Visualization of Folktales on a map by coupling dynamic DEVS simulation within Google Earth

Jean-François Santucci, Laurent Capocchi

Visualization of Folktales on a Map by Coupling Dynamic DEVS Simulation within Google Earth

Santucci Jean-François, Capocchi Laurent
SPE UMR CNRS 6134, University of Corsica, Campus Grimaldi, 20250 Corte, France
santucci@univ-corse.fr, capocchi@univ-corse.fr

Keywords: DEVS, Discrete Event Simulation, Google Earth API, Dynamic variable structures

Abstract: This paper deals with dynamic visualization of folktales on a map. The visualization is performed using a coupling of dynamic simulation with the google earth API. The simulation is based on the DEVS (Discrete EVent system Specification) formalism. We have defined a set of basic models to perform visualization of folktales using a software framework called DEVSimPy. DEVSimPy is a framework dedicated to DEVS modeling and simulation of complex systems. We also present in this paper a validation of the developed basic models based on folktales coming from the Corsican mythology.

1 INTRODUCTION

This paper deals with spatial visualization of folktales. This work is part of a global project concerning the study of myths according to the method defined by Claude Levi-Strauss (Levi-Strauss, 1955; Levi-Strauss,1969; Levi-Strauss,1971; Levi-Strauss,1978; Levi-Strauss,1981) when dealing with Structural Anthropology. The goal is to offer anthropologists the possibility to visualize on a map a dynamic progress of the story involved by a given myth. The visualization is based on a Discrete EVent system Specification simulation (DEVS) dynamically linked with the Google Earth API (Brown, 2006). The interest of such a work for anthropologists is to be able to visualize a story stemming from a myth obtained after a DEVS structural anthropology simulation. From a computer science point of view the interest is to define a generic way to propose dynamic visualization of a given story. We choose to perform the dynamic features of the problem of visualization using the DEVS formalism. The spatial visualization on a map has been done using the Google Earth API. We describe in detail in this paper how a dynamic coupling between DEVS formalism and Google earth has been realized. The proposed approach has been implemented using the DEVSimPy framework in order to develop a dynamic library of DEVS models. We have validated this library using a story stemming from a myth belonging to the Corsican oral culture. This myth is a “”reference” myth according to the meaning of “reference” as Claude Levi Strauss has defined it in his Mythologiques Book Series (Levi-Strauss,1969; Levi-Strauss,1971; Levi-Strauss,1978; Levi-Strauss,1981). The outline of the paper is as follows: we first give the context of the research: which is the structural analysis of myths using DEVS formalism. We also introduce in this first section how the DEVS formalism has been used in order to perform myth transformation. The DEVSimPy framework is briefly presented. Section 3 deals with dynamic visualization of folktales. First an overview of the proposed approach is described. Then we explain how we implemented the simulation of variable dynamic structures which are used in order to perform dynamic visualization of folktales on a Google Earth map. Then we detail the description of the dynamic relationship between the DEVSimPy library of models and the Google Earth API. The validation of the proposed library is presented in section 4 before the concluding remarks and future work.

2 BACKGROUND

We present in this section the background of the work that is the main contribution of this paper. The dynamic visualization of folktales belongs to a method called Structural Anthropology (Levi-Strauss, 1955). We have developed a software approach based on DEVS formalism in order to perform Structural Anthropology. This software approach leans on the use of the DEVSimPy framework. We first briefly introduce in this section how to perform structural anthropology using a computer science approach. The DEVS formalism is summarized in sub-section 2.2. The DEVSimPy framework is detailed in sub-section 2.3 before the presentation in sub-section 2.4 of the DEVSimPy implementation of folktale transformations as defined by Claude Levi Strauss when dealing with Structural Anthropology.

2.1 Structural analysis of myths

The structural analysis of myths has been defined by Claude Levi Strauss in his books. He specially explained how to apply his method in his famous book series (Levi-Strauss,1969; Levi-Strauss,1971; Levi-Strauss,1978; Levi-Strauss,1981). The method leans on the concept of myth transformation.
Myth transformation consists in performing the generation of myths from a first "reference" myth as Claude Levi Strauss calls it. The generation involves a set of basic transformations that are applied on the mythems of a given myth. A mythem has been defined in 1955 (Levi-Strauss,1955) as a sentence of a given myth composed by a term and a function. For example the following sentence:" the ogre lives near Casta" is described by the term "ogre" and the function "lives-near-Casta". All the sentences of a given myth can be expressed by a set of mythems. Then a set of transformations can be applied in order to generate a new myth. Seven basic transformations have been defined by Claude Levi-Strauss: homology, symmetry, opposition, homology, canonical formula, suppression of mythems, and addition of mythems. A set of work associating computer sciences and structural analysis of myth has been proposed in the past towards the objective of myth transformation modeling (Klein, Aeschlimann , Applebaum, Balsiger, Curtis, Foster, Kalish, Kamin, Lee, Price & Sasieder, 1997; Maranda,1967; Maranda 1968; Richard & Jaulin, 1971). These works are attempts to formalize the analysis of tales developed by Claude Levi Strauss. These approaches are based on computer sciences in order to perform systematic myth transformations software. However all these attempts did not succeed in this task. In order to find a solution to this problem we have defined a software myth transformation approach based on the DEVS formalism which is summarized in sub-section 2.2. The structural anthropology of myth as defined by Claude Levi Strauss also proposes the analysis of a given myth according to his spatial representation in the landscape. Since it is issued from oral cultures, a myth is told and retold in front of people. Usually a myth involves a set of places well known by the people who are listening to the story. When the story is told, the landscape linked with the myth is mentally viewed by the listeners. In this paper we describe in detail how after having performed a set of myth transformation we are able to propose the dynamic visualization of a myth on a map using a computer.

2.2 DEVS formalism

We briefly introduce the DEVS formalism (Zeigler, 1976; Zeigler, 1984; Zeigler, Praehoffer & Kim, 2000). In the DEVS formalism, one must specify: 1) basic models from which larger ones are built, and 2) how these models are connected together in hierarchical fashion.
An atomic model allows specifying the behavior of a basic element of a given system.
Basic models (called Atomic Models) are defined by the following structure:
AM = $< X, S, Y, C, \delta_{ext}, \boldsymbol{\delta_{int}}, \lambda, ta >$
Where,

- *X* is the set of input values,
- *S*: is the set of sequential states,
- *Y*: is the set of output values,
- $\boldsymbol{\delta_{int}}$, is the internal transition function dictating state transitions due to internal events,
- δ_{ext} the external transition function dictating state transitions due to external input events.
- λ is the output function generating external events at the output, and
- *ta* is the time-advance function which allows to associate a life time to a given state.

An atomic model allows to specify the behavior of a basic element of a given system. Connections between different atomic models can be performed by a Coupled Model (*CM*) (Zeigler,1976 ; Zeigler, 1984):
CC = $< X, Y, D, \{M_d / d \in D\}$, IC,EIC, EOC>
Where,

- X is the set of input values,
- Y is the set of output values,
- D is the set of model references, that is to say a set of names associated to the model's components $\{M_d \ / \ d \in D\}$ is the set of coupled model's components, with d being in D. These components are either atomic or coupled DEVS model,
- IC, EIC and EOC define the coupling structure in the coupled system (IC defines the internal coupling, transforming a component's output into another component's input within the coupled model ; EIC is the set of external input coupling, which connects the inputs of a coupled model to components inputs; EOC is the set of external output coupling

A coupled model, tells how to couple (connect) several component models together to form a new model. This latter model can itself be employed as a component in a larger coupled model, thus giving rise to hierarchical construction.

A simulator is associated with the DEVS formalism in order to exercise coupled model's instructions to actually generate its behavior. The architecture of a DEVS simulation system is derived from the abstract simulator concepts (Zeigler, 1990) associated with the hierarchical and modular DEVS formalism.

In order to implement DEVS modeling and simulation we use the DEVSimPy framework presented in sub-section 2.3.

2.3 DEVSimPy Framework

DEVSimPy (stand for DEVS simulator in Python language) is a collaborative Modeling and Simualtion (M&S) software. It is used in order to model and simulate complex systems based on the DEVS formalism in Python programming language.

The initial idea of DEVsimPy is to develop a Graphical User Interface based on the wxPython library (Sanner, 1999) around the PyDEVS kernel (Bolduc & Vangheluwe, 2001).

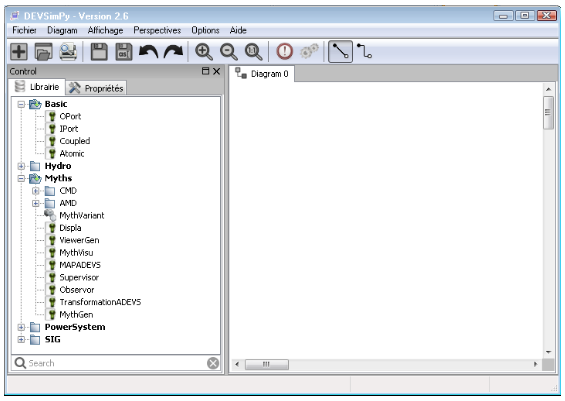


Figure 1: The DEVSimPy framework interface

PyDEVS is an API of the modeling and simulation algorithms of DEVS implemented in python language. It is used in the software ATOM3 (De Lara & Vangheluwe, 2011). WxPython is a blending of the wxWidgets C++ class library with the Python programming language.

The DEVSIMPY framework provides a friendly interface for discrete event modelling and simulation by integrating the basic classes of the PyDEVS kernel. Figure 1 gives an example of the window interface. You may notice in figure 1 that the window is split into two parts:

- The left part allows the user to visualize the classes which can be instantiated (using a drag and drop). Two kinds of classes appear on figure 1: (i) Basic classes which correspond to the basic components of Discrete EVent Specification formalism and already defined in the classical PyDEVS kernel (Bolduc & Vangheluwe, 2001); (ii) classes which have been defined in the context of the proposed software for performing myth transformations and myth analysis which may be found under the Myths and Atomic Models (AM) folder.
- The right part (called a canevas) which is dedicated to the design of coupled models (also called diagrams in the following) with a simple drag and drop of classes belonging to the left part of the window. The created coupled model can be saved under a format allowing it to be easily reused and shared.

During the design of a DEVS coupling between several atomic or coupled DEVS models, errors can occur due to the difficulty to follow multiple links. Using DEVSimPy, this task becomes fast and easy. Indeed, it proposes the coupling between models by using the drag-and-drop. Moreover, the port numbers are displayed to facilitate the identification of ports.

DEVSimPy has several other advantages and most of them are inherited from DEVS formalism. First, atomic and coupled DEVS models are considered as black-boxes which have both graphical and the behavioural features that can be configurable by a user-friendly dialog. The modular and the hierarchic aspect of the DEVS formalism is also proposed in DEVSimPy. Since coupled models can be composed by atomic or/and coupled models, DEVSimPy proposes an easy way to visualize the components belonging to a coupled model and their interconnections by double clicking. Secondly, as it is defined by the DEVS formalism, there is no need to implement the simulator in DEVSimPy after the modeling step has been performed. After the modeling task, the simulator is automatically built by clicking on a simple button in the toolbar of DEVSimPy. Thus, a dialog appears allowing the user to choose some of the following options: the verbose mode, the simulation algorithm, the profiling mode and the final simulation time.

Another benefit of the use of DEVSimPy is the possibility to define a dynamic library of atomic and coupled models. Each model is stored in a tree view and is directly accessible by a Developer. When he would like to instantiate a model, he has just to use a drag-and-drop from the panel library to the canvas. When the simulation is performed, it is possible to suspend the process and change the behavior of model under simulation. The loading of a new library is performed dynamically and the modification is immediately operational when the simulation restart. Even better, it is possible to modify the code of model during the suspension of the simulation and to reload it before to restart the simulation process. This property is often used during the model-debugging phase.

When a Developer needs to implement a new functionality, which is required by a specific domain modeling, he can use a plug-in approach. DEVSimPy is based on an efficient system of plug-in allowing the extension of functionality in a modular way. Overall, when the developer wants to visualize output data in a specific way or to debug models, he can create a plug-in based on the viewer model proposed in DEVSimPy. Plug-ins are independent of the DEVSimPy kernel and there are managed through a specific dialog dedicated to their import, activation, deactivation or configuration. This plug-in approach is very interesting from the point of view o the developer view. The Developer can load or unload plug-in depending on the nature of the studied domain without having to define new atomic models that can pollute the library.

The myth modeling requires several atomic models due to the important number of transformation processes. Therefore, the number of coupling between these models is important and the use of DEVSimPy makes sense. Moreover, the analysis of the simulation results can be done with a specific plug-in in order to facilitate both the historic and the final interpretation of the transformation of the myths. The implementation of a data collector (or viewer) using a plug-in is easily done with DEVSimPy.

2.4 DEVS simulation for myth transformations

We present the software framework we used in order to implement the DEVS models required to perform myths transformation.
The DEVS modeling process we develop is based on the three following steps:

- DEVS modeling of a mythem
- Data structure for representing a set of mythems (that is to say a myth)
- DEVS modeling of the transformation of a myth based on the application of the seven basic transformations

In order to represent a tale belonging to a set of variants of a myth we model the structure of a tale using a coupled model of the DEVS formalism. The coupled model is composed by a set of atomic models corresponding to the different parts of the tale. At the latest level we are using atomic models of the DEVS formalism in order to model the narrative structures of the tale. Each atomic model refers to a basic element of Claude Levi Strauss theory: the mythem involving two variables: a and x. A tale is therefore split into mythems and stores in a text file. The beginning of the story is described through the following four sentences called mythems described in figure 2.

1. *The ogre named Orcu knows the secret of fabrication of the Corsican cheese using some milk.*
2. *The shepherds are jealous of the Orcu because of this secret.*
3. *The ogre is captured because of an ingenious trap.*
4. *The shepherds ask for the secret of fabrication of the cheese.*

Figure 2: The four first mythems of a Corsican folktale.

We enumerate below for each mythem the value of the variables “a” and “x”:

- 1st mythem : a = orcu; x = secret

- 2nd mythem: a = shepherd; x = jealous
- 3rd mythem: a = orcu; x = trapped
- 4th mythem: a = shepherd; x = secret

We consider a set of folktales issued from Corsican mythology. Corsica is an island of the Mediterranean area where more than 6000 years ago megaliths emerged all over the island territory. These megaliths are signs that define a sacred space linked with a very old mythology. A set of folktales that are linked with the landscape of the Corsican Island has been generated using our approach. Figure 3 presents the DEVS modeling scheme allowing to generate a set of myths belonging to the Corsican Mythology. In order to obtain the visualization of the set of transformations concerning the myths already generated starting from the reference myth we used a graph representation. Graph visualization is a way of representing the relations between myths as diagrams where nodes are labelled with a given myth and oriented arcs between two nodes represents the fact that the successor node of the arc corresponds to a myth which has been generated from the myth corresponding to the predecessor node of the same arc. The graph representing the set of transformations obtain from the simulation of the DEVS model described in figure 3 is given in figure 4. The generated myths have been validated by the anthropologist specialized in Corsican Culture.

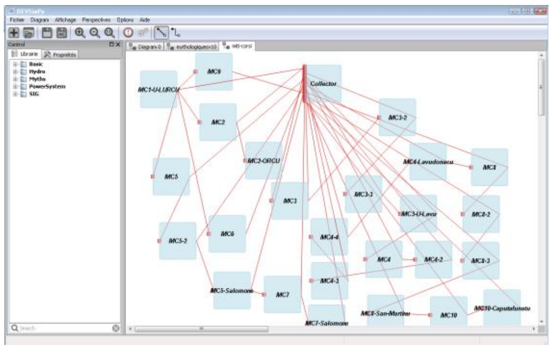


Figure 3: overview of the coupled model allowing the generation of s set of myths from an initial myth.

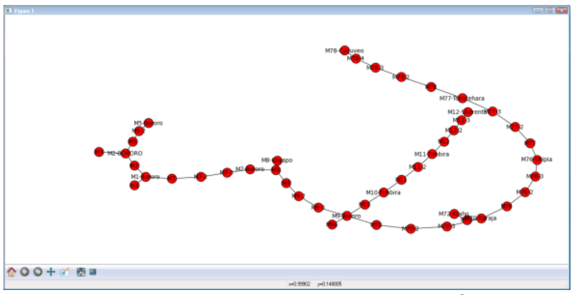


Figure 4: Resulting graph describing the transformations of myths after simulation of the DEVS model of figure 3

The next step is the visualization on a map of a given myth belonging to the set of myths already generated. This step is the main contribution of this paper and explained in section 3 while its validation is presented in section 4.

3 DYNAMIC VISUALIZATION OF FOLKTALES

The goal is to offer the possibility to dynamically visualize on a map the unfolding of a given myth. The myth under study is obtained using the myth transformation software presented in section 2. In this section after having pointed out the interest of such a study for the anthropologists we detail the proposed approach. We particularly explain how we have been able to deal with: (i) DEVS variable dynamic structures, (ii) an association of the Google Earth API and (iii) the DEVSimpY framework and dynamic visualization using Google Earth API.

3.1 Interest and proposed approach

The study of folktales is an important task belonging to the anthropologist domain. As we presented in section 2.1 one of the main contributor to this kind of study is Claude Levi Strauss. We summarized in section 2 how we have been able to help anthropologist to perform structural analysis using a computer. Once a set of myths belonging to the same geographical region has been generated using the structural analysis method, it is common for an anthropologist to study each myth individually. This study leans on a two kinds of analysis: analysis of a given myth according to the notion of codes as defined by Claude Levi Strauss in his Mythologiques book series(Levi-Strauss,1969; Levi-Strauss,1971; Levi-Strauss,1978; Levi-Strauss,1981); analysis of a given myth according to its contextualization in the landscape. In this paper we focus on the second type of analysis. The goal is:

- to visualize the unfolding of a folktale own to the story skeleton
- to sequence the story on a map which summarizes the flow of events.

We decide to combine the simulation of models involved in the DEVSimPy framework (see section 2.3) with some features of the Google Earth API in order to perform the dynamic visualization of a given folktale on a map. The following problems have been solved in order to implement this proposed approach:

- simulation of variable dynamic structures using DEVSimPy

- activation of google earth from the DEVSimPY framework

- dynamic refreshment on the map while the story skeleton is being simulated.

3.2 DEVS modelling overview

In order to perform the dynamic visualization of a folktale we have defined a DEVS modeling scheme by implementing the following set of specific atomic models: (1) Viewergen; (2) SIGviewer; (3) MythVisu; (4) PointMyth

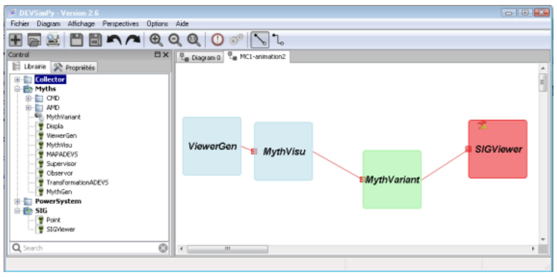
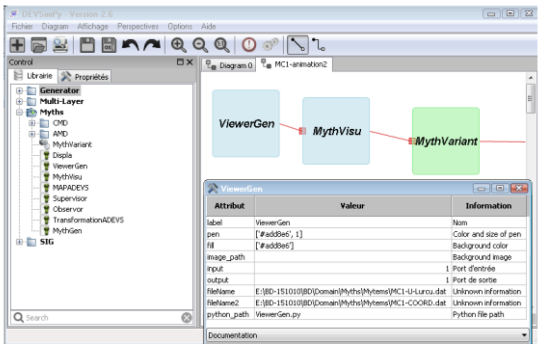


Figure 5: DEVS modelling for dynamic visualization

We also defined a coupled model called MythVariant that is used as the initial variable structure, which will evolve when a new mythem has to be visualized on a map. Figure 5 presents the DEVS modeling scheme involving four DEVS models: the three atomic models (Viewergen, MythVisu and SiGviewer) as well as the coupled model MythVariant. Furthermore instances of the PointMyth atomic model will be dynamically added into the MythVariant coupled model when a new mythem is encountered by reading a myth.

The inter-connection between the four DEVS models is given in figure 5. The goal of the atomic model called ViewerGen consists in: (i) scanning a text file corresponding to the given myth under study and generated as presented in sub-section 2.4; (ii) sending a message on its output for every mythem belonging to the given myth, where the message contains the term and the function involved in the mythem as well as the location of each of the mythems belonging to the myth. This information is located in the two files associated to the ViewerGen atomic model as seen on figure 6 (the files can be loaded using the two attributes: filename (term and function of the mythems) and filename2 (location of each mythem). The Viewergen atomic model is connected to the MythVisu atomic model which is in charge of the management of variable dynamic structures: for each message received on its input port, the MythVisu atomic model is able to add an instance of the PointMyth atomic model in the MythVariant coupled model. A message containing the location of the point associated with the current mythem (as well as the term and the function involved by the mythem) is then sent to the SIGviewer atomic model. The implementation of the MythVisu atomic model is detailed in sub-section 3.3.



Attribut	Valeur	Information
label	ViewerGen	Nom
pen	['#add8e6', 1]	Color and size of pen
fill	['#add8e6']	Background color
image_path		Background image
input	1	Port d'entrée
output	1	Port de sortie
fileName	E:\BD-151010\BD\Domain\Myths\Mytems\MC1-U-Lurcu.dat	Unknown information
fileName2	E:\BD-151010\BD\Domain\Myths\Mytems\MC1-COORD.dat	Unknown information
python_path	ViewerGen.py	Python file path

Figure 6: Properties of the MythVisu atomic model.

The goal of the SIGviewer atomic model is to: (i) open the google earth window; (ii) print the point corresponding to the current mythem on the map. The implementation of the SIGviewer atomic model is detailed in sub-section 3.4

3.3 Variable dynamic structures

In order to perform dynamic visualization of a given myth we had to implement a way to perform DEVS simulation of variable dynamic structures. In order to deal with the modeling of dynamic variable structure system a set of scholars have proposed in the recent past to extend the DEVS formalism (Barros,1997; Hu, Zeigler,Mittal,2005; Baati, Frydman, Giambiasi, 2007a; Baati, Frydman, Giambiasi, 2007b ; Hui & Wainer, 2006 ; Barros, 2003). The proposed implementation leans on the definition of a special coupled model involving a classical atomic model usually called Supervisor component (Baati,Frydman,Giambiasi, 2007a ; Baati,Frydman,Giambiasi, 2007b) which drives the dynamic structure discrete event system modifications and coupled models representing the set of potential structures.

Our approach supports changes in structure by the introduction of a special atomic model that keeps in its internal state the structure of a network of models. Changes in the state are automatically mapped into changes in structure. We have called MythVisu the special atomic model in charge of the management of dynamic variable structures in the case of this application.

Figure 7 illustrates the empty DEVS coupled model called MythVariant while the MythVisu atomic model coding is given below:

```
1class MythVisu(MythDomainBehavior) :
2  def __init__(self,fileName=""):
3     MythDomainBehavior.__init__(self)
4     self.msg = None
5     self.LAT = []
6     self.LONG = []
7     self.state = {'status':'ACTIVE',
8    'sigma' : INFINITY}
9
10 def intTransition(self):
11    self.state['sigma'] = INFINITY
12
13 def extTransition(self):
14    mv= self.OPorts[0].outLine[0].host
15    msg = self.peek(self.IPorts[0])
16    if msg != None:
17          self.msg = msg
18    mv.componentSet = []
19    m = PointMyth.PointMyth(latitude
20        = …)
21    m.timeNext=m.timeLast=
22    m.myTimeAdvance = 0.
23    m.addInPort()
24    m.addOutPort()
25    mv.addSubModel(m)
26    mv.connectPorts(mv.IPorts[0],m.IPo
27    rts[0])
28    mv.connectPorts(m.OPorts[0],
29       mv.OPorts[0])
30    self.state['status']='ACTIF'
31    self.state['sigma'] = 0
32
33 def outputFnc(self):
34    self.msg.value = 0
35    self.msg.time= self.timeNext
36    self.poke(self.OPorts[0],self.msg)
37
38 def timeAdvance(self): return
39   self.state['sigma']
```

The dynamic modification of the initial empty coupled model MythVariant (pointed out on figure 4) is realized by the δ_{ext} transition function of the atomic model called MythVisu as it may be seen in the code presented above (see statement line 25 `mv.addSubModel(m))`. The variable m is an instance of the PointMyth atomic class model and is computed by the statement line 19 and 20:
`m = PointMyth.PointMyth(latitude =…)`

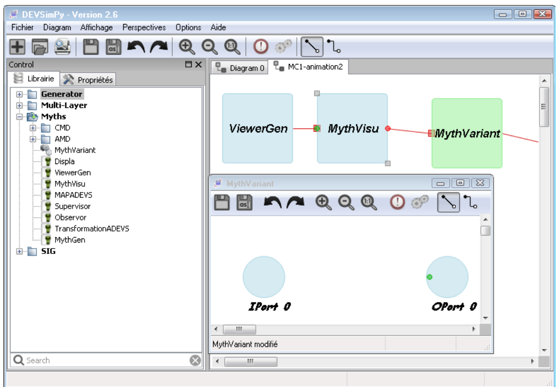


Figure 7: Illustration of the MythVariant coupled model

The class MythVisu described above points out five methods corresponding to the four basic functions of an atomic model and the constructor method of the class (called *__init__)* which consists in initializing the all attributes of the component (cf. the code above from line 2 to line 8). The DEVS δ_{int}, function of the atomic model MythVisu is coded through the intTransition method (described in lines 10 and 11) and consists in assigning the current value of the state variable sigma to infinity in order to indicate that the atomic model is waiting for an external event. The DEVS δ_{ext}, function of the atomic model MythVisu is coded through the extTransition method (cf. from line 13 to 31) which mainly consists in generating new atomic models in the associated coupled model called MythVariant. The DEVS λ function is coded through the outputFnc method of the atomic model MythVisu -from line 33 to 36) and consists in sending a message on the output port of the atomic model own to the statement of line 36:
`self.poke(self.OPorts[0],self.msg)`
Finally the DEVS ta function is coded through the timeAdvance method of the atomic model MythVisu (lines 38 and 39) and consists in returning the current value of the state variable sigma.
We have detailed in this sub-section how to manage dynamic variable structures using the DEVSimPy framework. This concept of dynamic variable structures is used in order to dynamically modify the

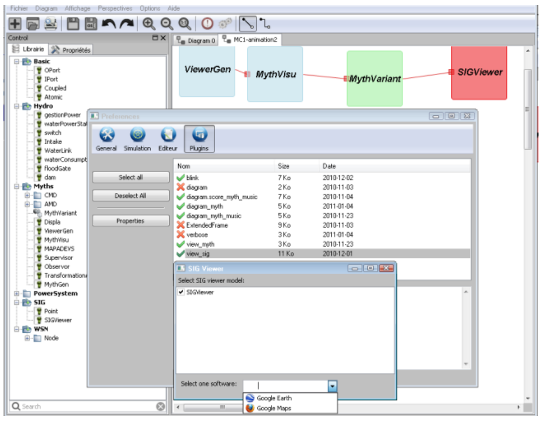
coupled model MythVariant every time an event corresponding to the detection of a new mythem occurs. In sub-section 3.4 and 3.5 we describe in detail how we have been able to link the dynamic evolution of the coupled model MythVariant with a dynamic visualization of mythems on a map. This visualization leans on the use of the Google earth API.

3.4 Google Earth invocation

The Google earth invocation leans on the two following atomic models: PointMyth and SIGViewer. The link between the DEVSimPy software and google earth is performed through a KML (Keyhole Markup Language) file (Google Inc, 2007). KML is a specialized type of XML that enables to build and organize points, lines and other information on a google earth map. The δ_{ext} function of the SIGviewer atomic model allows the management of the placement of marks on google earth using a KML file as it may be seen below:

```
1def extTransition(self):
2  activePort = self.myInput.keys()[0]
3  msg = self.peek(activePort)
4  point = msg.value[0]
5 kml_tree = KML.KML_Tree(self.fn)
6  ### if there is no Folder, we create
7  ###it with a placemark
8    kml_tree.add_placemark(…..)
9   ### some Folder are present
10 else:
11  folder_list = ……….
12  if folder_list != []:
13    folder = folder_list[0]
14    place_list =………
15    if place_list != []:
16     place = place_list[0]
17     kml_tree.add_placemark(folder,
18     point, place)
19    ### the new placemark is added on
20    ###the existing Folder
21    else:
22     kml_tree.add_placemark(….)
23 else:
24  kml_tree.add_placemark(….)
25  kml_tree.write(self.fn)
26 self.state['sigma'] = 0
```

Furthermore the dynamic visualization will be performed using the plug-in *view_myth* of DEVSimPy. This plug-in allows the user to associate the initialization of the software which will be concerned with the dynamic visualization. The user has to choose between Google Earth and Google Map as it may be seen in figure 8.



Nom	Size	Date
✓ blink	7 Ko	2010-12-02
✗ diagram	2 Ko	2010-11-03
✓ diagram.score_myth_music	7 Ko	2010-11-04
✓ diagram_myth	5 Ko	2011-01-04
✓ diagram_myth_music	5 Ko	2010-11-23
✗ ExtendedFrame	9 Ko	2010-11-03
✗ verbose	3 Ko	2011-01-04
✓ view_myth	3 Ko	2010-11-23
✓ view_sig	11 Ko	2010-12-01

Figure 8: ViewMyth plug-in initialisation

3.5 Dynamic visualization

The dynamic visualization of the placement of points on the map is realized by setting a refreshment of the KML file associated with the SIGviewer atomic model like described above. The file is modified every time an event is sent to the SIGviewer atomic model own to the δ_{ext} function presented in sub-section 3.4. The detection of an event is performed through the statement pointed out line 3 of the previously detailed code of δ_{ext}: `msg = self.peek(activePort)`. Then the KML file is modified according to the code presented above in sub-section 3.4 (form line 6 to line 25). Furthermore the KML file can be checked by the Google Earth API periodically in order to have dynamic refreshment by enabling the auto-update function of the Google Earth API. Using this option the KML file source is regularly reloaded at an interval that we have specify. The modification of the KML file is performed during the DEVS simulation of the overall DEVS modelling presented in sub-section 3.2. However one of the main interests of the proposed approach is to offer the possibility of a dynamic printing on a map the unfolding of a folktale. We therefore have to perform a step-to-step simulation in order to activate the refreshment of the KML file associated with the SIGViewer using the previously introduced method. We use a special plug-in called Blink belonging to the DEVSimPy features allowing to perform a step-to-step simulation as described in figure 9. Using the Forward button of the Blink Logger window we are able to control the generation of the KML file. By setting an appropriate period of refreshment (for example 10 second) the user is able to see the placement of every mythem on the screen and read the required information concerning the folktale as it is described in the next section.

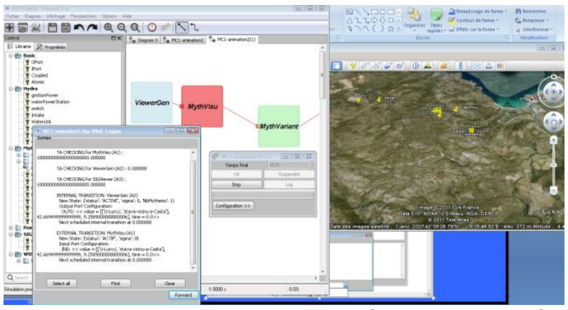
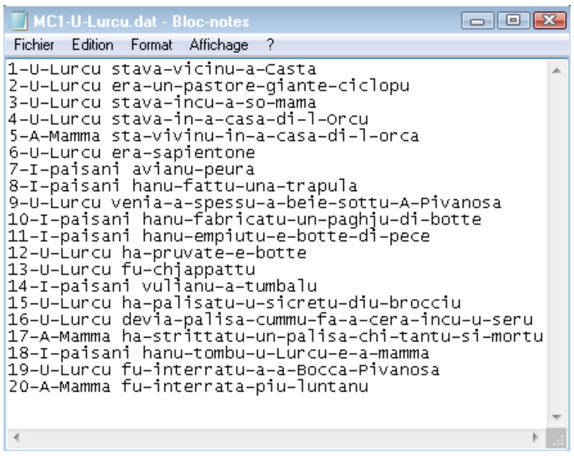


Figure 9: Step-to-step simulation of the generation of points on the map.

4 VALIDATION

In order to validate the previously DEVS models we consider a set of myths belonging to the Corsican mythology. We detail the validation of the first myth of this set. This myth is untitled U Lurcu which means the ogre in Corsican language. The twenty mythems belonging to this folktale are given in figure 10. They are expressed in Corsican language. As presented in sub-section 2.4, each mythem is composed by a term and a function. Furthermore each mythem is numerated.


MC1-U-Lurcu.dat - Bloc-notes
Fichier Edition Format Affichage ?

```
1-U-Lurcu stava-vicinu-a-Casta
2-U-Lurcu era-un-pastore-giante-ciclopu
3-U-Lurcu stava-incu-a-so-mama
4-U-Lurcu stava-in-a-casa-di-l-Orcu
5-A-Mamma sta-vivinu-in-a-casa-di-l-orca
6-U-Lurcu era-sapientone
7-I-paisani avianu-peura
8-I-paisani hanu-fattu-una-trapula
9-U-Lurcu venia-a-spessu-a-beie-sottu-A-Pivanosa
10-I-paisani hanu-fabricatu-un-paghju-di-botte
11-I-paisani hanu-empiutu-e-botte-di-pece
12-U-Lurcu ha-pruvate-e-botte
13-U-Lurcu fu-chjappattu
14-I-paisani vulianu-a-tumbalu
15-U-Lurcu ha-palisatu-u-sicretu-diu-brocciu
16-U-Lurcu devia-palisa-cummu-fa-a-cera-incu-u-seru
17-A-Mamma ha-strittatu-un-palisa-chi-tantu-si-mortu
18-I-paisani hanu-tombu-u-Lurcu-e-a-mamma
19-U-Lurcu fu-interratu-a-a-Bocca-Pivanosa
20-A-Mamma fu-interrata-piu-luntanu
```

Figure 10: the twenty mythems in Corsican language of the folktale under test

We present in figure 11 the result of the simulation of the myth presented in figure 10. The screenshot describes the sequence of points that have been automatically generated during the simulation process. All the mythems are represented on the map and identified through a number pointing out the order of generation corresponding to the order of the mythems in the given folktale. Furthermore by clicking on the information button of one of the generated point the user may read the content of the mythem as it may be seen on figure 11.

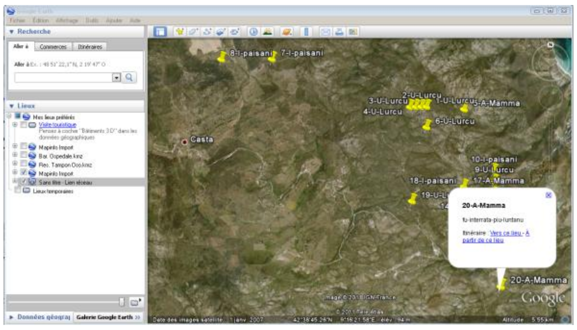


Figure 11: Screenshot of the result of generation on the map of the points involved in the folktale under test

There are therefore two ways of using the simulation: (i) by selecting the step-to-step simulation -the user has in this case chosen the Blink plug-in as seen in sub-section 3.5 and is able to discover the unfolding of the myth interactively by clicking on the forward button; (ii) by performing a complete simulation and own to the enumeration of the generated points on the map the user is able to read the unfolding of the myth on the map by clicking on the different points of the story one after the other. The main interest for an anthropologist is to be able to visualize how the story takes place in its natural environment. Since the places of the myth very often correspond to sacred places of the considered territory it is very important for an anthropologist to see how they are linked together through the unfolding of a myth.

5 CONCLUSIONS

We have presented in this paper set an approach to dynamically visualize the unfolding of a folktale. This work is part of a multi-disciplinary research concerning Structural anthropology of myths and computer science: the goal is to offer a software approach to help an anthropologist to perform Claude Levi Strauss myth analysis method. This method is based on the concept of mythem. Each folktale (or myth) is decomposed into a set of mythems which are the basic elements of a given story. The proposed approach leans on an association of the DEVS formalism and the Google Earth API. We have detailed the set of DEVS models which has been defined in order to: (i) dynamically read a text file containing the story where each line of the file corresponds to a mythem ; (ii) dynamically print on a map each mythem using the Google Earth API. We introduced a software framework called DEVSimPy which has been used in order to implement the set of DEVS models required in order to perform the dynamic visualization of myths. We also described the validation of these DEVS model using a concrete folktale belonging to the Corsican mythology. The future work will consist in deeply collaborating with an anthropologist in order to model the 813 myths defined by Claude Levi Strauss in his *Mythologiques Series* and a set of more than 200 myths coming from the Corsican Mythology and known by the anthropologist. Our future work will also concern the dynamic simulation of a given myth in a 3D environment.

REFERENCES

Baâti, L., Frydman, C. & Giambiasi, N. (2007a), LSIS-DME M&S Environment Extended by Dynamic Hierarchical Structure DEVS Modeling Approach. *ACM/SMS proceedings of the Spring Simulation Conference*, Norfolk, Virginia, USA.

Baâti, L., Frydman, C. & Giambiasi, N. (2007b). Algorithm for DEVS structure changes. *Proceedings of the 6th EUROSIM Congress on Modeling and Simulation*, Ljubljana, Slovenia.

Barros, F.J. (1997). Modeling Formalism for Dynamic Structure Systems. *ACM Transactions on Modeling and Computer Simulation*, 7(4), 501-514.

Barros, F.J. (2003). Dynamic structure multiparadigm modeling and simulation, *ACM Transactions on Modeling and Computer Simulation, 13(3)*, 259-275.

Bolduc, J.S. & Vangheluwe, H, (2001). *The modeling and simulation package PythonDEVS for classical hierarchical DEVS*. Technical report MDSL-TR-2001-01, McGill University, Montréal, Canada

Brown, M.C. (2006), *Hacking Google Maps and Google Earth*, Canada: Wiley Publishing, Inc

De Lara, J. and Vangheluwe, H.(2001), "AToM3: A Tool for Multi-formalism and Meta-modelling.", In *Proceedings of the 5th International Conference on Fundamental Approaches to Software Engineering* (FASE '02), Ralf-Detlef Kutsche and Herbert Weber (Eds.). Springer-Verlag, London, UK, UK, pp. 174—188.

Google Inc. , KML 2.1 Reference, 2007, http://earth.google.com/kmlkml_tags_21.html.

Hu, X., Zeigler B. P. & S. Mittal, (2005). Variable Structure in DEVS Component-Based Modeling and Simulation, *Simulation*, 81(2), 91-102

Hui, S. & Wainer, G. (2006). A Simulation Algorithm for Dynamic Structure DEVS Modeling. *Proceedings of the Winter Simulation* Conference, WSC 06, 815 – 822.

Jason, H. & Segal, D. (1977). *Patterns in Oral Literature*. The Hague: Mouton Publishers

Klein, S., Aeschlimann, J.F., Applebaum, M.A., Balsiger, D.F., Curtis, E.J., Foster, M., Kalish, S.D., Kamin, S.J., Lee, Y.D, Price, L.A. & Sasieder, D.F. (1977). Modeling Propp and Levi-Strauss in a Metasymbolic Simulation System. In Jason & Segal (Ed.), *Patterns in Oral Literature* (pp. 141- 222), The Hague: Mouton Publishers.

Lévi-Strauss, C. (1955). The structural study of myths. *Journal of American Folklore, 68(270)*, 428-444.

Levi-Strauss, C. (1969). *Introduction to a Science of Mythology 1. The Raw and the Cooked,* New York: Harper & Row

Levi-Strauss, C., (1973). *Introduction to a Science of Mythology 2. From Honey to Ashes,* New York: Harper & Row.

Levi-Strauss, C. (1978). *Introduction to a Science of Mythology 3. The Origin of Table Manners,* New York: Harper & Row.

Levi-Strauss, C. (1981). *Introduction to a Science of Mythology 4. The Naked Man.* New York: Harper & Row

Maranda, P.(1967). Computers in the Bush: Tools for the Automatic Analysis of Myths. In J. Helm (Ed.), *Essays on the Verbal and Visual Arts: Proceedings of the 1966 Annual Spring Meeting of the American Ethnological Society* (pp. 77-83), Seattle: University of Washington Press.

Maranda, P. (1968). Analyse Quantitative et Qualitative de Mythes sur Ordinateur. In J. C. Gardin & B. Jaulin, (Ed.), *Calcul et Formalisation dans les Sciences de l'Homme* (pp. 79-86), Paris: Editions du Centre National de la Recherche Scientifique.

Richard, P. & Jaulin, R. (1971). *Anthropologie et calcul,* Paris: Union Générale d'édition.

M. F Sanner, "Python: a programming language for software integration and development," *J. Mol. Graphics Mod* 17, pp. 57–61 , 1999.

Zeigler, B.P. (1976). *Theory of Modeling and Simulation*. New York: Wiley.

Zeigler, B.P. (1984). *Multifaceted Modeling and Discrete Event Simulation*. London: Academic Press.

Zeigler, B.P. (1990). *Object-Oriented Simulation with Hierarchical, Modular Models,* London: Academic Press.

Zeigler, B.P., Praehofer, H. & Kim, T.G. (2000). *Theory of Modeling and Simulation. Second edition,* London: Academic Press.