Proceedings of the IASTED International Conference
Applied Modelling and Simulation
September 1-3, 1999, Cairns, Australia

Using Adaptive Models for Systems Description

Jorge Rady de Almeida Júnior, João José Neto
Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia de Computação e Sistemas Digitais
e-mails: jrady@pcs.usp.br, jjneto@pcs.usp.br

Keywords: Adaptive Models, Adaptive Statecharts, Systems Modeling

Abstract

Synchronized Statecharts [1] have shown to be adequate devices for expressing most features of real-time and reactive systems. The emphasis is in the explicit statement of orthogonal aspects, such as internal behavior, communication interfaces and synchronization issues of the system being modeled.

This paper explores Adaptive Statecharts, which improves Synchronized Statecharts by including adaptive features used to describe dynamic aspects of the behavior of the target system. The presence of adaptive features in a formal model makes it suitable for the specification of systems whose reactions to external events may change dynamically. So the conceptual difference between adaptive and non-adaptive devices is that the former allows describing in a more natural way those systems capable of modifying their own behavior in response to external stimuli.

Introduction

Adaptive features may be useful whenever the target system spontaneously changes its own set of user-available operations. This is the case with customizable protocols, man-machine interfaces and a wide range of intelligent applications, in particular those with learning abilities or variable features. [2]

Some fields of application of adaptive formalisms are: general context-dependencies; control and supervision systems; on-line graphical interfaces; customizable systems of many kinds; computer art; computer-aided tools; educational; fashion; medical; gaming; learning; tutoring, etc. We illustrate the use of the proposed adaptive statecharts by means of a small case study in the domain of computer art.

The adaptive approach is illustrated through a computer animation example employing an alternative way to represent 2-d scenes made up of moving geometric shapes. In this system users are allowed to temporarily disturb the animated sequence by issuing commands that externally force shapes out of their original path by moving them, rotating them and changing their sizes.

An available tool – ECSS (Evaluation of Critical Systems Specifications) – has been used to test our system. This program allows representing and simulating adaptive statecharts, and upgrades in many aspects an earlier system named STAD (ADaptive Statecharts) [3], [4] e [5]. ECSS and STAD are computer tools for representing and simulating adaptive statecharts. It permits the simulation of the main features of that example.

Based on the results achieved through the experiments made with ECSS, a program prototype named SYSTEM 2D was developed, which retains in its algorithms all adaptive features included in the experiments.

Moving scenes are compressed as a sequence of pictures representing only significant snapshots instead of containing all frames in the movie. This packed format movie is then decompressed by interpolation, at run time. Users are allowed to issue commands and the resulting perturbations modify the original stream by adding new commands, which are dynamically interpreted the same way original frames do.

A good approximation of the sequence of original frames may be obtained by means of our adaptive model, which performs the needed interpolations between consecutive frames in the packed movie, so restoring intermediate snapshots.

The main advantages of this technique is that the compressed movie is expected not only to spend less space in memory and hard disk, and save communication costs, but also to allow users to interact with the animations at run time.

In order to generate and interact with those moving scenes an image editor and a viewing program is needed. In this paper we sketched that viewer to illustrate the use of adaptive statecharts to describe the decompression process while handling the disturbances issued by the user.

SYSTEM 2-D

SYSTEM 2-D allows the visual animation of geometric shapes - points, straight lines, rectangles and circles, forming a movie.

Shapes are grouped together in frames, and the sequence of frames edited by the user defines the expected movie. Movies are run by interpolating shapes between two consecutive frames in the packed movie.

The exhibition of the resulting sequence of frames at some adequate rate gives to an observer the desired animation effect.

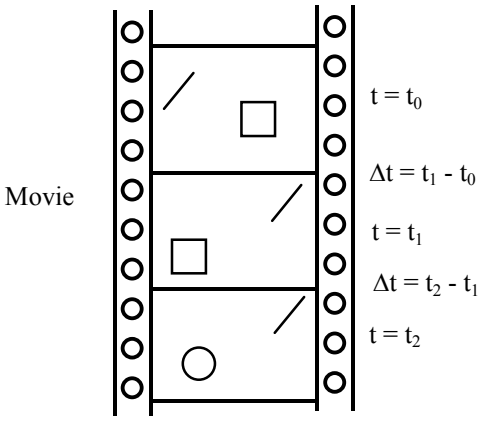

Figure 1 – Movie with frames

The principle of decompression with exhibition purposes is the interpolation made between geometric shapes contained in the frames edited by users. These frames represent the desired sequence. The interval of time between two consecutive frames is one another parameter that must be considered in the interpolation.

Figure 2 presents the basic structure of the proposed system. Basically two modules compose the system: Movie Edition and Movie Exhibition. Data related to all movies inserted in SYSTEM 2D are available in a common database. Exhibition of the movies starts by retrieving the corresponding data from that database.

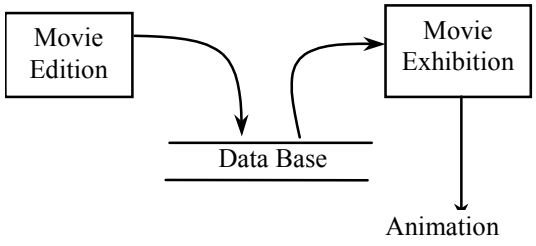

Figure 2 – SYSTEM 2D

One of the distinguished feature of this system is the ability users have to interact with the movie being exhibited. That is made by using the mouse to select an object to be disturbed, and then change of its position in the computer screen by dragging it through a mouse command.

By means of function buttons, users are allowed to select other operations to be performed, such as size modification, color changes, rotation action, fading effects, etc. Once selected the desired action, a click of the mouse over the desired shape will apply the corresponding function to the shape.

Modeling of SYSTEM 2-D

SYSTEM 2D has two sub-modules, as illustrated in figure 3 – an Editor and an Viewer.

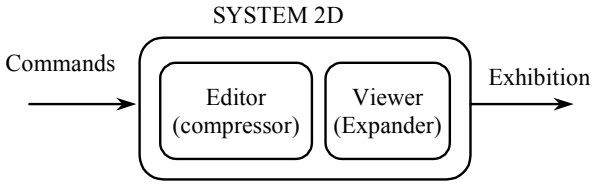

Figure 3 – Statechart of System 2-D

The bubble Editor, as presented in the figure 4, represents the compressor module, which allows the user to specify the basic geometric shapes: points, circles, rectangles and straight lines. The Editor has sub-modules that perform the initial drawing, change the size of the shapes, move them in the frame, delete shapes, apply colors and rotate them.

User must define the name of the film and then edit the desired sequence in the movie. That is specified by means of its initial frame and its corresponding evolution. The viewer module performs the animation of the sequence by presenting its frames sequentially at adequate time intervals.

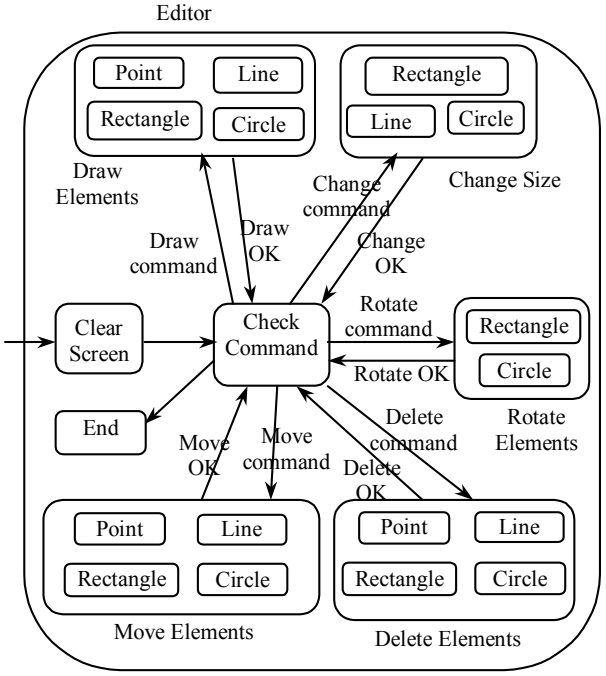

Figure 4 – Expansion of bubble Editor

The editor is composed by the following main table's stores the data of each film:

table Straight Lines: Fields – Name of the movie, number of the straight line, angle, length, x-coordinate of the origin, y-coordinate of the origin, number of the frame

table Rectangles: Fields - Name of the movie, number of the rectangle, length of side 1, length of side 2, x-coordinate of the top left corner, y-coordinate of the top left corner, number of the frame

table Circles: Campos - Name of the movie, number of the circle, ray, x-coordinate of the center, y-coordinate of the center, number of the frame

table Points: Campos - Name of the movie, number of point, x-coordinate, y-coordinate, number of the frame

Data recovery is made by the viewer from the database, as illustrated in figure 5. The Viewer initially identifies the shapes in the frame. Then it makes its interpolation with the previous frame, exhibiting the resulting sequence of frames. The Viewer executes the proper frame animation by sequencing the resulting frames with an adequate timing.

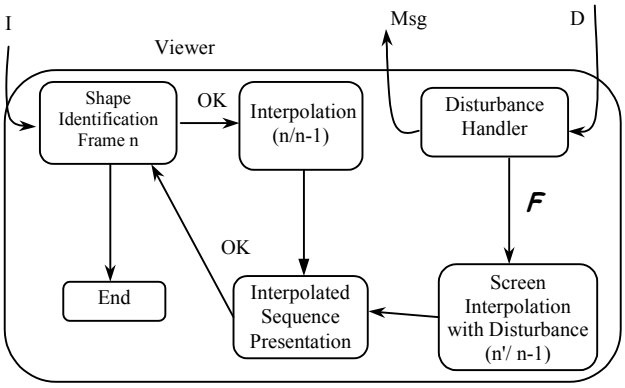

Figure 5 – Expansion of bubble Viewer

The sequence of actions that are executed by the Viewer for each take in the movie is:

- Identification of all shapes in frame n (in the packed movie)
- Drawing of all objects of screen n
- Identification of all shapes in frame n+1
- Interpolation of intermediate frames, according to the duration of the current take and the selected refresh time for the frames
- Drawing of the interpolated frames

Figure 6 illustrates the assembly of the exhibition statechart of an interpolated sequence.

Disturbance handling is essential to the implementation of dynamic features imposed by the user interactive disturbances.

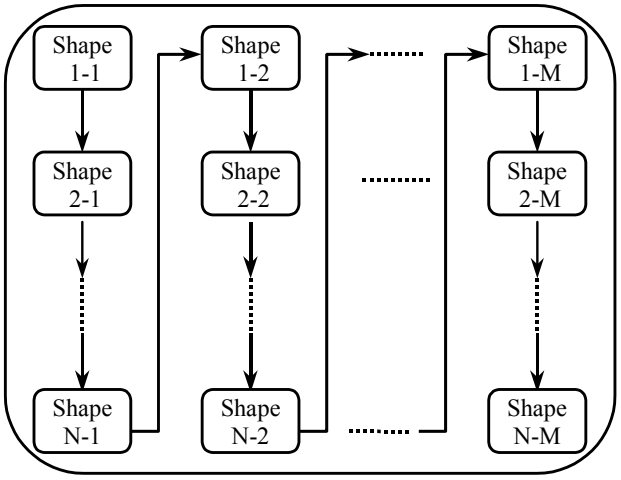

Figure 6 –Interpolated Sequence Presentation

While shapes are moving, user may change their positions by dragging them with the mouse. This action affects the original frame sequence. Such a disturbance forces the system to handle that event as modeled in the statechart of figure 7.

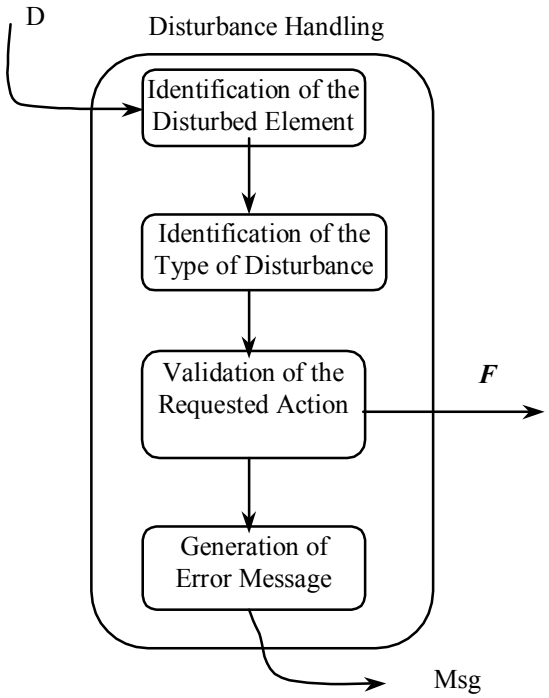

Figure 7 – Bubble Disturbance Handler

The disturbance is recognized and handled by the module Disturbance Handler, and a new interpolation is accomplished (new frame - with disturbance / previous frame). The new interpolated sequence is exhibited to the user. The modified sequence of frames starts at the moment of the disturbance, and replaces the former one. This replacement is implemented by performing an adaptive action.

This adaptive action causes the creation an adequate number of groups to correct the original sequence of frames for the affected shape. Each new basic group is composed by a new bubble, equivalent to a new state of an automaton and by two new connections. There is also a basic group that is suppressed from the original statechart that represents the initial interpolation. One bubble and two connections are deleted from that original sequence. These are replaced by the elements added by the new basic group.

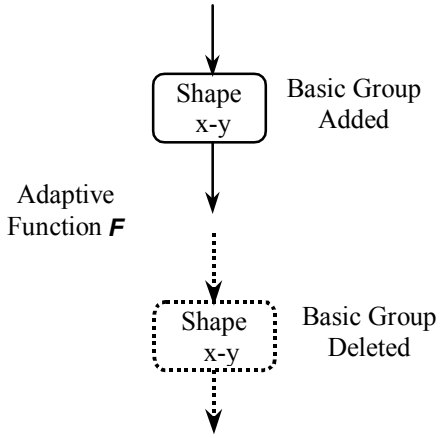

Figure 8 – Adaptive Function F

The execution of an adaptive action is illustrated in the example described below.

The disturbance handling occurs by first identifying the disturbed element, as well as the disturbance type. Then the path change is tested for adequacy. For possible changes, the adaptive function is activated. Otherwise, an error message is generated, indicating the occurrence to the operator.

The complete modeling of this example was performed by using ECSS (figure 9). Figure 10 presents a work frame of SYSTEM 2D.

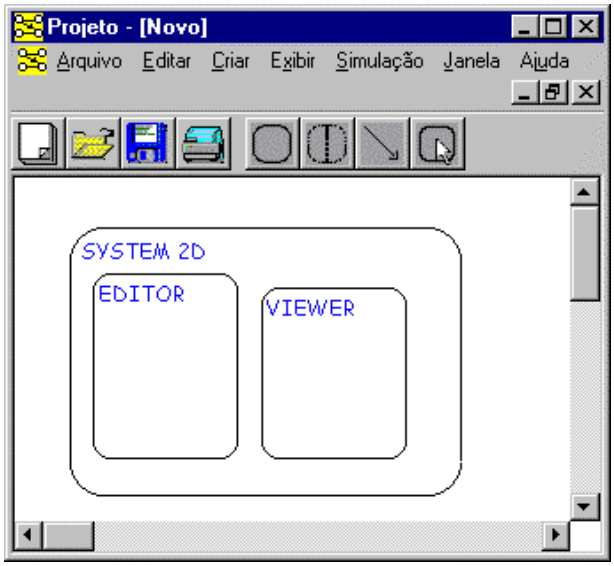

Figure 9 – Modeling of System 2-D through ECSS

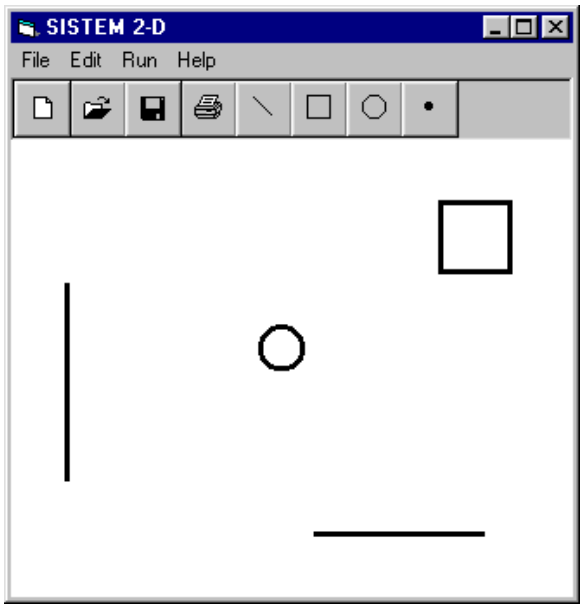

Figure 10 – A Frame in SYSTEM 2D

Example

This example refers to figure 11, which shows two frames represented by the user. Those frames contain two points, one in coordinate (0,0) and another in coordinate (5,5). The corresponding displacement of shapes should be accomplished in 5 seconds.

Figure 12 shows a trace of the sequence of positions assumed by the two points. At the instant $t + \Delta t$ seconds, the point number 1 was moved by the user through a mouse action from its coordinate (0,2) to the coordinate (3,2). This is a disturbance and the trajectory of point 1 has to be corrected. In the figure 12 we can see also the new calculated trajectory of point 1, starting from the coordinate it had at the time the disturbance was applied.

The adaptive statechart representing the original undisturbed sequence of frames is presented in figure 13. The resulting statechart after handling the occurrence of the disturbance is presented in figure 14.

It should be noticed that starting from coordinate (0,2) for the point 1, the adaptive function starts by removing three basic groups associated to point 1, and adding three new basic groups related to the same point. The execution of the adaptive action causes the correction of the trajectory that was modified by the disturbance imposed by the operator.

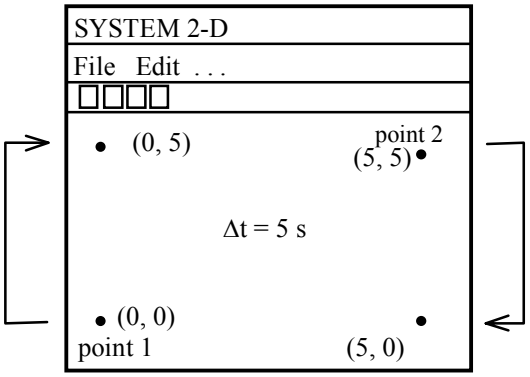

Figure 11 – Example of a movement between two screens

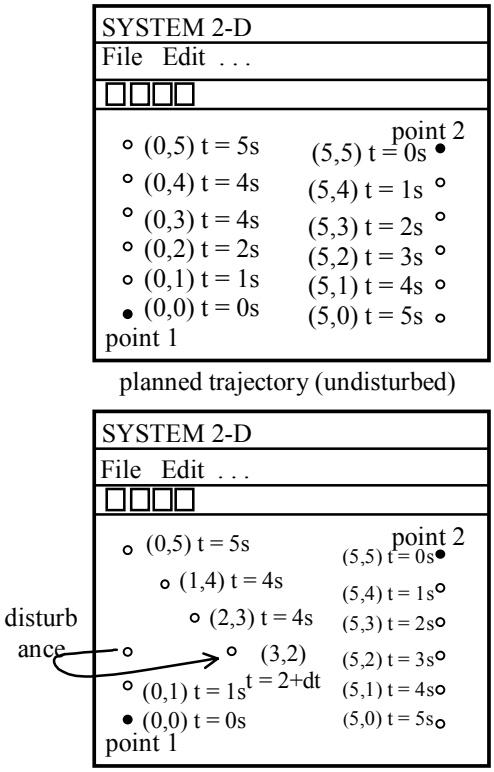

Figure 12 – Example of a movement between two frames with a disturbance (empty dots represent successive coordinates in the path of the points)

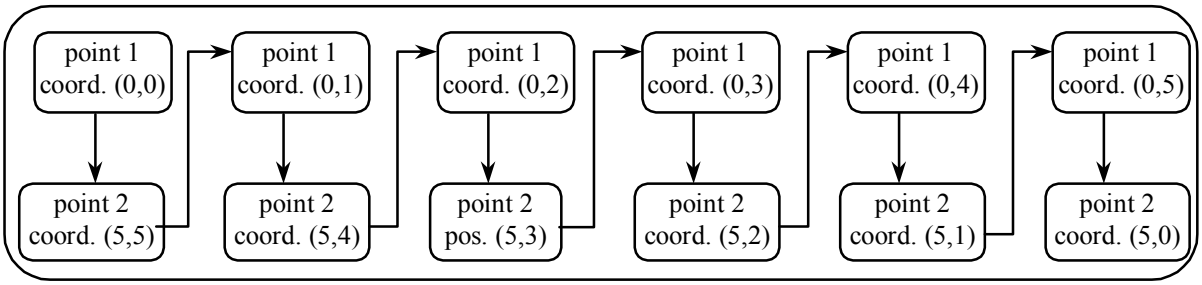

Figure 13 – Statechart of the Example without disturbance

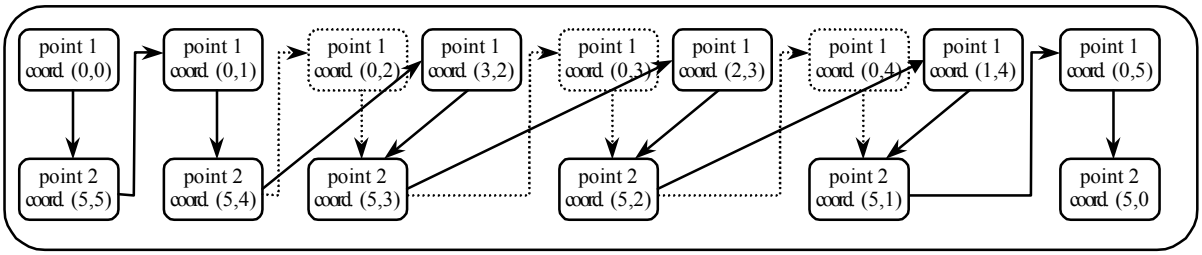

Figure 14 – Statechart of the Example after the Handling of a Disturbance

(dashed lines indicate deleted elements)

Conclusions

The main advantages of this technique is that the compressed moving scene is expected not only to spend less space in memory and hard disk, and save transmission costs, but also to allow users to interact with animations. The program built in agreement with the adaptive technique allows checking the effectiveness of this modeling type.

Similar applications of adaptive devices may be used, for example, to simulate member movement, for handicapped people, in athlete and choreographic training, in pedagogical objectives, in the production of cartoons, children learning, and so on.

One characteristic not yet implemented in SYSTEM 2D is the creation of a module that can assembly a movie from sequences extracted from already existing frames (in several formats). This module could assemble screens stored in the database, determining and extracting relevant frames from the original movie, capturing automatically the relevant screens only.

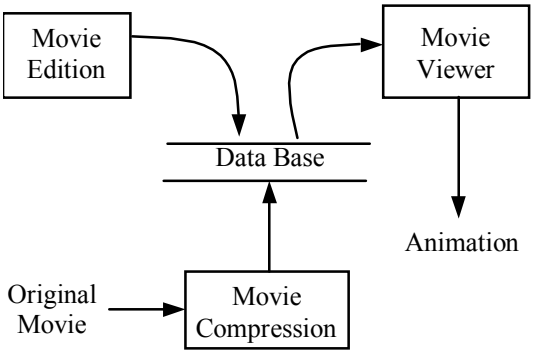

Figure 15 – SYSTEM 2-D

Another evolution of SYSTEM 2D is the automatic execution of semantic actions, starting from the modeling tool (ECSS). Semantic actions could include move, delete, create and rotate shapes. With this type of implementation ECSS will become a tool with prototyping capacity.

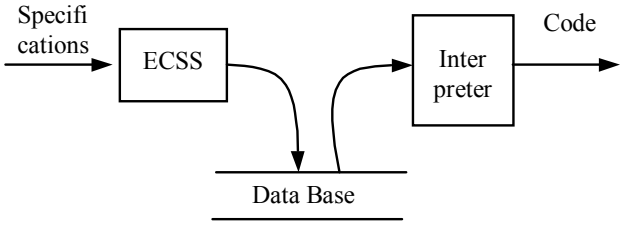

Figure 16 – Advanced SYSTEM 2-D

Another characteristic to be implemented is the possibility of several types of movements of the objects in the user's frames, such sinusoidal, movement speed alternation of the objects, such as fast in the beginning and slow in the end, and vice-versa.

References

[1] J. José Neto, J. R. Almeida Jr. e J. M. Santos, Synchronised Statecharts for Reactive Systems, *Applied Modelling and Simulation,* 1998, Honolulu, Hawaii, USA, 246-251

[2] R. S. Rubinstein, J. N. .Shutt, Self-modifying finite automata: An introduction, Information *Processing Letters*, v.56, n.4, 24, p.185-190, 1995.

[3] J. R. Almeida Jr. e J. B. Camargo Jr., ECSS - A Tool using Adaptive Statecharts for Evaluation of Critical Systems Specifications, to be published in *17th International System Safety Conference,* 1999, Orlando, USA.

[4] A. Sowmya, S. Ramesh, Extending Statecharts with Temporal Logic, *IEEE Transactions on Software Engineering*, vol. 24, no.3, march 1998, p. 216-231.

[5] O. Maler, Z. Manna, A. Pnueli, From Timed to Hybrid Systems, *Proceedings of the REX Workshop on Real Time: Theory and Practice*, 1991, Springer Verlag, New York.