A generic approach for initializing complex models

Hasina Lalaina Rakotonirainy
EDMI, University of Fianarantsoa
Fianarantsoa, Madagascar
hasina.rakotonirainy@cirad.fr

Jean-Pierre Müller
GREEN, CIRAD
Montpellier, France
jean-pierre.muller@cirad.fr

Bertin Olivier Ramamonjisoa
EDMI, University of Fianarantsoa
Fianarantsoa, Madagascar
bertinram@yahoo.fr

Abstract

Researchers are increasingly using complex models to understand socio ecosystems (SES) with biophysical and social dynamics, and their interactions. The initialization and parameterization of these models require a lot of data, and their semantics and media are heterogeneous. Despite various efforts, the proposed approaches are fairly ad hoc and no general framework has been proposed to initialize complex SES models. This paper aims at providing a generic framework for the initialization problem, bearing in mind that model initialization is not only building the initial state of the model, but also specifying time series (for example, climate series) and parameterizing processes. We propose to reformulate the model initialization issue as a transformation of data (collected by thematicians) into data structures as used by computer programs. This enables the use of Model Driven Engineering (MDE) concepts and the implementation of domain specific languages (DSL) needed to initialize the models.

I. Introduction

For around twenty years, researchers have sought to investigate socio-ecosystems [1] taking into account biophysical and social dynamics, and their interactions. To that end, modelling is an approach that is increasingly being used by researchers to understand these interaction challenges, producing increasingly complex models.

As socio-ecosystem models are complex, their initialization and parameterization require many data from socio-environmental systems, gathered by thematicians (ecologists, economists, sociologists, etc.). However, the semantics of those data, and their support (databases, Excel files, XML, CSV, shapefiles, etc.) are heterogeneous. In other words, simulation models handle data structures that do not necessarily correspond to the data gathered by the thematicians. In addition, the data structures, and therefore their needs, very often evolve during the modelling process because of the complexity of the system, which calls for continuous modification of the initialization process. It is becoming very important to have a new way of describing complex models and the existing data in order to initialize the model, and to express the transformations of data structures in a generic way, in order to be able to create all the elements of the model independently of the way the model is programmed.

In this article, a distinction is therefore made between data derived from the observation of a reality, possibly already interpreted, and data structures which are IT structures that represent the possible states of a simulation model. Data are physically stored in support (files, databases), while data structures are internal to a computer program.

When constructing socio-environmental models [2], [3], model initialization consists of constructing the initial state of the model, the specification of time series (e.g. climate series) and parameterization of the processes. To cope with this initialization problem, Hill and Vickers [4] constructed a modular language making it possible to express the hierarchical structure of the specifications of the components of a system in order to configure it. David et al. [5] constructed the XELOC language, based on the XML language, to automatically initialize a multi-agent model from semantic maps provided by thematicians. Despite these efforts, the approaches proposed are relatively ad hoc. No general framework has yet been considered for initializing complex models such as socio-environmental models.

It is proposed here to reformulate the model initialization issue in terms of a problem of transformations of data into data structures, and data structures into other data structures. Thus, the model initialization process consists of describing the sources of heterogeneous data of thematicians in order to generate the available data structures, describing the data structures of the simulation model that we consider as being functions of time, then describing the chain of possible transformations between these data structures to create the elements of the model.

We showed in [6] that by using the concepts proposed by MDE (Model Driven Engineering) [7], it is possible to formulate generically the problem of initializing and observing complex socio-environmental models by using domain specific languages (DSL). In order to be able to use MDE, we have to formulate the initialization process in terms of MDE. The purpose of this article is to propose a common framework for the initialization problem by implementing three DSLs in order to specify the data and data structures of the simulation model, with flexibility, along with the chain of transformations between them, in order to initialize the model.

In order to be generic, our second proposal is to consider that the construction of the initial state of the model, the specification of the time series (e.g. climate series) and the parameterization of the processes are one and the same thing, namely the partial specification of time functions. In fact, a simulation can be seen as the construction of a time function defined from the initial time to the final time, for which initialization is a partial specification. The initial state is the value of that function at the initial time, a time series is a sub-part of that function and parameterization specifies some time functions in different forms such as differential equations.

Thus, in section II, we introduce and justify the methodology used to design the DSLs for initializing complex models. In section III, we present the implementation of three domain specific languages L1, L3 and L2 that can be used to resolve the initialization problem for complex socio environmental models in a generic manner. Lastly, before concluding, we present some results in section IV.

II. Methodology

According to [7], MDE is a design and development methodology for model-based software. It is used to easily construct DSLs and is increasingly being used to develop complex applications [8]. One of the advantages of using DSLs is to enable modellers to focus themselves on their concerns and their field of research, in order to easily construct models that reflect the complexity of the explanations they give of reality. The application code is then generated automatically. For DSL construction, OMG (Object Management Group)[9] proposed the modelling pyramid (Figure 1) which is based on five basic concepts: the real system, the model that represents that system, the meta-model used to define the model, the meta-meta-model that serves to specify the meta-model and the transformations between the levels [10].

We showed in [6] that by using the concepts proposed by MDE, we are able to formulate the problem of initializing and observing socio-environmental models in a generic manner (Figure 2). This paper describes its implementation for initialization.

III. Use of languages L1, L2 and L3

We propose to define the following three DSLs to specify the initialization process:

- A language (L1) to specify how to create data structures using numerous data from different heterogeneous sources;
- A language (L3) to initialize the model in the form of the partial specification of a time function;
- A transformation language (L2) making it possible to map the data structures generated from language L1 and specified by language L3, in order to generate the initial state of the model with some time series and parameterization.

A. Language L1

Language L1 is a DSL used to describe the different data sources and the semantics of the available data independently from their source. The aim of L1 is to be able to generate data structures from heterogeneous data sources.

1) Principle of language L1: For language L1, we need a meta-model that makes possible to access any data source and specify the data structures to be generated independently of the data sources. The data derived from L1 might possibly be considered as the initial state of the model (the data structure of the model on its initial date), parameters, or chronological series. In the literature, some similar functionalities exist already for generating heterogeneous data sources. ETL tools (Extract - Transform - Load) such as Informatica, Talend Open Studio, GeoKettle, Pentaho DataIntegration, CloverETL are used to extract, transform and automatically generate data from one set of data support to others. The Geotools project is a free and open source GIS toolkit developed in Java. It can also be used to access and manage different georeferenced data sources. In addition, Wiederhold [11] proposed a data integration architecture called ”mediator architecture” or ”virtual integration”. The principle is to have a mediator that provides an overall scheme and unique vocabulary to express user queries. Inmon [12], Widom [13] and Chrisment et al. [14] presented the ”data warehouse” or ”materialized integration” architecture. Whilst in virtual integration data remain stored in the original source, with materialized integration data are extracted from the original source and stored in a data warehouse. Our specifications were largely inspired by these approaches and implemented using EMF (Eclipse Modelling Framework) [15]. The difference lies in the generation of data structures rather than data.

2) Meta-model of language L1: For its implementation, we used the Ecore meta-meta-model [15] of EMF. We defined a meta-model (Figure 3), of which:

- Part is used to specify heterogeneous data sources. They may be databases, Excel files, shapefiles, CSVs and even input graphic interfaces, so ultimately any sort of supports and applications that supply data.

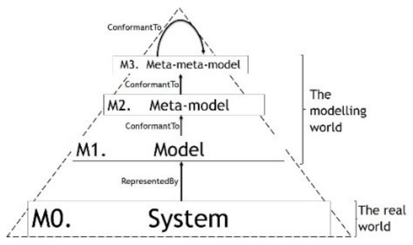


Fig. 1. OMG modelling pyramid

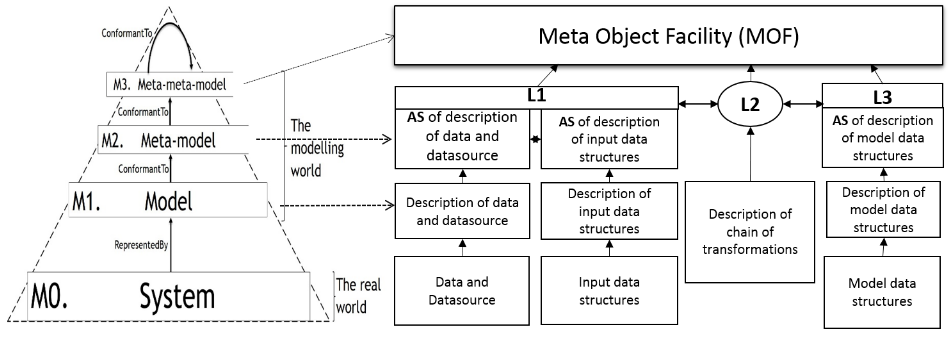
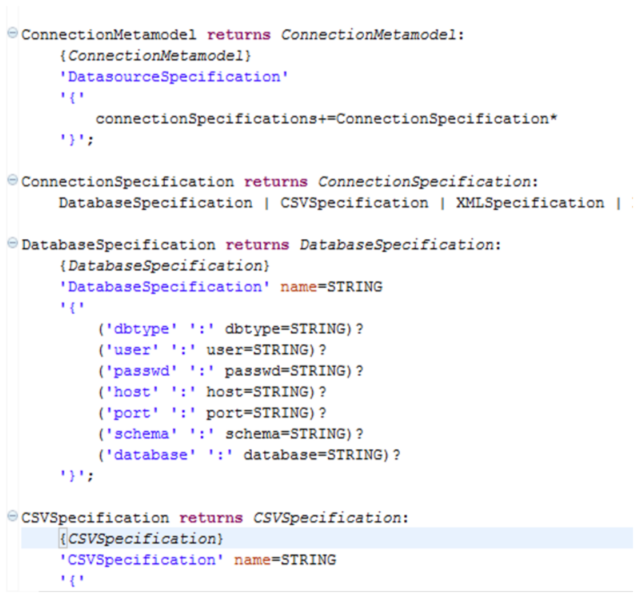

ConformantTo
M3. Meta-meta-model
ConformantTo
M2. Meta-model
ConformantTo
M1. Model
RepresentedBy
M0. System
The modelling world
The real world
Meta Object Facility (MOF)
L1
L2
L3
AS of description of data and datasource
AS of description of input data structures
AS of description of model data structures
Description of data and datasource
Description of input data structures
Description of chain of transformations
Description of model data structures
Data and Datasource
Input data structures
Model data structures

Fig. 2. Mapping the modelling pyramid to initialization implementation

- Another part is used to specify a generic data structure to be constructed from diverse data sources, because although data are stored or entered on diverse support, their processing needs them to be represented in data structures.

3) Model derived from L1: A textual or graphical concrete syntax can be associated with a meta-model. In order to construct a model that conforms to L1, we defined a textual concrete syntax with the XText [16] framework that uses the EBNF [17] metalanguage, and a graphical concrete syntax with the GMF (Graphical Modelling Framework)[18] , in addition to the tree concrete syntax automatically generated by the EMF tools. An extract of the textual concrete syntax of L1 is shown in Figure 4. Some editors based on these concrete syntaxes are supplied to thematicians to specify the data sources they use in order to generate a generic data structure. Nevertheless, it is also possible to construct the specification of data structures from metadata when they exist.

We applied this approach with a set of real heterogeneous data sources used by the Mirana model [2]. This model supplies tools for managing databases and loading Excel files and shapefiles (Figure 5). It is therefore possible to pass through the graphic interface derived from the model itself (Figure 5) to specify data sources. The concrete syntax corresponding to this specification is then generated automatically.



```
ConnectionMetamodel returns ConnectionMetamodel:
    {ConnectionMetamodel}
    'DatasourceSpecification'
    '{'
        connectionSpecifications+=ConnectionSpecification*
    '}';

ConnectionSpecification returns ConnectionSpecification:
    DatabaseSpecification | CSVSpecification | XMLSpecification |

DatabaseSpecification returns DatabaseSpecification:
    {DatabaseSpecification}
    'DatabaseSpecification' name=STRING
    '{'
        ('dbtype' ':' dbtype=STRING)?
        ('user' ':' user=STRING)?
        ('passwd' ':' passwd=STRING)?
        ('host' ':' host=STRING)?
        ('port' ':' port=STRING)?
        ('schema' ':' schema=STRING)?
        ('database' ':' database=STRING)?
    '}';

CSVSpecification returns CSVSpecification:
    {CSVSpecification}
    'CSVSpecification' name=STRING
    '{'
```

Fig. 4. Extract of the textual concrete syntax of language L1

B. Language L3

Language L3 is a DSL used to describe the simulation model.

1) Principle of language L3: Initialization not only specifies the state of the system at the initial time, but also the partial specification of certain time series (e.g. climate

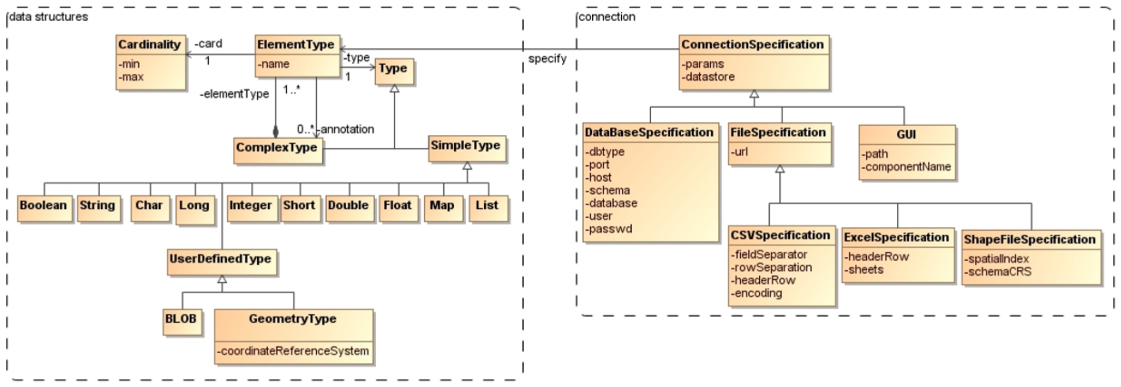
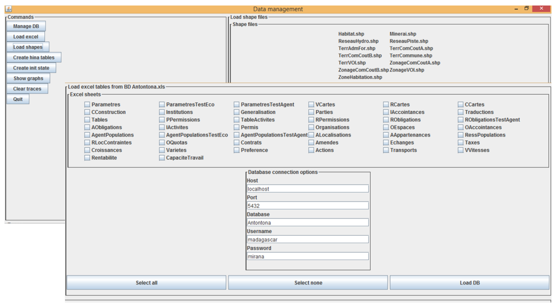


Fig. 3. Meta-model of language L1



Load shape files

Shape files

Habitat.shp, ReseauHydro.shp, TerrAdmFor.shp, TerrComCoutB.shp, TerrVOI.shp, ZonageComCoutB.shp, ZoneHabitation.shp, Minerai.shp, ReseauPiste.shp, TerrComCoutA.shp, TerrCommune.shp, ZonageComCoutA.shp, ZonageVOI.shp

Load excel tables from BD Antontona.xls

Excel sheets

☐ Parametres	☐ ParametresTestEco	☐ ParametresTestAgent	☐ VCartes	☐ RCartes	☐ CCartes
☐ CConstruction	☐ Institutions	☐ Generalisation	☐ Parties	☐ IAccointances	☐ Traductions
☐ Tables	☐ PPermissions	☐ TableActivites	☐ RPermissions	☐ RObligations	☐ RObligationsTestAgent
☐ AObligations	☐ IActivites	☐ Permis	☐ Organisations	☐ OEspaces	☐ OAccointances
☐ AgentPopulations	☐ AgentPopulationsTestEco	☐ AgentPopulationsTestAgent	☐ ALocalisations	☐ AAppartenances	☐ RessPopulations
☐ RLocContraintes	☐ OQuotas	☐ Contrats	☐ Amendes	☐ Echanges	☐ Taxes
☐ Croissances	☐ Varietes	☐ Preference	☐ Actions	☐ Transports	☐ VVitesses
☐ Rentabilite	☐ CapaciteTravail				

Database connection options

Host: localhost
Port: 5432
Database: Antontona
Username: madagascar
Password: mirana

Fig. 5. Graphic interface for loading data into the Mirana model

series), along with the parameterization of the processes, or even explicit or differential equations. Consequently, the target is not merely a state represented by a data structure but must be seen as a function of time, which will be partially specified. We therefore feel it is important to consider this total function as a theoretical object that is partially defined on initialization and completed by simulation. Formally, we therefore have the function $\Sigma : [t_{initial}, t_{final}] \rightarrow S, t \rightarrow s_t$ in S, where S is the set of possible system states. This function is also called the trajectory in [19]. The initial state becomes the specification of $\Sigma(t_{initial})$. A time series or parameterization become the specification of part of Σ.

2) Meta-model of language L3: Using Ecore [15], the language L3 serves to describe the simulation model as being a function of time. Many types of functions are possible, namely:

- $\Sigma = s \forall t$, such that the value is the same for all times during simulation;
- $\Sigma = f(t)$, where f is an explicit function of time (e.g., $a * t + b$);
- $\Sigma = df(t)$, where df is a simple differential equation of time;
- $\Sigma = F(t)$ which defines a difference equation;
- $\Sigma = step((t_1, s_1), ..., (t_n, s_n))$, which defines a constant function as bearing $t_1 \geq t_{initial}$, $t_n \leq t_{final}$ and it is considered that $\Sigma(t) = s_1$ in the interval $[t_{initial}, t_2[$, s_i in the interval $[t_i, t_{i+1}[$ and s_n in the interval $[t_n, t_{final}]$;
- $\Sigma = I((t_1, s_1), ..., (t_n, s_n))$ defines a linear interpolation or other more complex interpolation model, as $t_1 \geq t_{initial}$, $t_n \leq t_{final}$.

These functions of time can be specified in different ways depending on their nature. The meta-model that we propose to specify:

- A set of parameters (to set explicit time functions and differential equations);
- A set of parameters and a time step (to set the difference equations);
- A sequence of values, a start date and a time step (to set the constant functions per stage);
- A list of pairs $time/value$ with the type of interpolation (to set linear interpolation or other more complex interpolations models).

An extract of the meta-model is shown in Figure 6.

3) Identification of the elements to be initialized in the model: To simplify the construction of time function specification, we defined a graphic interface (Figure 7) that can be used to extract the different elements or concepts of the simulation model of interest to us, along with their properties whatever the language used for programming. It is then necessary for each selected attribute to define a time function from those proposed, in order to obtain a set of time functions that conform to language L3.

C. Language L2

L2 is a transformation language. Its purpose is to provide to the modeller a generic transformation tool that is simple to handle and capable of constructing partial specification of the simulation model as a function of time

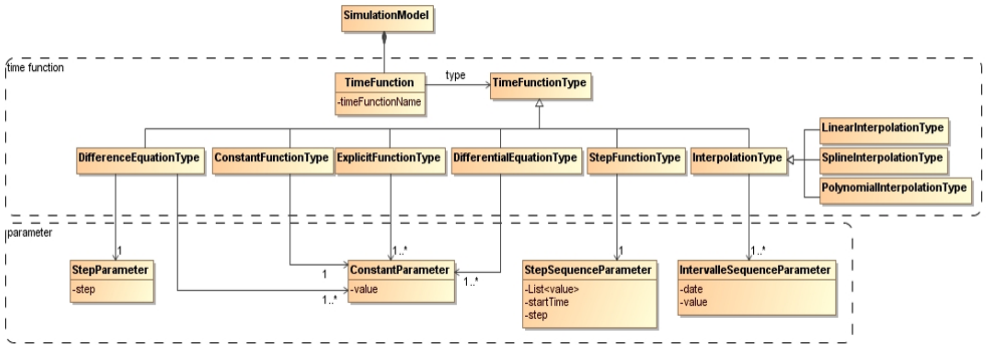
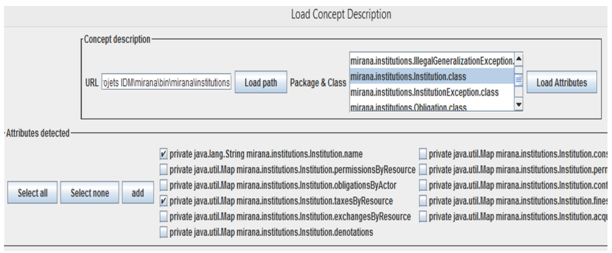


Fig. 6. Extract of the language L3 meta-model


Load Concept Description

Concept description

URL `ojets IDM\mirana\bin\mirana\institutions` Load path Package & Class

mirana.institutions.IllegalGeneralizationException.
mirana.institutions.Institution.class
mirana.institutions.InstitutionException.class
mirana.institutions.Obligation.class

Load Attributes

Attributes detected

Select all | Select none | add

- [✓] private java.lang.String mirana.institutions.Institution.name
- [ ] private java.util.Map mirana.institutions.Institution.permissionsByResource
- [ ] private java.util.Map mirana.institutions.Institution.obligationsByActor
- [✓] private java.util.Map mirana.institutions.Institution.taxesByResource
- [ ] private java.util.Map mirana.institutions.Institution.exchangesByResource
- [ ] private java.util.Map mirana.institutions.Institution.denotations
- [ ] private java.util.Map mirana.institutions.Institution.cons
- [ ] private java.util.Map mirana.institutions.Institution.perr
- [ ] private java.util.Map mirana.institutions.Institution.cont
- [ ] private java.util.Map mirana.institutions.Institution.fines
- [ ] private java.util.Map mirana.institutions.Institution.acqu

Fig. 7. GUI to extract the data structures of the simulation model

based on data derived from the data structure generated in L1 and from the model specified by L3.

1) Principle of language L2: Remember that the basic principle of model transformations is to start off from a model M_a that conforms to a source meta-model MM_a, then to specify the possible links between the source meta-model and the target meta-model in order to generate another model M_b that conforms to the target meta-model MM_b, such that MM_a may be equal or not to MM_b. In the literature, there are several types of tools that can be used for mapping, transformations or compositions of models such as QVT, ATL, AML, AMW or Kompose [10]. In the model transformation we are developing:

- Languages L1 and L3 are source meta-models, and language L3 the target meta-model;
- Part of the target model can be introduced as source model (Figure 8);
- The transformations can be in cascade.

Nevertheless, when transforming data structures into time functions, the transformation rules vary from one model to another. They depend on the modellers and cannot be defined generically. We therefore propose designing a domain specific language, adapted to thematicians that can be used to specify the transformations they wish in order to initialize and parameterize the model. To implement L2, we applied the rule-based model transformation principle [20] which more effectively met our requirements.

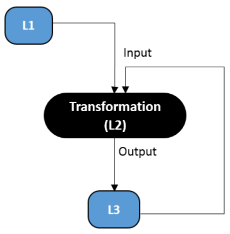


Fig. 8. Transformation of L1 to L3

2) Meta-model of language L2: The meta-model of language L2 comprises three major parts (Figure 9), namely:

- An "input" part to specify the input data structures needed for transformation. These data structures are derived from the meta-models of languages L1 and L3 (as sources);
- A "transformation" part to describe all the possible transformations of the data structures derived from L1 and L3, in order to create some new data structures and initialize and parameterize the simulation model;
- An "output" part to specify the type of output that conforms to the meta-model of language L3 (the target meta-model) after transformation.

3) Model derived from L2: The construction of a transformation model is shown in Figure 10. It can be specified from a concrete syntax such as GMF (on the left), or XText (on the right) or another. It is then possible to have transformation blocks in cascade, blocks linked to each other by inputs and outputs, to form a non cyclical graph up to obtaining the complete system.

We implemented the grammar of language L2 with XText (Figure 11). It enables the user to:

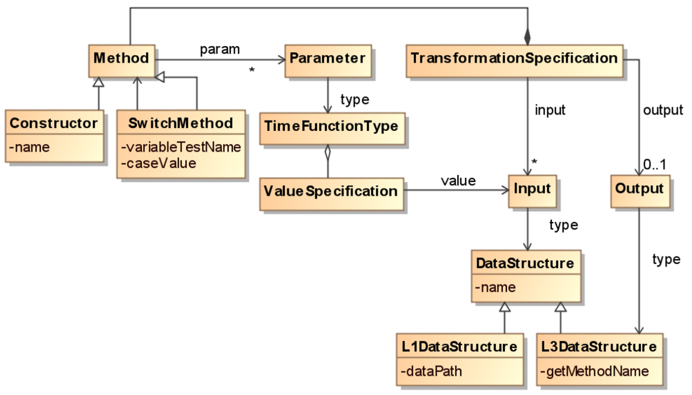
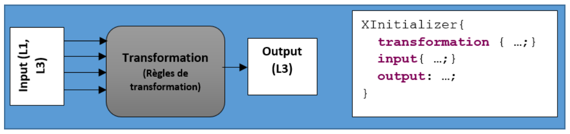
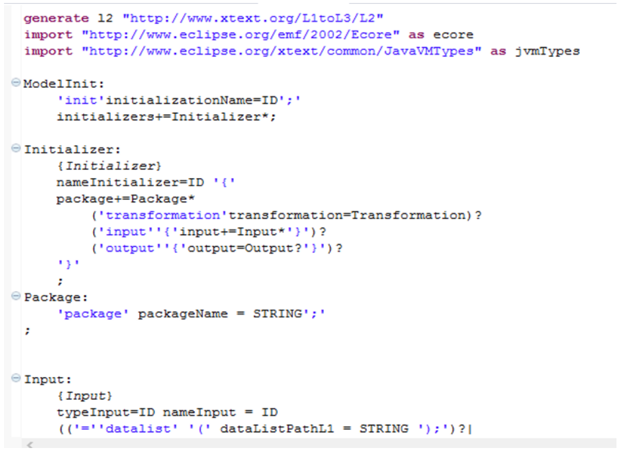


Fig. 9. Extract of L2 meta-model



```
XInitializer{
   transformation { …;}
   input{ …;}
   output: …;
}
```

Fig. 10. The three parts of L2 concrete syntax



```
generate l2 "http://www.xtext.org/L1toL3/L2"
import "http://www.eclipse.org/emf/2002/Ecore" as ecore
import "http://www.eclipse.org/xtext/common/JavaVMTypes" as jvmTypes

ModelInit:
    'init'initializationName=ID';'
    initializers+=Initializer*;

Initializer:
    {Initializer}
    nameInitializer=ID '{'
    package+=Package*
        ('transformation'transformation=Transformation)?
        ('input''{'input+=Input*'}')?
        ('output''{'output=Output?'}')?
    '}'
    ;
Package:
    'package' packageName = STRING';'
;


Input:
    {Input}
    typeInput=ID nameInput = ID
    (('=''datalist' '(' dataListPathL1 = STRING ');')?|
```

Fig. 11. Extract of L2 concrete syntax

- Construct and initialize the model from external data (L1) and from elements of the model (L3)
- Specify how to recover the value of each parameter of the time functions described in L3. In fact, the principle is to specify the path where the data used to parameterize the functions are located, then to use them during transformation.
  - For a constant function ($\Sigma(t) = s$), the specified path enables the extraction of a single parameter
  - For an explicit time function ($\Sigma = f(t)$) or for a simple differential equation of time ($\Sigma = df(t)$), the specified path enables the extraction of a sequence of values with a fixed time step.
  - For a constant function by stage ($\Sigma = step((t_1, s_1), , (t_n, s_n)))$ or a linear interpolation ($\Sigma = linear((t_1, s_1), , (t_n, s_n))$ such that $t_1 \geq t_{initial}, t_n \leq t_{final}$, the specified path enables recovery of a sequence of values with intervals.

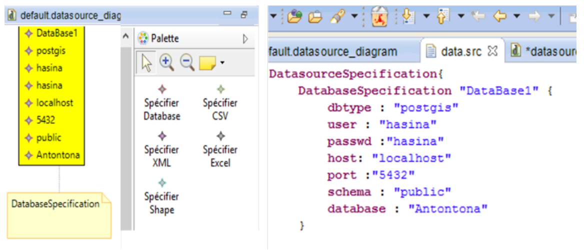
All these specifications are independent descriptions of other parts of the system.

IV. Implementation

With the DSLs L1, L3 and L2 that we propose, we are able to exploit all the heterogeneous data provided by thematicians for parameterizing simulation models.

A. Construction of the generic structure from heterogeneous data sources

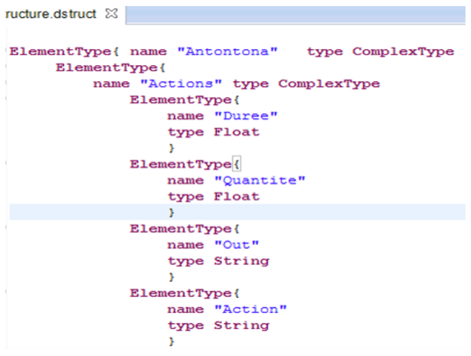
Using language L1, we can generate the input data structure of the Mirana model [2] from a set of real, heterogeneous data sources. Figure 12 shows an example of L1 graphical and textual concrete syntax use to generate a code from the specification of a data source. The generated code makes it possible to read the scheme



```
DatasourceSpecification{
    DatabaseSpecification "DataBase1" {
        dbtype : "postgis"
        user : "hasina"
        passwd :"hasina"
        host: "localhost"
        port :"5432"
        schema : "public"
        database : "Antontona"
    }
```

Fig. 12. Extract of a data source specification with L1

of the database then generate the input data structure of the model (Figure 13).



```
ructure.dstruct ⊠

ElementType{ name "Antontona"   type ComplexType
     ElementType{
         name "Actions" type ComplexType
             ElementType{
                 name "Duree"
                 type Float
                 }
             ElementType{
                 name "Quantite"
                 type Float
                 }
             ElementType{
                 name "Out"
                 type String
                 }
             ElementType{
                 name "Action"
                 type String
                 }
```

Fig. 13. Extract of the data structure generated from a specification of data sources

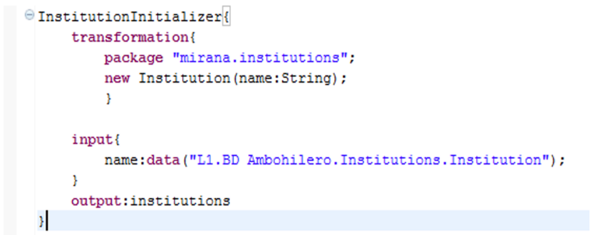
B. Specification of the model

Using the graphic interface that we described in section III-B3, we can extract all the data structures of the Mirana model with their properties. These data structures with the input data structures generated by L1 are then manipulated by language L2 to generate the application code enabling model initialization.

C. Specification of the transformation chain

With language L2, we are able to construct different data structures that can be used to initialize all the data structures of the Mirana model. We give below a few examples of using language L2 to initialize certain data structures of the model.

1) Creation of institutions in the model: In Mirana, an institution is a set of norms that are constitutive for vocabulary and regulatory for "laws". To create institutions in the model, it is necessary to specify the place where its initializer or its constructor is located, then to specify the input data needed for its construction (Figure 14). It is up to the generator to then seek the constructor in the source code of the model, along with the data from the generic structure instantiated by L1 to be able to construct elements of the "Institution" concept.



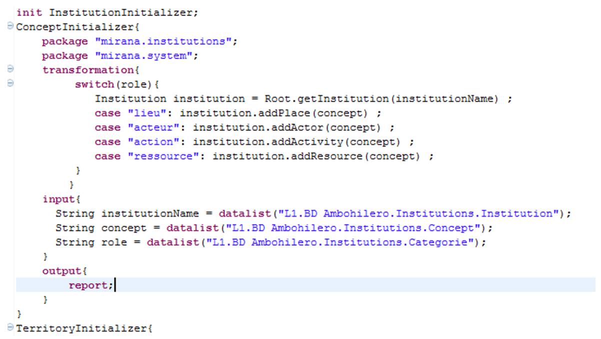
```
InstitutionInitializer{
    transformation{
        package "mirana.institutions";
        new Institution(name:String);
        }

    input{
        name:data("L1.BD Ambohilero.Institutions.Institution");
    }
    output:institutions
}
```

Fig. 14. A way of constructing institutions with L2

2) Construction of constitutive norms in the model: In the Mirana model, the constitutive norms define the terminology in the form of a set of norms structured by generalization/specialization relations and all/some relations. These norms may denote either objects (identifiers), or sets of objects (concepts or categories). For each norm, it is specified what type of object it can designate. Thus a distinction is made between:

- The roles that designate individuals seen as parts of an organization;
- The resources that designate stocks;
- The places that designate territories or parts of territories;
- The actions that designate activities.

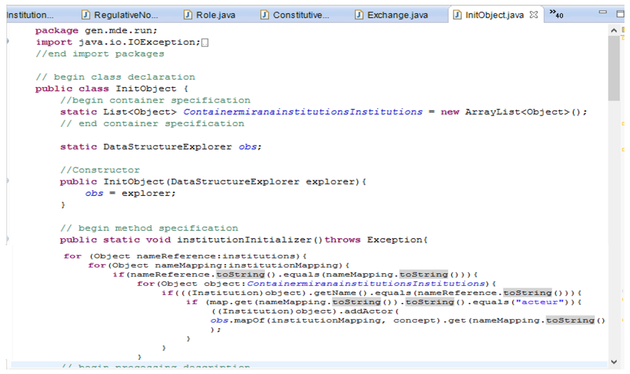
The specification with L2 described in Figure 15 can be used to generate the code that makes it possible to define the constitutive norms of each institution.



```
init InstitutionInitializer;
ConceptInitializer{
    package "mirana.institutions";
    package "mirana.system";
    transformation{
         switch(role){
            Institution institution = Root.getInstitution(institutionName) ;
            case "lieu": institution.addPlace(concept) ;
            case "acteur": institution.addActor(concept) ;
            case "action": institution.addActivity(concept) ;
            case "ressource": institution.addResource(concept) ;
         }
       }
    input{
      String institutionName = datalist("L1.BD Ambohilero.Institutions.Institution");
      String concept = datalist("L1.BD Ambohilero.Institutions.Concept");
      String role = datalist("L1.BD Ambohilero.Institutions.Categorie");
    }
    output{
        report;
    }
}
TerritoryInitializer{
```

Fig. 15. Extract of the specification of constitutive norms with L2

3) Model initialization code generator: Specification of the transformation chain with L2 enables the automatic generation of the application code to construct and initialize the model (Figure 16).



```
Institution...  RegulativeNo...  Role.java  Constitutive...  Exchange.java  InitObject.java ⊠  »40
    package gen.mde.run;
    import java.io.IOException;
    //end import packages

    // begin class declaration
    public class InitObject {
        //begin container specification
        static List<Object> ContainermiranainstitutionsInstitutions = new ArrayList<Object>();
        // end container specification

        static DataStructureExplorer obs;

        //Constructor
        public InitObject(DataStructureExplorer explorer){
            obs = explorer;
        }

        // begin method specification
        public static void institutionInitializer()throws Exception{
         for (Object nameReference:institutions){
             for(Object nameMapping:institutionMapping){
                 if(nameReference.toString().equals(nameMapping.toString())){
                     for(Object object:ContainermiranainstitutionsInstitutions){
                         if(((Institution)object).getName().equals(nameReference.toString())){
                             if (map.get(nameMapping.toString()).toString().equals("acteur")){
                                 ((Institution)object).addActor(
                                 obs.mapOf(institutionMapping, concept).get(nameMapping.toString()
                                 );
                             }
                         }
                     }
```

Fig. 16. Extract of the code generated by L2 to initialize the Mirana model

V. Conclusion

We have shown by implementing languages L1, L2 and L3 that it is possible to formulate generically the initialization problems for complex models such as socio-ecosystems, by using the concepts proposed by MDE. We have thus been able to provide some tools and know-how enabling thematicians to easily exploit the data available to them. This approach has been applied to facilitating the initialization of the Mirana complex model.

Acknowledgements

This research was funded by a PhD grant from AIRD/CIRAD.

References

[1] C. Redman, J. Grove, and L. H. Kuby, "Integrating social science into the long-term ecological research (lter) network: social dimensions of ecological change, ecological dimensions of social change," *Ecosystems*, vol. 7, pp. 161–171, 2004.

[2] S. Aubert, J.-P. Müller, and J. Ralihalizara, "Mirana: a socio-ecological model for assessing sustainability of community-based regulations," *International Environmental Modelling and Software Society (iEMSs)*, p. 8, 2010.

[3] M. Belem, R. J. Manlay, and J.-P. M, "Catmas: A multi-agent model for simulating the dynamics of carbon resources of west african villages," *Ecological Modelling*, vol. 222, no. 2022, pp. 3651 – 3661, 2011.

[4] G. Hill and S. Vickers, "A language for configuring multi-level specifications," *Theoretical Computer Science*, vol. 351, no. 2, pp. 146 – 166, 2006, algebraic Methodology and Software Technology The 10th International Conference on Algebraic Methodology and Software Technology 2004.

[5] D. David, D. Payet, R. Courdier, and Y. Gangat, in *JFSMA*.

[6] H. L. Rakotonirainy, J. Müller, and B. O. Ramamonjisoa, "Towards a generic framework for the initialization and the observation of socio-environmental models," in *Model and Data Engineering - 4th International Conference, MEDI 2014, Larnaca, Cyprus, September 24-26, 2014*, 2014, pp. 45–52.

[7] J.-M. Jézéquel, B. Combemale, and D. Vojtisek, *Ingénierie Dirigée par les Modèles : des concepts à la pratique...*, ser. Références sciences, Ellipses, Ed. Ellipses, Feb. 2012.

[8] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. C. L. Kats, E. Visser, and G. Wachsmuth, *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. dslbook.org, 2013. [Online]. Available: http://www.dslbook.org

[9] OMG, "About the Object Management Group," 2014, 05.10.2014. [Online]. Available: http://www.omg.org

[10] M. EL HAMLAOUI, "Etat de l'art sur la gestion de la cohérence de modèles hétérogènes," 2012. [Online]. Available: http://www.irit.fr/publis/MACAO/RapportIRIT-MEH-et-al.pdf

[11] G. Wiederhold, "Mediators in the architecture of future information systems," *Computer*, vol. 25, no. 3, pp. 38–49, March 1992.

[12] W. H. Inmon, *Building the Data Warehouse,3rd Edition*, 3rd ed. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[13] J. Widom, "Research problems in data warehousing," in *Proceedings of the Fourth International Conference on Information and Knowledge Management*, ser. CIKM '95. New York, NY, USA: ACM, 1995, pp. 25–30. [Online]. Available: http://doi.acm.org/10.1145/221270.221319

[14] C. Chrisment, G. Pujolle, F. Ravat, O. Teste, and G. Zurfluh, "Les entrepôts de données," in *Traité Informatique des Techniques de l'Ingénieur - H3870*, G. Zurfluh, Ed. Techniques de l'Ingénieur, février 2005, p. 10.

[15] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*, 2nd ed. Addison-Wesley Professional, 2009.

[16] "Xtext." [Online]. Available: http://www.eclipse.org/Xtext/

[17] E. S. S. Standard, "EBNF: ISO/IEC 14977: 1996 (E)," *URL http://www. cl. cam. ac. uk/mgk25/iso-14977. pdf*.

[18] E. Consortium *et al.*, "Eclipse graphical modeling framework (gmf)(2007)."

[19] B. P. Zeigler, T. G. Kim, and H. Praehofer, *Theory of modeling and simulation : integrating discrete event and continuous complex dynamic systems*, 2nd ed. San Diego, Calif. : Academic, 2000, previous ed.: 1976.

[20] D. S. Kolovos, R. F. Paige, and F. A. Polack, "Model comparison: A foundation for model composition and model transformation testing," in *Proceedings of the 2006 International Workshop on Global Integrated Model Management*, ser. GaMMa '06. New York, NY, USA: ACM, 2006, pp. 13–20. [Online]. Available: http://doi.acm.org/10.1145/1138304.1138308