Object Oriented Tool to model and simulate discrete event systems

R. Ferreiro García, X. Pardo Martínez, J. Vidal Paz
Dep. Ind. Eng. University of A Coruna
{ferreiro, pardo, vidal}@udc.es
E.S. Marina Civil, Paseo de Ronda, 51. 15011-A Coruna

Abstract

This paper illustrates the methodology to develop a real time object oriented tool (RTOOT) destined to design and implement SFC based applications by means of an object oriented virtual engineering environment, the HP-VEE. Arrangement of proposed RTOOT consists in developing a library which comprises a set of user objects capable for implementing the mathematical model of sequential function charts on the basis of Petri Nets, being IEC 1131-3 standard language compliance [4]

Introduction

Developing SFC (sequential function charts) editors requires the ability to model Petri nets based function charts. The math-model of a function chart is derived from that of Petri nets [5]. A function chart is a direct graph defined as a quadruplet:

[X, T, L, X_0],

where

X = (x_1, ...x_m) is a finite, non empty set of steps

T = (t_1, ...t_n) is a finite, non-empty set set of transitions. X and T represents the nodes of the graph.

L = (l_1, ...l_p) is a finite, non empty set of directed links, linking either a step to a transition or a transition to a step.

X_0, which belongs to X, is the set of initial steps.

These steps are activated at the beginning of the process and determine the initial situation. Moreover, the graph is interpreted, meaning that

- with the steps, commands or actions are associated.
- with each transition, a logic transition condition is associated.

Steps are represented by labelled squares, transitions by dashes. In addition to the static representation, the graph also has a dynamic aspect defined by evolution rules. The ability of described math-model structure to manage complex control and supervision or diagnostic tasks, as most important the following:

- sequence level supervision
- modelling and simulation of discrete event systems
- monitoring and diagnosis of sequential processes included batch processes
- operating procedures representation

- large data-base partitioning
- general procedures in control and control representation for sequential reasoning.

In a general context, the global task may be described by an SFC representation, one of the graphic language of the IEC-1131-3 standard.

Object Based Library Design

The proposed library comprises a set of user objects useful to implement sequential function charts. The architecture of a SFC graphic interface is developed by means of four user objects defined by object blocks [1,2,6]:

Step & transition
Parallel branching & transition
Optional divergence branching & transition
Optional convergence branching
The symbols adopted for proposed user objects are shown in figure 1

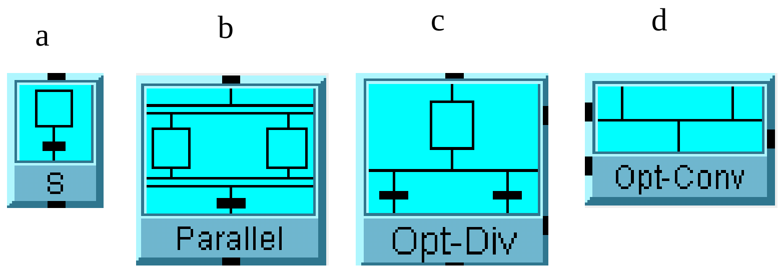

Fig. 1. Symbols associated with the user objects to implement graphic SFC's

Object block (a) is a user object capable to support the definition of any type of action and its associated transition.

Object block (b) is a user object capable to support parallel sequencing of any kind of actions, that means multithread tasks which comprises sequential actions associated to its own transition, where every thread is executed with independence of each other and when the last action of every thread is executed, then, a general condition for transition is true and the multitask is completed.

Object block (c) is a user object capable to support any kind of action similar to the case of the object block (a). The essential difference is in the evaluation of the condition for transition. That is, when condition for transition is true, a selector is implemented to decide which of several links is to be selected according the definition of the transition.

Object block (d) is a user object empty of actuation, and it is responsible to link any convergent branch of SFC.

User Object Concatenation

Concatenation of described objects is based in the IEC 1131-3 standard, which means that there must be beginning and end blocks, and any action is followed by at least a transition and every transition must be followed by an action or an end block. A feedback link arrangement is also supplied to ensure the closed loop interaction cycle until an end command is encountered. Figure 2 shows two different common sequences, an optional branch with divergence and convergence and a parallel branch. The parallel branch object is capable to define and support all the attributes of a macro stage. Both optional and parallel branches are coupled into a sequence of object blocks

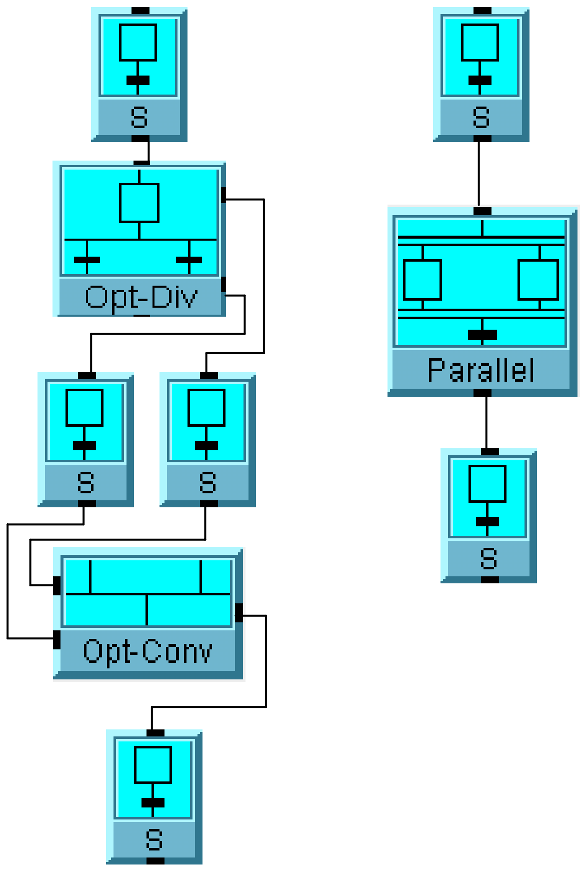

Fig. 2. The main sequences: optional and parallel branches

Application on a Batch Process Control Problem

In order to illustrate the procedure to implement a sequence of control actions, let us consider the storage/heater tank described in figure 3.

Two activities can be performed, simultaneously or not, on the storage/heater tank: The loading operation and the heating operation. The end of operation conditions depends on the liquid volume to be heated.

The model of the storage/heating tank described in figure 3 is composed of four discrete states. The states correspond to the following configurations of the storage/heating tank:

- Inactive state
- Loading phase
- Loading/heating phase. A simultaneous set of actions
- Discharging phase.

The operation in normal conditions is a sequence of actions described as:

- Filling the tank till a minimum level
- Filling and heating till the reference temperature and level

- Discharging the tank till the zero level.

Both, the SFC and the object based SFC for the mentioned sequence are shown in figure 4

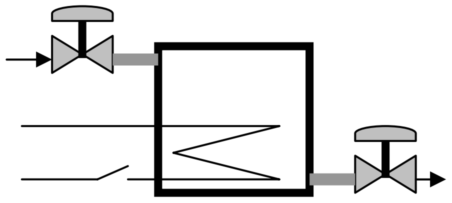

Fig. 3. The storage/heating tank as an application to implement Object based SFC

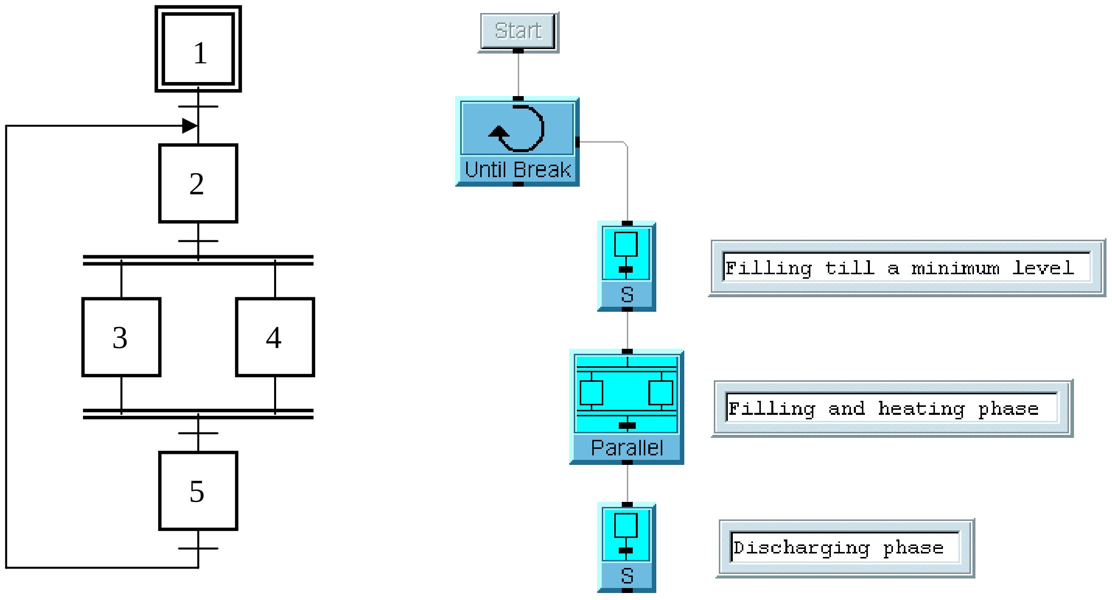

Fig. 4. Left: SFC for the storage/heating tank. Right: Object based SFC

Results and Conclusions

Proposed object oriented tool to implement object based SFC's is developed under a virtual engineering environment (HP-VEE), and object oriented SFC has been implemented for the described platform.

In figure 5 we can see two threads executing at the same time: in the left is the object based SFC, and in the right are the data results of the simulation.

Figure 6 shows the implementation of the filling tank algorithm. It corresponds with the stage 2 of the SFC.

Figure 7 shows the stages 3 and 4 executing in parallel. We define two threads, one for the filling phase and the other for the heating phase, and we syncronize them using a semaphore technique. When both threads finish, the stage 5 starts (discharging phase) until the tank is empty, at this moment the last transition is verified and stage 2 is activated.

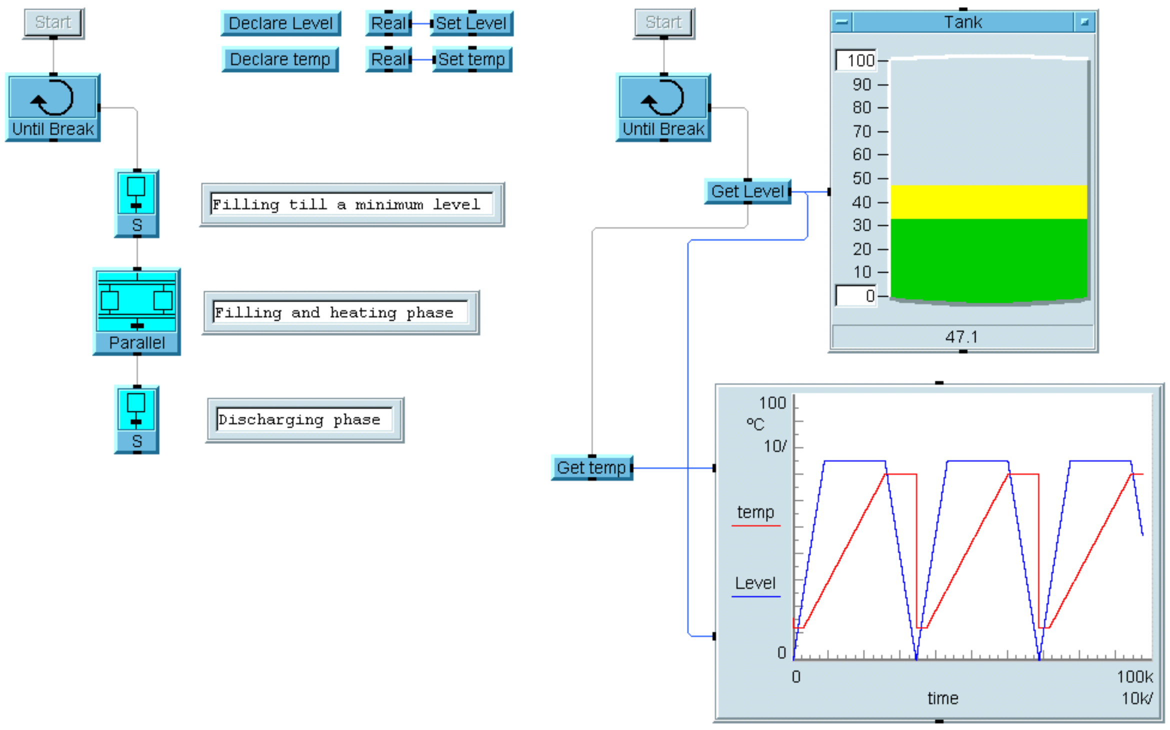

Fig. 5. Object based SFC and data results of the simulation

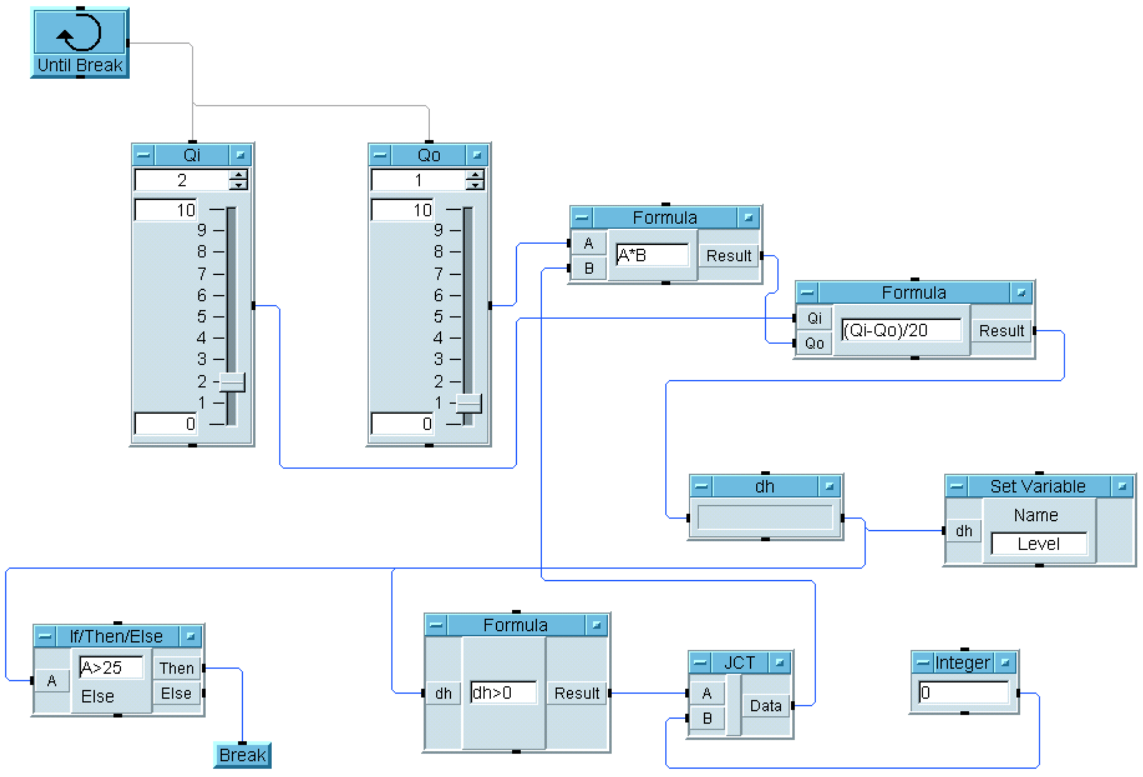

Fig. 6. The stage 2: filling tank algorithm

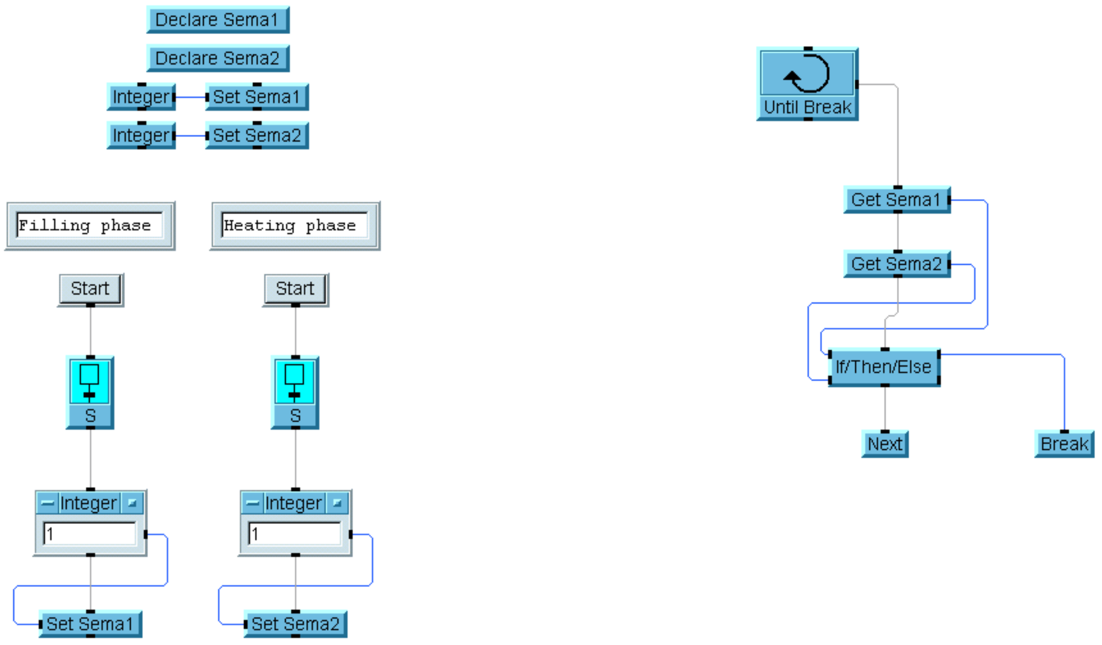

Fig. 7. The stages 3 and 4 of the parallel branching

The ability of object based block to define control actions associated to transitions has revealed as useful not only in virtual processes or problems but also in diagnostic tasks. Complex algorithms can be solved due to the algorithmic capacity of HP-VEE and Matlab libraries using active X controls. Counting and timing is a simple task provided that these blocks are included in the HP-VEE libraries.

References

[1] K.E. Arzen, *Sequential function charts for knowledge-based real-time applications*, Proceedings of the Third IFAC Workshop on AI in Real-Time Control, California, 1991

[2] R. David, H. Alla, *Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems*, Prentice Hall, London, 1992

[3] R. Helsel, *Visual Programming with HP-VEE*, Hewlett-Packard Professional Books, Prentice Hall, New Jersey, 1997

[4] *International Standard 1131-3* (1993), Bureau Central de la Commiusion Electrotechnique Internationale 3,rue de Varembé, Geneve Suisse.

[5] Kyung-Huy Lee and Moo-Young Jung (1995). *Flexible process sequencing using Petri Net Theory*. Computers Ind. Engng. Vol. 28, No 2, pp 279-290

[6] L. Marcé, P. Le Parc, *Defining the semantics of languages for programmable controllers with synchronous processes*, Control Engineering Practice 1 (1993) 79-84