According to this diversity, we introduce for the modelization aspect, three main notions which are : the *State*, the *Memory* and the *View*.

3.1.1. State We talk about STATE when considering the fact that graphical interactive system evolves after the execution of an user action. The notion of STATE describes then the set of informations which characterize the system when it is on an *entry point*.

The notion of **entry point** allows to describe a situation in which the specified system is waiting for an action coming from the external world, that is from user (mouse pression, key stroke, etc).

After the execution of an action, the system can stay in the same State or changes it according to the effect of the action on the information characterizing the State.
the State is defined in the specification process with the following informations :

– the set of internal data or objects of the system,

– the current pannel(s) of buttons (let us note it Butpan),

– the activated cell (noted Cell-c) or activated cells (Cells),

– the current activated menus,

3.1.2. Memory The different objects manipulated by the system are data or internal values. They constitute the Memory of the system. These data exist on different levels : for the user they are seen through graphical representations. For the system, they have internal representations and can be accessed by names or identifiers. The Memory is then the set of objects manipulated by the system in the internal level.

For a set of objects, they are a collection of identifiers, a collection of values (and a collection of pictures for those which can be visualized) with a correspondancy in the semantic level.

The value of an object can be a graphical external form or a functional term describing the construction of the object. For this reason we use abstract data types [7] to describe them and we consider the terms of the types as the objects values.

Then any object can be defined according to the Memory by an equation where the left member is sometimes taken to be the identifier and the right member is the value of the object in the sense defined above. As the value of an object can be its graphical representation, we have graphical forms in equations and this allows us to express selection and direct manipulation in the specification.

3.1.3. View The View is a set of graphical objects. These objects are those belonging to Memory and which can be visualized on the screen. We can see them as the visual part of the Memory. In this sense, The View can be taken as a particular subset of the Memory.

The View of the system is defined in the specification by the following informations :

- the currently manipulated object (we'll note simply *obj-c*),
- the other visual objects manipulated by the system,
- the operation pannels representing available actions.

These three notions STATE, MEMORY and VIEW (S, M, V) constitute sets of object descriptions in the form of *equational system.* The equations have the particularity to include graphical representations (we considered only bidimensional representations) and to allow *deductions* and *equational inferences.*

They constitute the backgroung of the specification method and allow us to specify in a given time the set of objects characterizing the system.

3.1.4. Relationship between STATE, MEMORY and VIEW

Considering the definitions given above, the notion of STATE generalizes the others. But, we need to express distinctly Memory because, it allows us to deal with object constructions or internal representations in the specification. An object can then be referenced through the current State of the system or through its description in the Memory.

The interest is the ability to reason about the dynamic evolution of object during manipulations. Considering an object belonging to the current State implicitly implies that it belongs to the Memory.

About View, we need to define it because in the specification process emphasis is on graphical object and as a specific representation is needed, we have to distinguish graphical objects from others. Note that graphical representation of an object can be used in place of the object itself. So given the View is circonstancially sufficient for a subset of the Memory.

These relations between State, Memory and View justify the notation S_M, V we adopt in the rest of the paper; S_M expresses the inclusion of the Memory in the State and V stands for the requirements on the graphical level.

Then, the specification of a system and its behaviour is written according to theses relations between State, Memory and View, completed by notions and notations we'll introduce.

3.2. Specification technique

The technique is based on the description of the *successive situations* characterizing the evolution of the system during the execution. For this reason we use S, M, V to characterize each *situation* of the evolution, that is the notion of *Configuration.*

The evolution from a situation to another is characterized by the notion of *Transaction.*

3.2.1. Configurations and Transactions

Configuration We call Configuration (noted C), a complete description at a given time, of the system by using S, M and V and we note $C = \langle S_M, V \rangle$.

The specification consists then in describing, going from the initial configuration, the final configuration obtained by the effect of the execution of one or more actions. We get then the set of the system configurations, that is the sequence of configurations obtained when specifying the effect of all available actions on the different entry points.

In a configuration $C = \langle S_M, V \rangle$, V is the set of the specifications of the graphical objects of the system, S_M is the set of objects (graphical or not) belonging to the system (internal representations , menus, button pannels, mouse position, graphical cells or active windows, etc) and M is the set of specifications of the system objects in *abstract syntax* form.

Transaction For us, the notion of *Transaction* stands for a transition involved by an *external action* of the user. There is no relation with other signification for example in Database Systems.

A transaction expresses the transition from a configuration to another. This transition is constrained by the interpretation of *external events.* These latter represent the actions of users on the graphical environment devices.

Since all operations are activated through graphical interfaces, they can be handled in the specification by external events. The transactions we'll be represented by transition rules plus action in the following form:

$$\langle S_M, V\rangle \longrightarrow_{ee} \langle S'_M, V'\rangle$$

where *ee* stands for the external event that leads the system from the configuration $\langle S_M, V\rangle$ to the resulted one $\langle S'_M, V'\rangle$. We presented in [1] a more wide development of interaction handling based on external events.

We introduce an appropriate formalism where inferences are implicit in order to manage equational inferences.

Transaction formalism To make thing clear, we introduce specific notations to represent transactions.

The style is inspired by the operator symbol $\vdash$ " à la Sintzoff "[10]. We use this symbol with the same signification as Sintzoff :

$$\begin{array}{|l} \text{a} \\ \hline \text{b} \end{array} \qquad \text{b is consequence of a, b is deduced from a, a implies b}$$

For the compacity of the expressions, *a* and *b* are not always in predicate form. Thus, the name of an action, say *act*, represents the predicate : "the user applies the action *act*".

Presentation of transactions A transaction we'll be specified in the following manner :

$$\begin{array}{|l} \text{<State, View >} \\ \hline \begin{array}{|l} \text{external evt.} \\ \hline \text{<State' , View'>} \end{array} \end{array}$$

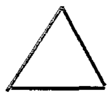
This *rule* (or *operational definition*) has the following signification : if the system is in the configuration $\langle State, View\rangle$ and the *"external evt."* is interpreted then, the system moves to the configuration $\langle State', View'\rangle$.
In such a transaction, we give only the information on which modifications are applied, the others are implicit. In the operational definitions, the terms of State, Memory and View have respectively the form $State \vdash expression$, $Mem \vdash expression$, $View \vdash expression$. That means, the selection of an expression among those characterizing State, Memory, and View.

3.2.2. Algebraic formalization of graphical elements

For View, the elements allowing to handle formally the graphical environment where the system evolves are the following:

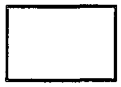
Variables of graphical objects We use these variables to build graphical terms. In the formal point of view, the semantics is that defined for variables in Universal Algebra. Several kinds of graphical objects are used in graphical systems: data or object, cells, windows or applications.

Variable of data



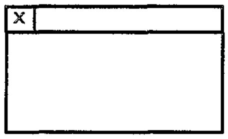
In the algebraic and logic point of view, it is a variable the type of which is graphical data.

Variable of cells



In the algebraic and logic point of view, it is a variable the type of which is graphical cell.

Variable of graphical windows



is a variable of the type graphical window.

Constants

$\underline{\vee}$ $\underline{\vee}$ graphical representations of "released click" and "hold click"

"Mouse cliks" are operators intended to send external events, they are constants. Other events, icons and predefined graphical objects are all graphical constants.

Graphical terms A graphical term is a graphical variable or a composition of graphical terms with an appropriate graphical operator.

Basic graphical operators

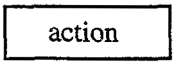
■ Selection operator

↗ Designation operator

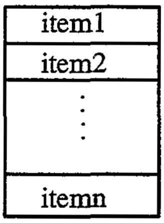
These two operators produce graphical terms when applied to other graphical terms. They allow us to properly write selected graphical objects or designated ones. The inclusion of graphical objects is an implicit graphical operation but the operator hasn't an explicit visual representation. Other graphical operators can be defined to compose graphical terms.

Closed graphical terms

A closed term (or ground term) is a term without variables.



a button



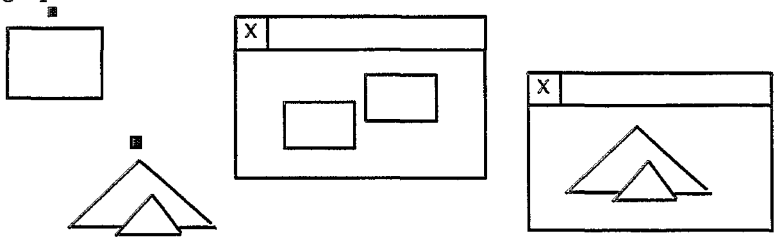
a menu

A button has two components: a graphical one and the other internal which links an action to the button. Formally, the graphical component is defined by a non-closed term when the button hasn't a name.

There is a name variable which we'll take a given name and then we'll have a closed term. A complete menu is a closed graphical term. Indeed, in the formal point of view, a menu is the composition of n buttons (and/or submenus) with the implicit graphical operator of vertical organization.

Non-closed graphical terms

Non-closed graphical terms or just graphical terms if no confusion are graphical terms with variables.



these terms can be combined with graphical operators to produce more complex graphical objects.

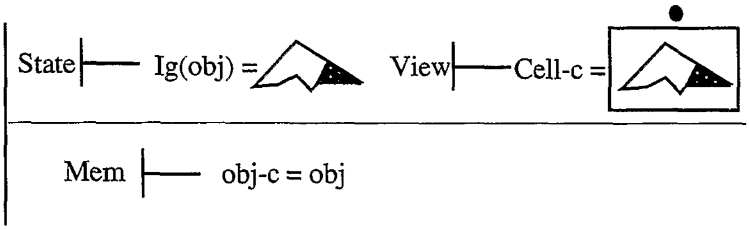
Graphical variables and terms will be instanciated in operational rules for the description of the objects manipulated by the system specified. The transformation of internal representations to graphical ones can be defined by morphisms.

3.2.3. Properties

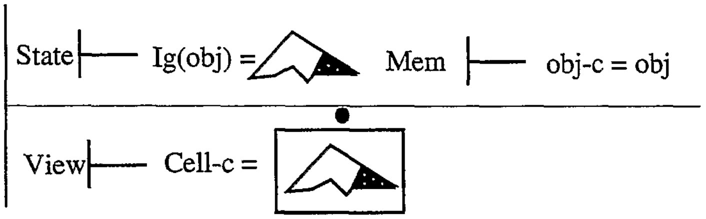
The Memory-View isomorphism By definition, the View is linked with the Memory. The perception of this link is the fact that each object of the Memory which has a graphical representation (in the View) is represented by this unique representation. This latter gives access to the object.

Any action on an object affects its graphical representation and conversely. To formalize this correspondancy between objects and they graphical representations in View, we define the three following rules expressed in the transaction formalism.

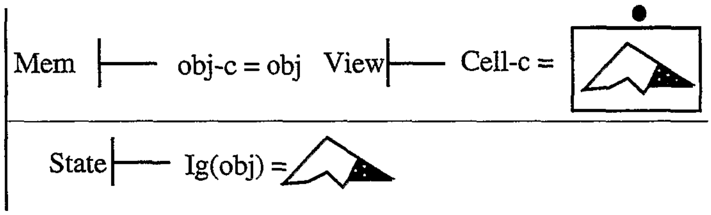
By convention, obj-c represents the current object, Ig a graphical interpretation function which gives for a given object, its graphical representation and t stands for an object of the Memory. The Memory-View isomorphism is defined as follows :



(I1)



(I2)



(I3)

The invariant I1 expresses the fact that the selected graphical object in View is that of the current object in Memory. I2 expresses that to the current object in Memory is the one corresponding to the selected graphical object in View. I3 expresses that if there is a selected graphical object in View and a current object in Memory so the graphical object is the representation of the current object in Memory.

These rules are *invariants* for the specification and consequently, in the other rules, we'll use it without any recall.

Conservation of operations

To handle the history of user operations and the possibility of cancelling them, all operations are recorded. This is expressed as rules in the specification of the system. These rules (also taken to be invariants) are the following:

$$\frac{\text{State} \vdash \text{Seq-oper} = (\beta)}{\dfrac{\text{action}}{\text{State} \vdash \text{Seq-oper} = (\beta; \text{D}(\text{action}))}} \qquad \mathit{(I4)}$$

$$\frac{\text{State} \vdash \text{Seq-oper} = (\beta)}{\text{View} \vdash \text{Hist} = \text{Ig}(\beta)} \qquad \mathit{(I5)}$$

D is a mapping standing for the description of operations. A sequence of elements is noted between brackets and the elements are separated by ';' . Example (**elt1** ; **elt2** ; **...**; **eltn**).

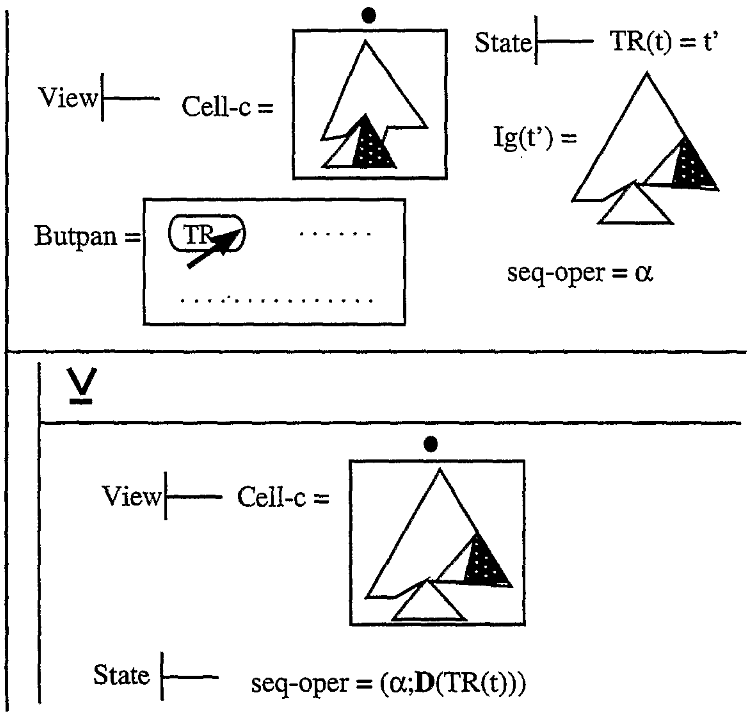
I4 expresses that after each applied action, a description of the executed operation is recorded in the Memory. The invariant I5 expresses that a graphical object $Ig(\beta)$ is associated to any recorded operation sequence β.

4. Applying the technique to specify a system

In this section, we apply the defined notions for the specification of a transaction (a step in the behavior of a given system). Then we give the proof of correctness of the transaction by using invariants. Another application of the technique to specify a graphical interface can be found in [2].

4.1. An example of specification

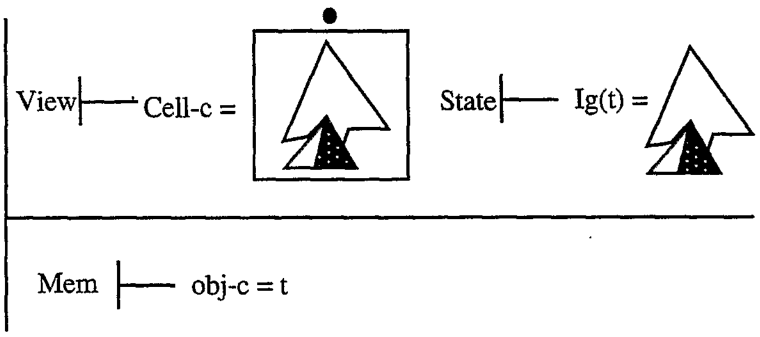
We want to specify the application of a transformation noted **TR** to an object **t**. The following transaction specifies that: from a configuration where the image of **t** is selected and the mouse cursor placed on the button representing **TR**, if an external event occurs then the action associated to the button is applied to the selected image and we get another configuration.



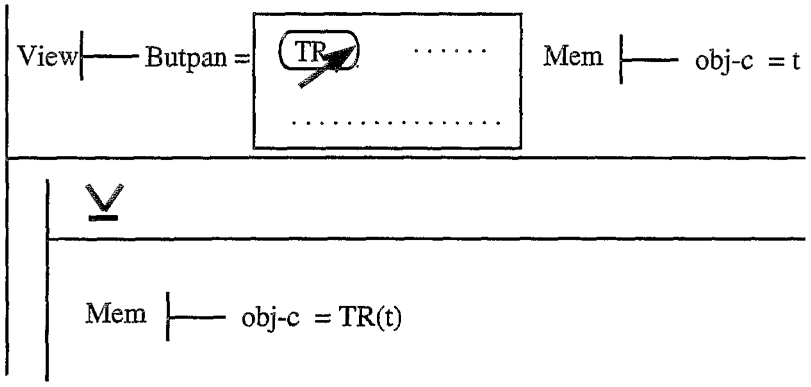
Then we have to prove the correctness of this specification using the invariants we have defined. The idea is to obtain all information contained in the final configuration,going from the initial configuration and by using equational deduction and invariants.

4.2. Proof of correctness

From the initial configuration, if we use the invariant I1 we obtain :



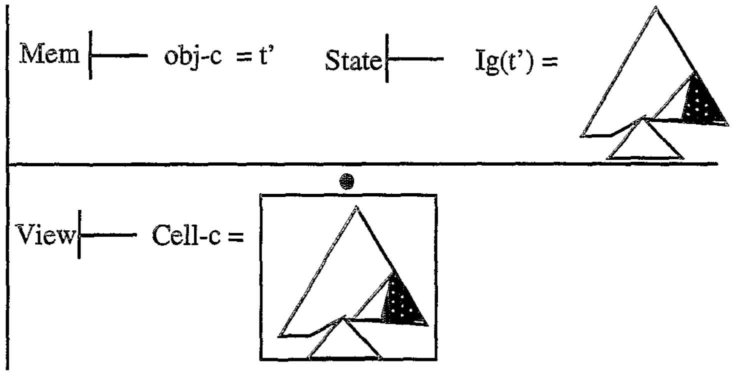
We get an information that we add to those which are in the current configuration. Then, considering the mouse-click effect (remember the operator), we get the following transaction:



Combining this result with the information available in the current State, we obtain :

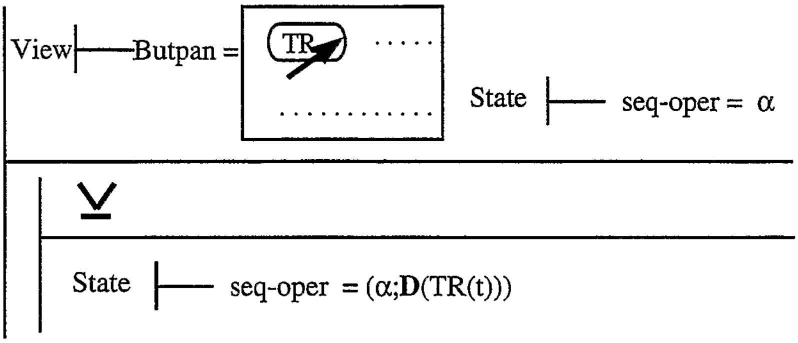
$$\frac{\text{Mem} \vdash \text{obj-c} = \text{TR(t)} \qquad \text{State} \vdash \text{TR(t)} = \text{t'}}{\text{Mem} \vdash \text{obj-c} = \text{t'}}$$

Then, with the invariant I2, we have



where we recognize a part of the informations of the final configuration of our specification.

In the same way, considering invariant I4 (which expresses the record of all operations) we obtain directly from the initial configuration the following result:



In addition, we have found all information contained in the final configuration of the considered specification by deduction and by applying the defined invariants; we conclude that the specification is correct.

5. Related works and concluding remarks

We have presented an approach based on transition rules allowing to formally specify graphical interactive system. The specificity of these rules is perceived by the fact that the formalism we have used is graphical and allows us to manipulate graphical terms which we defined formally as algebraic objects. Graphical aspects of systems are formalized, the link between objects and their graphical representations established. Interaction between the system and its external world are handled with external events on which it reacts. The result of the specification is a set of rules (or operational definitions) giving the description of the behaviour of the specified system. A system of invariants is used for correctness proof with regard to operational semantics of applied actions.

Compared with the most of existing formalisms which are textual, the approach we present, differs by the visual aspect of the formalism and the management of graphical and interactive aspects related to the environment of development. However we must note two approaches which seem more related to ours; In [3], the authors focus their work on the formalization of the behaviour of graphical objects. These sensitive objects are considered as reactive systems which react to user actions. The language Esterel is used as the support of their formalization.

The difference with our approach can be summarized in two points: on one hand they deal with the behaviour of isolated objects instead of an entire system and use a textual language, on the other hand they don't deal with graphical aspects.

The second approach is that of *statecharts* presented in [8]. Here the formalism used is visual too. The specification with *statecharts* is based on state diagrams with notions of concurrency, communication and hierarchy; external events are also used between two states. The difference with our approach is the lack of the formalization of graphical aspect, graphical representation and the link with objects manipulated; however we don't have their notions of concurrency and communication. But this is not our initial motivation and can be the subject of future works as extension of the current.

References

[1] C. Attiogbé and J-L. Durieux. *Handling interaction in software specification.* Actes de " Vienna Conference on Human Computer Interaction / Septembre 1993" — LNCS Vol. 733, 1993.

[2] C. Attiogbé, J-L. Durieux, and L. Lucrèce. Description d'une interface interactive et graphique pour des systèmes de manipulation symbolique et ébauche de spécification formelle. *Actes des JFLA92 - Revue BIGRE,* (76–77), Février 1992.

[3] Dominique Clément and Janet Incerpi. *Specifying the behavior of graphical objects using Esterel.* In Proceedings of the International joint conference on Theory and Practice of Software Development. TAPSOFT'89. LNCS Vol. 352, 1989.

[4] J. Coutaz. *PAC, an implementation Model for Dialog Design*, pages 431–436. Proceedings of INTERACT'87 – Stuttgart, September 1987.

[5] J. Coutaz. *Interface Homme-Ordinateur : Conception et Réalisation.* Dunod, 1990.

[6] P. Franchi-Zannettacci. *Attribute specifications for graphical interface generation.* Rapport de Recherche INRIA, 1989.

[7] I. A. Goguen, J.W. Thatcher, E.G. Wagner, and J.B. Wright. *Abstract Data Types and Initial Algebra and the correctness of Data representations.* Proceedings of conference on Computer Graphics, Pattern recognition and Data structures., 1977.

[8] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[9] G. D. Plotkin. *A Structural approach to Operational Semantics.* DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark, 1981.

[10] M. Sintzoff. Expressing program developments in a design calculus. *Nato ASI Series - Springer Verlag*, F36:343–365, 1987.

[11] Dr. A. Viereck. *A Software Engineering Environment for Developping Human Computer Interfaces.* In the Proceedings of the 7th Interdisciplinaty Workshop on Informatics and Psychology — Visualization in Human-Computer interaction LNCS 439, Schärding, Austria Mai 1988.