A Virtual Verification and Execution of Grafcet Using VRML.

Authors:
Mohamed MHAMDI, LIP2, Faculté des Sciences de Tunis, mohamed.mhamdi@isetso.rnu.tn
Mohamed MOALLA, LIP2, Faculté des Sciences de Tunis, mohamed.moalla@fst.rnu.tn

Abstract — A systemic approach is privileged in Automatics. A given system design description must be structured in an operative part and a sequence control part. From the system point of view, the GRAFCET (French Standard NF C03-190/ European Standard CEI IEC 848) is today a well known and recognized graphic tool in educational and industrial world for describing the specification of a sequence control system. In a teaching context the operative part is often represented by didactic components. Therefore a student must: a) Elaborate the Grafcet according to a system viewpoint; b) Implement the whole or part of the Grafcet on a PLC; c) Validate the behavior of the operative part according to the implementation mentioned in b).

In this paper we describe a system allowing the accomplishment of these three tasks without the need of a real PLC nor an operative part. A visual simulation is possible by using a VRML viewer to validate the conformity of the Grafcet with the awaited automatism behavior. The virtual didactic models are stored in a library of reusable components. The reusability of components is based on the following points: a) The 3-D description of the components is made by using the standard language VRML, which makes them portable on many types of platforms. b) The storage of the components in a spatiotemporal database based on a relational model gives to them a modular structure that facilitates reusability. c) The component is a software component that can be used in a virtual environment (simulation) or a real environment (Co-simulation). It can also be combined with different types of sequence control systems in order to create new types of automatism.

Index Terms — Grafcet, Vrml, Spatiotemporal DataBase, Sequence Control System, virtual reality, 3D Simulation, Concurrent Execution, Web-Based Teaching, Synchrone Language.

INTRODUCTION

Actually Grafcet is becoming a means of communication between the developer and his customer, it is used as a tool for the specification of the sequence control systems. However one of the strong points of Grafcet is the easiness of passing from the model to the technological implementation thanks to the perfecting offered by the Grafcet. After being a language of specification, Grafcet becomes then a language of implementation of sequence control systems. Consequently we speak nowadays about: Grafcet according to a system point of view, Grafcet according to an operational point of view and Grafcet according to a sequence control point of view.

In a teaching context, the operative part is often represented by didactic components. Therefore a student must: a) Elaborate the Grafcet according to a system viewpoint; b) Implement the whole or part of the Grafcet on a PLC; c) Validate the behavior of the operative part according to the implementation mentioned in b).

During the realization of these three tasks, the student meets some difficulties from different origins:

- Difficulties due to the automatic system like: the identification of the system input/output matters, the coordination of the operative tasks and the definition of the relative movements of subsets kinematically linked.
- Difficulties due to the operative part like: the identification of the actuators, pre-actuators and sensors.
- Difficulties due to the sequence control part like: the realization of the logical equations associated to the transition or ; the identification of the parallelism and the alternative sequences.
- Difficulties related to the dialogue between the operator and the sequence control part like the identification of the components of the dialogue and their functions.
- Difficulties related to the implementation of the automated system like: the connection of a pre-actuator to an actuator or a sensor to a PLC Input/Output module and the Implementation on a PLC the whole or a part of Grafcet.

To solve these problems the ideal is to place the student in a real environment of training, thing which is not always possible[1]. The principal idea is then to be able to realize an integrated environment, entirely configurable, allowing creation and development of programs in a simulation mode. Our system includes an Implementation of a “Grafcet Engine” and a virtual simulation of Operative Parts in 3D. However, to realize this system we should treat two types of problems which are:

- Problems related to the technique of simulation which has the following technical disadvantages:
  - No standard language really exists to describe the models of the system components which have dynamic behaviors such as engines, arms manipulators etc.
  - No one proposes reference models to represent the system components and their complex dynamic behaviors.
Great and long effort must be done by the programmer in order to implement the components of automated systems.

Discrete event simulators and robot simulators are expensive. Moreover, the simulation environments of many products are not open and they are limited to some particular computer and operating system platforms.

The developer cannot easily change the behavior and the geometric level detail of models according to the desired precision.

In order to solve these problems, the standard language, the reference model, the inexpensive environment and its independence of all platform types are strongly needed. To satisfy these needs we find that a simulation system based on a VRML viewer is a good choice. Indeed VRML can be employed like a component modeling language (operative part) and the simulation can be run via a VRML viewer.

Problems related to the GRAFCET:

The expression power of Grafcet led the industrialists to develop in their material many Grafcet languages. None of these implementations has the same semantic behavior. Especially these languages interpret the base rules of Grafcet and integrate badly the assumption of zero-time by the model of evolution. This has multiple consequences: firstly the users can confuse the industrial language and model; secondly the Grafcets are not compatible between the industrialists and finally it is possible to program systems specified by Grafcet in different ways by using different Grafcet languages[2].

To circumvent some of these problems, we have conceived and implemented in our system a “Grafcet engine”. This engine can be considered as a training tool of Grafcet and automatism, in a simple way and without worrying about the evolution and the heterogeneity of the industrial languages.

THE GENERAL ARCHITECTURE OF THE SYSTEM.

Based on an integrated approach, Figure 1 shows the general architecture of the Grafcet and automatics training system that we have realized. After knowing that the components simulated by our system are those that we find mainly in the laboratories of teaching and which are generally didactic components, The problems arising from such architecture are:

- How to model and implement in VRML the virtual didactic components knowing that each one has a specific geometry and behavior.
- How to store the VRML implementation of components within a reusable and a public component library (Data Base).
- How to represent the virtual model as a software component which can be used in a virtual environment, and which can be also combined with different types of sequence control systems in order to create new types of automatism.
- How to model and implement a Grafcet engine in a VRML world.

The object of the following sections is to present the adopted solutions of each type of problem.

VRML AND GRAFCET.

Till now, the classic method to implement the sequence control systems specified by Grafcet, is to use PLC. In this section our aim is to replace the PLC by a programmable component (for the simulation). We think that the sequential core of PLC as the input/output modules can be replaced by a specific programmable component designed in Grafcet with keeping the power that characterize the input/output of the PLC. This is done through what we have called “Grafcet Engine”.

For the modeling of the Grafcet engine two approaches are possible:

- Modeling based on the local Grafcet evolution rules: this approach consists in expressing the rules in concurrent equations. The translation is simple and systematic.
- Modeling based on the translation of Grafcet to an other model: This approach consists in translating Grafcet into an other model characterized by the power of the mathematical aspect, or formal tools. Particularly, some works based on this approach are already performed[10] (finite state automata, Petri nets, transition systems and signal language). In this section, we are interested by the first approach for the simple reason that we are interested by only the simulation aspect where the implementation of local Grafcet evolution rules is sufficient.

Here we present the five Grafcet evolution rules:

- Rule 1: at the beginning the initial steps are active.
- Rule 2: a transition can be fired if its condition is true and if all the preceding steps are active.
- Rule 3: when a transition is fired, all the preceding steps are simultaneously desactivated and all the following steps are simultaneously activated.
- Rule 4: if several transitions can be fired, they are all fired simultaneously.
- Rule 5: if a step is simultaneously activated and deactivated, it remains active.

To implement the engine in a VRML environment it should be noted that VRML 97 allows the programming using different high-level languages. In practice, the standard defines only the interface between the viewer (Cosmo Player for
example) and the program which is written in another language. Thus, in theory, any language (C/C++/Java...) can be implemented by a viewer[15][16].

Accordingly to this, we present a first engine implementation based on the VRMLScript language which is supported by the majority of VRML viewers and we present too, a second implementation based on an extension of VRML via the concept of “Reactive Script”. In both next sections the illustration of the two implementations will be done based on the example shown by figure 2. It represents a producer-consumer model. The “I1” condition implies the production of an element and the “I2” conditions implies its consumption.

The Implementation of the Engine in VRMLScript.

This approach consists in writing the local Grafcet evolution rules as concurrent VRML equations. The fourth evolution rule says that all simultaneously clearable transitions must be immediately and simultaneously cleared. Then for each transition “ti” we can write a VRML equation “Ti” to represent the firing of the transition. As it has been specified by the second evolution rule, it depends on transition condition (Ri) on conjunction of preceding steps “Pre(ti)”:

\[ Ti = Pre(ti) \land Ri \]

The third evolution rule specifies the consequences of a transition clearing according to preceding steps (deactivated) and the succeeding steps (activated). Thus we can formulate the activation equation “Xi” for each step “Xi” as:

\[ Xi = \neg succ(xi) \land (Xi \lor Pre(xi)) \]

Obviously, these equations suppose the existence of a stable state, if not, they will have a non deterministic behavior. In order to be sure that the actions are stable during a constant time and that they are deactivated during a reset time, the synchronization is done according to a sequential process outputs.

VRML is based on a discrete time model, through a special node called "time sensors". Such type of node permit us to implement the execution of the Grafcet engine. Being active "Time Sensor " generates a temporal event at the beginning of each cycle which activates the execution of script node gathering the transition equations and those of step activation. The frequency of "Time Sensor " must be sufficiently small to guarantee the access to a stable state. The period of "Time Sensor " must be higher or equal to the maximum reaction time. Figure 3 shows a summary of VRML scripts which represents the implementation of the Grafcet engine relative to the example of figure 2.

The Implementation of Engine in Reactive Script.

The principal problem of the previous solution is that we have supposed that a stable situation exists. But the synchronous languages like Esterel propose an approach for resolving the problem of boolean stability. Since the synchronous hypothesis is at the base of Grafcet, it will be more suitable then to implement it in a synchronous language like Estrel[7]. Based on steps and transitions, the Grafcet formalism integrates the principal characteristics of Esterel: the hierarchy by the embedding of macro-step, parallelism by the divergence in "AND", the determinism by the limitation of the number of accessible states, the concurrency by the description of independent Grafcet in the same macro-step and the communication by instantaneous diffusion of signals. A grafcet can then be translated automatically in an Estrel program.

The proposed translation considers each step of the Grafcet as a reactive system. Each step is able to react instantaneously to a state modification generated by other steps. Thus, the third evolution rule of Grafcet applied to steps 3, 4, and 5 of the example can be written in Estrel as:

```
present RTL3 and RTL5 and not I2 then
emit X4_SET;
emit X3_RESET;
emit X5_RESET;
end present;
```

Where Xi_SET and Xi_Reset indicate respectively set and rest orders of activation about step xi. RTLi indicates the “ready to leave” signal which is emitted by the step Xi if it is active.

In this approach the Grafcet engine is considered as a reactive system which must always be able to provide an immediate response when it is solicited by the environment. The Grafcet evolution is then a series of reactions. Relatively to an environment scale time each reaction is considered as instantaneous.

The synchronous assumption defines a scale of discrete logical time, constituted of instants corresponding to each reaction of the system. The events having started the reaction are considered as simultaneous events. Moreover, the reactions are done at null time: the emitted signals during a reaction are then simultaneous to the signals which have caused the reaction (the production of the reaction at the same logical time). A reaction is then instantaneous, which avoids the concurrent partial reactions, source of indeterminism.

Let’s see now how we can integrate Esterel in a VRML model[8][9]. The “reactive script” concept is based on the use of SL language which is a variety of Esterel language. The implementation is done using “C reactive”, “Tcl/ Tk” or Java language. These scripts are very adapted to the development of the synchronous reactive applications under web. The VRML Script of the preceding implementation will be replaced then by a reactive script. As shown in Figure 4.
VRML must generate temporal events to the reactive script; in other means, a "TimeSensor" node send cyclic events to generate temporal points to the script part. ESTEREL treatment must be done between two events generated by a "TimeSensor" node. The principal supposition here is that the time is external and controlled by VRML.

The concurrent execution relative to the Grafcet is executed at the beginning of each temporal point (generated by VRML) until the end of the treatment. At the beginning of each cycle, the script receives from the scene a set of events which represents the entries of PLC. At the end of the cycle it generates other returned events which represent the outputs of PLC.

**VMRL AND THE DIDACTIC COMPONENT.**

**The Modeling of the Didactic Component.**

The need of a reference model for the representation of a didactic component is as much critical than the component is more complex and constituted of a variety of modules. The didactic component represents the operative part which regroups all the tools which are at the base of the physical process. The inputs of the component are composed of the outputs orders send by the sequence control part(Grafcet Engine) and the process parameters at the entry of the system. The outputs correspond to both : The informations going back to the control part and the executed actions that give an added value.

For modeling component and in order to have a valid virtual pattern, we have used the advanced concepts in the simulation and visualization fields[4]. The functional models of simulation and visualization tend to regroup the visual data and their treatments within objects that encapsulate them. Generally such type of model can be divided into three parties:

- The visual model (in our case the geometrical part of the component): represents the simulated phenomenon.
- The graphic user interface (GUI): used to control the user input values, the visual representation and the simulation execution.
- The control of the simulation (the dynamic part of the component): controls the behavior of the simulation in response to the need of user or other events (mainly the outputs events of the Grafcet Engine).

We can divide simulation models to two types. The first type, called the "off-line" simulation represents the simulation in which the system obtains the needed key values for simulation from a mathematical model implemented outside of the simulation model. So, the calculation of these simulation values is not directly implemented in the simulation module. Figure 5(b) shows an « off-line » simulation functional model. The “off-line” simulation is no more convenient for modeling our didactic components because the “off-line” simulation data are treated by a very sophisticated algorithms before be simulated. Once results are ready, they are sent to the simulation control module. This leads us to say that the real time aspect needed to react to the user events and the I/O events which are exchanged with the Grafcet Engine, is not possible. In other words, after modification of one of the simulation parameters, we cannot obtain a real time modification result.

The second type, called “on-line” simulation, represents the simulations which integrates in the simulation control module, the mathematical model of simulated phenomena. The user can change the simulation parameters in order to affect immediately the behavior of the simulation. This type of simulation can immediately show the modification of the input values.

The simulation resolution engine calculates without stopping the behavior of the visualization model using the actual input values and the local time within simulation. This simulation model is the more appropriate model to simulate our didactic components which are characterized by a behavior based on a real time animation and reaction to events. Figure 5 (a) shows an « on-line » simulation functional model.

**The VRML Implementation of the Didactic Component.**

To make the process of modeling and VRML implementation of the different kinds of component more systematic, more uniform and simpler, we propose a reference model for the implementation of each module that constitutes the virtual didactic component. Referring to the “on-line” simulation, the representation of the reference model can be implemented in VRML according to the graphic model shown in Figure 6.

a) **The implementation of the visible part of the component .**

Visually, the component is recognized by:

- The user graphic interface (GUI) witch is divided into tow parts: The input interface ( user inputs) and the output interface ( presents the visual data to the user). The GUI can contain some blocs known as “Sensors” ( having only input function), a visible geometry and some control scripts. The input and output events generated by the GUI are then routed to the control part.
- The “Geometry” part: used to visualize the geometric aspect of the didactic component. From the point of view representation and grouping of 3D objects, The VRML architecture is based on the same concepts of a classic
graphic scenes. The scene can be constructed using the basic nodes (boxes, spheres, cones or polygons) or user-defined nodes which are generally PROTO type nodes.

b) the implementation of the dynamic part of the component.

The time is of primary importance in the representation of the component. This can be justified by the dynamic aspect which represents the operational evolution of the component. Maybe it is not the moment to describe in detail the contribution of time in our system but it is more interesting to describe how this time must be used[1][3]. By making association between the variation of one of the module characteristics and time we provide a description of both: periods of time and the evolution of the component state inside these periods.

VRML is based on a discrete model of time, implemented through a special node called "time sensors" that control animation. To animate an object, this must receive periodic events in order to modify one of its characteristics (position, color, etc). VRML introduces primitives for animation and interaction based on a simple event model: each object in the virtual environment can receive input events and send output events, these events are communicated between the objects through defined "routes" before and completely independent from the structure of the graphic scene.

In the proposed model the "control" part control the animation of the 3D scene which reflects the behavior of the component. So, the behavior of the module which is defined by its reactions to the requested operational supplies (from PLC or IGU) according to its state, is described in a complementary view called "control". This behavior is implemented in a textual form using languages like "VRMLScript" or "JavaScript". In order to respect the implementation concepts of VRML, the use of a language is not absolutely needed and the behavior can be implemented using another technique which is the "KeyFraming" that consist to specify some position values of the component and the viewer engine calculates the mechanical evolution of the component between these positions during an interval of time.

To illustrate the two possibilities of animation, we present in the figure 7(b) the keyframing technique based on the use of interpolator type node which for an initial value, an end value and a given fraction of cycle, calculate the corresponding interpolation. The linear interpolation is the only possible interpolation through these nodes. Figure 7(a) shows the implementation of the same animation in a programmed way by using a script node.

THE STORAGE OF THE SPATIOTEMPORAL CHARACTERISTICS OF THE DIDACTIC COMPONENT.

Within section 3 we have presented the spatial characteristics of the didactic component (visual aspect) and its temporal characteristics (dynamic aspect), thing that we called spatiotemporal characteristics. In this section we present our proposition about the storage of these characteristics within a format that must accomplish the following goals:

- Recording the spatiotemporal characteristics of the component within an open format.
- Facilitate the reusability of the components and the translation from and to VRML format.
- Allow a modularity description of the component by a regrouping of 3D objects.
- Facilitate the check, and the modification of the objects encoded in VRML by using a relational language.

It is difficult to find a standard for representing spatiotemporal data, and even more for spatiotemporal data having several geometries. However we can speak about two approaches for modeling this type of data:

- PERCEPTORY: developed by the geometric research center at Laval University in Quebec.
- MADS: developed at “Lausanne university” in partner with the computing laboratory of the “Brussels Private University” and the database laboratory of the “Federal Polytechnic School in Lausanne”, an approach that have been used and accomplished in the MurMur project[6].

In our works, we have been inspired by MADS formalism, in order to extend the «entity-association» model used to conceive our spatiotemporal data base. Just like Percepyory, MADS formalism uses pictograms to represent the space and the temporal characteristics of an entity, an attribute or an association between two attributes. Thus, for the geometries, it is possible to have the cases shown in figure 8(a). The hierarchy emphasizes the inheritance which can propagate. So, a geometry can be redefined by refinement, as an example a "Simple Geo" can be redefined then in "Line" or "Point", but not in "Line Set". Concerning the temporal part, Figure 8(b) shows temporal pictograms. For the same entity, it is possible to mix various pictograms, as shown by figure 9: the left part is reserved to the temporal characteristics and the right one to spatial characteristics. The spatiotemporal specification can be also attributed to associations as its shown in table 1. Actually the modeling of the data is made manually as it is shown in Figure 9.

This modeling enabled us to have a better approach about the nature of the VRML data and their interaction. After it is being realized, it is necessary to implement it within a management data base system. To realize our first prototype we have used Oracle8i[17].

Once the base structured and filled, it will be necessary to execute queries based on these data. For that an editor of query will be also developed and obviously, it will be associated to a viewer allowing an interactive selections of didactic components and the execution of complex student queries.

To record a component within this base it is necessary to record three parts having the following significances:

1. The visual component part: the relative entities to this part are (Shape, Geometry, Appearance, Cylinder, Extrusion, ElevationGrid, IndexedFaceSet, IndexedLineSet, PointSet, Box, Cone, Sphere, Texture, TextureTransform, MovieTexture, ImageTexture, PixelTexture, Appearance, Material, Sensors, Position).
2. The dynamic component part: the relative entities to this part are (Behavior, ControleGUI, GeneralBehavior, RegularBehavior, TimeSensor ...).

3. The connection component: the relative entities to this part are (InterfaceEngineG7).

The main idea is to structure the virtual didactic component in order to be able to model component having a great complexity. As shown in the conceptual data model (figure 9), the relation that exists between modules of the didactic component is a relation of composition presented by the association “Constitute”. This relation is retained by our model in order to accomplish the modularity and hierarchy aspect.

The obtained result from the application of this approach, contains a single root module (presented by one record within the “DidacticComponent” entity), which is the totality of the didactic component. The module is broken up into a set of child modules that replaces it, each one of these modules encapsulates its own decomposition. So, the obtained model is strongly structured and modular.

CONCLUSION.

Figure. 10 and figure 6. show the prototype that we have implemented based on the three principal ideas developed in this article: a) the design and the implementation of a Grafcet engine under VRML; b) the modeling and the implementation of the virtual didactic component using VRML; c) the storage of the VRML codification into a spatiotemporal data base.

Based on the idea of the “virtual laboratories” [12] [13] which imply an informational rebuilding of the real laboratory and its state, through animations and 3D scenes; we try actually to extend our system to allow the teaching of the Grafcet and automatics in a connected mode; in other words by making the tele-experimentation on distant PLC and didactic components through a virtual environment which uses the modeling and the implementation mentioned a) and b).

ACKNOWLEDGEMENT.

A part of this research was accomplished and financially supported by the High Institute of Technology of Sousse. The authors would also thanks Dr. H. H. Saliah for his valuable technical comments.

REFERENCES.


Figures and Tables.

FIGURE. 1
The General Architecture of the System
DEF EngineG7 Script {
  field SFBool T1 FALSE
  field SFBool T2 FALSE
  // processing activation of steps
  X1=reset || T1 & & (X1 || T2)
  X2=IT2 & & (X2 || T1)
  X3=IT4 & & (X3 or T1)
  X4=reset || T3 & & (X4 || T4)
  X5=IT4 & & (X5 || T3)

  function reset (v){RESET=v}
  function I1(v){FI1=v}
  function I2(v){FI2=v}
  function Cycle()
  {
    // The latching of input events
    i1=FI1
    i2=FI2
    reset=Reset
    // processing transition clearability
    T1= i1 && X1
    T2=!i1 && X2
    T3=i2 && X4
    T4=!i2 && X3 && X5
    // Sequential process
    if reset
      O1=FALSE
      O2=FALSE
    else
      O1=X2
      O1=X5
  }
}

FIGURE. 2
Producer-Consumer G7

FIGURE. 3
Implementation of the Grafcet engine using VRMLScript

FIGURE. 4
Reactive Script.

FIGURE. 5
The Simulation functional Models.

TABLE. 1
MADS Topological Relations

<table>
<thead>
<tr>
<th>Spatial Type</th>
<th>Pictograms</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disjunction</td>
<td></td>
<td>The linked entities have some disjointed geometry</td>
</tr>
<tr>
<td>Adjacency</td>
<td></td>
<td>The linked entities have some adjacent geometries (without common surface)</td>
</tr>
<tr>
<td>Crossing</td>
<td></td>
<td>The linked entities have a common part of which the dimension is less to the dimension of the two entities</td>
</tr>
<tr>
<td>Overlapping</td>
<td></td>
<td>The linked entities have a common part of which the dimension is equal to the dimension of the two entities</td>
</tr>
<tr>
<td>Inclusion</td>
<td></td>
<td>The geometry of an entity is completely inclusive in the geometry of the other entity</td>
</tr>
<tr>
<td>Equality</td>
<td></td>
<td>The geometry of the two linked entities is equal</td>
</tr>
</tbody>
</table>
FIGURE 6
The Component Didactic Model

FIGURE 7
VRML Animations.

FIGURE 8
MADS Pictograms
FIGURE. 9
SpatioTemporal Data Base Model

FIGURE. 10
THE CONNECTION INTERFACE BETWEEN THE PLC AND DIDACTIC