Abstract—Decision making processes in some contexts, such as construction of infrastructures, require the design of prospective experiments. These experiments are designed to test hypothesis raised in the design phase. Following an adaptive methodology, these experiments will feed new iterations. In many cases, the development of these experiments is so costly that they must be performed using simulators. In this paper, an approach for developing data mining simulators based on Model Driven Engineering is presented. These simulators can be fed from data stores (i.e., multidimensional data stores) and may generate new data stores with the results they provide. This approach is intended to reduce the uncertainty at decision making as well as to speed up the iterative process.

Index Terms—data mining, data miners, simulation, decision making, complex systems, model driven engineering

I. INTRODUCTION

Many future constructions of infrastructures that are managed by companies require a thorough decision making process. This decision making process is supported by a set of prospective experiments which attempt to reduce the uncertainty so that better decisions can be made. These prospective experiments provide feedbacks that support the design so that either a decision can be made, new hypothesis can be taken into account or new prospective experiments can be designed. Sometimes, these prospective experiments cannot be performed in real for many reasons. Among them, the most common reason is that the execution of those experiments is really costly.

In silico simulations provide a way to perform this kind of experiments in a cheaper way. For instance, a design of a new highway requires many decisions to be made. However, performing prospective experiments of how the highway could be in reality would be really costly. In this case, in silico simulations can help to reproduce the way in which the cars would transit on this highway.

This kind of prospective experiments usually require the use of big amounts of data that must be analysed and used. Therefore, the use of a proper data store where this data can be stored is important. To this end, the use of a multidimensional approach for taking advantage of the data may be taken into account. A multidimensional approach for data store enhances the capabilities of taking advantage of big amounts of data. The target of this type of architectures is to support the decision making through the use of different techniques: Key Performance Indicators (KPI) design [3], data mining methods [5], information visualization patterns [7], [6]... Such techniques can be fed by the multidimensional data store.

In the example presented before, the prospective experiments that are performed using a simulation will be fed by a data store and will generate a new data set where the results of the simulation are stored. In this case, the simulations can be considered as data miners. This is, simulations are fed with data that they use for making calculus over them and generate a new data set which is used to extract conclusions.

Since the software development of these simulations is time-consuming, it is necessary a way of improving the development performance. In this paper, a framework for developing data miners based on Model Driven Engineering approach is presented. This framework is intended to speed up the creation of simulators which uses a complex system and a discrete timing. This framework uses a Model Driven Engineering (MDE) approach. Among others, the use of MDE enhances the capability of component reuse and move away implementation details providing a high level language in which models are expressed (Domain Specific Languages). Therefore, an MDE approach is proposed as a methodology for developing data miners that are fed by data stores. With the purpose of showing how to develop data miners under this framework, a case study is presented which addresses a prospective experiment of people flows in shopping centres.

II. APPROACH

The key for a successful decision making is in the information. The more information there is available, the better the decision will be made. For this reason, several processes must be started which address the information gathering. This process will reduce the uncertainty existing when making decisions.

For instance, a company that intends to increase the number of sales of its shops may retrieve information concerning the customers likes. This information can be retrieved from the sales that can be registered at every shop. After some months registering these sales, big amounts of data can be obtained. These data must be processed in a way that allows to be queried. At this point, multidimensional approaches for developing data stores can be taken into account as a method to structure the data so that these data can be queried afterwards.
However, the data stored in this structure are not outputting any interesting information per se. For this reason, different methods can be constructed allowing to take advantage of the data. These methods may involve the Key Performance Indicators (KPI) design, data mining methods or information visualization patterns, among others.

KPIs [3] are intended to develop some aggregative calculus over the data series. The way in which the data is aggregated must be designed by an expert of the domain so that finally interesting information can be extracted.

Data mining [5] methods can be fed by either KPIs calculus or raw data in order to find out valuable information on the data sets. These methods normally generate an output that can be registered in a new data store. This output can feed other methods that process data. At this level, the concept of data mining is really open as it can include statistical methods, classifiers, simulations, optimisers or pattern searchers, among others.

Information visualisation consists in using visual representations of data which step up the human cognition[7]. This is an important stage when analysing data since a proper visualisation may reveal information that could not be possible to extract using other visual representations.

A. Simulation as data miners

In organisations, decision making processes are crucial since the well-functioning of the organisation itself and third-parties depend on them. These processes require as much information as possible to make better decisions. According to the nature of the decisions, this information can be gathered in different ways.

Some decisions that are made in many organisations concern future developments that must be carried out. For instance, a new line of a subway, a new building or a construction of a power grid. In these cases, prospective experiments are necessary in order to find out the different trade-offs of every decision that is made in these developments.

This kind of prospective experiments may involve the development of simulations whenever these experiments are not possible to perform in reality. These simulations will help to identify issues and ideas that must be considered when making decisions.

In figure 1, the life cycle of a decision making process for a future development is presented. In a first iteration, and taking into account some high-level considerations about how this future development should be, some ideas are conceived. In order to find out whether these ideas will work well or not, different prospective experiments are designed. To this end, KPIs[3] must be designed since they are required to support the decision making process which will feedback valuable information in form of new ideas. These KPIs are intended to make visible information which is hidden in the data provided by the experiments. After this, the simulations must be designed and developed according to the experiments requirements. Once the simulations have been executed, results will be available. This output must be managed in a way that enables a fast querying system so that KPI calculations can be performed and used for the decision making process. This process will involve some changes in the ideas conceived which shall be tested afterwards when another iteration is initiated.

In next sections, a framework based on Model Driven Engineering for developing simulation-based data miners is presented. Under this framework, a data miner that performs a simulation can be developed. This data miner can be fed using data stores and its outputs can be stored in a new data store (Figure: 2).

B. Model Driven Engineering

Model Driven Engineering [2] [1] [4] is a methodological approach to develop software based on models. This is, the functionality of the software is defined in models and, using a processor, the software is generated according to these models. The main advantage of this approach is that the software development at the model level is conceptually described using a language that is close to the domain problem.

This methodological approach uses several abstraction levels between the software design and its implementation. These levels allow for a design of the software in a high level language where the concepts are closer to the domain of the problem in which the software will be used. In this way, not only developers, but also domain experts may have an important role on the software development.
III. PROPOSAL

The proposed framework supports the development of complex systems simulations. This framework has been addressed using a MDE (Model Driven Engineering) approach. The use of this methodology for developing complex system simulators presents the next advantages:

- Fast development of complex system simulations from the scratch
- Use of a metamodel which describes the complex system elements
- Full compatibility between simulation models developed under a concrete metamodel
- Many modellers may be working under the same metamodel and integrate their models easily
- Separation of concerns: elements description are fixed at the metamodel level meanwhile their way of acting (behavioural aspects) are variable allowing having different releases in which an element behaviour may be represented
- Modular development which allows improving the engine without affecting previously developed models
- High performance engine whose optimisation is constantly being improved
- Complete set of tools which assists the different processes for creating a new simulator
- Easy access to the experiments modification through changing only the definition simulation model

A. Architecture

The framework design is the results of a thorough analysis of several complex systems. When analysing several and diverse complex system, an abstraction process to extract common points is done (MDE). A complex system finally consists of many elements and the interactions that they have among each other. The complexity of the system consists in the amount of elements and their interactions. Therefore, a complex system description regards the definition of elements and interactions. The element definition is composed of two views: static and dynamic. The first one concerns the features description; the description of the element in terms of attributes, variables, context... The second one is related to behavioural aspects which describe the way in which an element behaves in reality. This separation of concerns enhances the possibilities to customise already developed behaviours or, even, create new ones and plug them to a model element.

The framework architecture has been developed considering these concerns. The main element is the metamodel (Figure: 3) which describes the elements that exist in the complex system to represent. This description regards their static views. This metamodel establishes a common language which allows to all modellers to speak in the same language, so that, integrating different and diverse simulation models or behaviours becomes in an easy task.

The metamodel can be translated into several formats: HTML, XSD and Java. The first one, HTML, allows to observe the metamodel elements, as well as their properties, in an easy and comfortable way by using a browser. The XSD translation returns a XML scheme model that helps to validate the model construction from a semantic point of view and provide valid next tokens to add when writing models. Finally, the most important one, the Java classes that implement the metamodel elements and their features. These Java classes are used in combination with the Simulator Engine and behaviours from the repository in order to build a simulator.

The simulation model contains the description of the scenario to be simulated and is expressed in XML. This scenario is described in terms of element instances, behaviours and interactions (by contention, id...). The elements that are instantiated have to match with the elements contained in the metamodel. Behaviours are inside the repository. Models can be built by using a XSD in a XML editor or by using the tool Profiler, which enables to create huge scenarios in an automatic manner and using data that may be incomplete.

After generating the model, the Simulator Generator will parse it in order to compile the required Java classes from the metamodel and repository. Everything is put together with the Simulator Engine in order to create the Simulator. This simulator represents a light version of the complete one as it only contains the classes that are necessary to run the parsed model.

B. Metamodel

The metamodel constitutes the main pillar of the framework architecture since it defines the environment to be simulated by describing the complex system elements that compound it. The way in which the metamodel is defined has two main processes: structure design and elements description. The structure is designed using a layers approach, so that elements...
can be categorised. After observing several complex systems, three kinds of model elements have been abstracted which configures the uppermost layer:

- **Entities.** Elements that are contained in the model scenario (E.G. understanding the power grid as a complex system, a washing machine would be an element)[8].

- **Agents.** Interpreted as intelligent actors, they are able to interact with entities or other agents and make decisions (E.G. people living in the household where the Washing Machine is)

- **Connections.** Elements that link entities (E.G. washing machine power connection to supply)

In figure 4 an example of a metamodel structure is represented. The first classification (scenario, topology and population) is related to the kinds of elements presented above. Entities are placed in the scenario, connections in the topology and agents in the population. Other taxonomies are designed according to the nature of the complex system.

Regarding the elements description, there are several aspects that can be used. The list below exposes the different points of view from where an object can be described:

- **Features.** Attributes of the element which do not change their values along the simulation (E.G. the capacity of a washing machine)

- **Variables.** Attributes of the element which change their values along the simulation (E.G. the power of a washing machine)

- **Contains.** Set of elements that can be placed within the element (E.G. a household can contain several appliances)

- **Context.** Elements that influence the element which is being defined (E.G. a radiator depends on the internal temperature of the household, so that, the radiator context is the household)

The example below presents the static description of a household. This household description has one feature, one variable, two contains and one context. As feature, the height of the household is expressed. The personCount variable indicates the number of people who are inside the household. This is variable since people leaving and joining the house would affect this value. The household can contain powerMeters and powerEquipments. A list of those powerEquipments is part of its context since such list will be used by the powerMeters to calculate the overall consumption.

```
<class name="Household" parent="Location">
  <feature name="height" type="double" units="meter" default="value="3"></feature>
  <variable name="personCount" type="int"/>
  <contain name="powerMeter" type="PowerMeter"/>
  <contain name="powerEquipment" type="PowerEquipment"/>
  <context name="powerEquipmentList" type="PowerEquipment" replicated="true"/>
</class>
```

Listing 1. Metamodel entity example of a Household

C. Model

The simulation scenario is expressed through the simulation model. This model contains a set of instantiations of the elements that are in the scenario to be described. The next example presents a simple scenario where a household containing a power equipment is represented. Furthermore, there is an agent which is linked to such household.

```
<simulation>
  <environment>
    <building>
      <household id="h0" height="2.5">
        <washingMachine>
          <behavior name="WashingMachineBehavior" releases="Operational"/>
        </washingMachine>
      </household>
    </building>
    </environment>
  </environment>
  <population>
    <familiarUnit linkedTo="h0">
      <behavior name="FamiliarUnitBehavior" releases="CoupleWithChildren"/>
    </familiarUnit>
  </population>
</simulation>
```

Listing 2. Model example

The model construction can be addressed in several ways depending on the complexity of the scenario to represent. If the scenario to represent is composed by few elements, then, it is faster to build the model by using a XSD scheme. However, when simulating huge scenarios, an automatic way to fill the simulation model is really helpful. Based on this idea, Profiler tool addresses this kind of problems where huge scenarios are wanted to be built up. For example, a scenario which

1The XSD is a schema that supports the model creation process by providing guidance.
represents a whole city cannot be modelled by hand since many buildings, facilities or power lines have to be described. In this case, Profiler may parse databases for generating the whole city. Furthermore, this tool is helpful for parameters variation so that several models can be created where the only difference among them are that some parameters have different values. This allows to study the emergent behaviour of the system according to certain parameters. Profiler also allows to parse models that are partially defined since it attends to profiler clauses which identify the sections that require to be filled.

D. Tools

Concerning the previous diagram where the whole architecture is presented, there are different tools which addresses different issues when simulating experiments. The list below briefly presents the main issue that each tool addresses:

- Model editor. This tool is part of the architecture but it is not an own development of the framework. This functionality is transferred to any model editor which supports XSD files for writing simulation models.
- Profiler. Tool for creating models in an automated manner specially oriented to develop huge scenarios based on statistical or incomplete data.
- Repository. Digital storage where behavioural aspects of the elements that are in the metamodel are described.
- Simulator Generator. Based on a simulation model, this tool compiles all needed java classes from the metamodel translation and the instantiated behaviours from the repository. All this compilation is integrated with the engine generating a simulator which precisely simulates the input model.
- Simulator. This tool is responsible for parsing the model, creating the objects and running the simulation. This tool has the ability to export results into different formats, so that, posterior analysis can be performed from the experiment.
- Metamodel browser. As well as the model editor, this tool is not a framework development. This functionality is provided by any browser since the metamodel is translated into html format.

E. Simulations life cycle

In the introduction section a usual methodology to carry out in silico experiments was exposed. One of the items within the tasks list concerned the execution of simulations from where the results are obtained. Based on many simulations we have developed, there is a common pattern about how to proceed to develop simulations. This pattern consists of four main tasks:

- Data preparation. When transferring the reality to a simulation model, part of such reality is neglected due to several reasons: no meaningful data, lack of data or computational issues. However, the data used for each simulation can widely vary from one to another. Therefore, a main task for developing simulations consists in preparing data in order to be represented in the model. Depending on the data format, coherence, completeness and complexity the task will be more arduous or not. The final goal is to have the data with a format that allows to be parsed by Profiler. However, some small experiments may not have a complicated process for preparing the data.
- Model creation. This step is divided in two main parts: creation of the simulation model and creation/modification of model elements or behaviours. The first one is mandatory for developing a simulation since in such simulation model the experiment scenario is described. The second one could be optional depending on the experiment requirements (E.G. a new element that may be needed or behaviour). Based on the data formatted in the step before, the Profiler may be helpful to generate the scenario, specially, when it is huge.
- Model simulation. The model simulation step finalizes with the results gathering. However, it does not only consist in running the model in the simulator and wait for the results, but also verifying and validating that everything is correct (this process is known as calibration). This calibration process concerns the verification of the correct working of each simulation element by checking that the outputs they have are the expected ones. Furthermore, the experiment requirements must be confronted with the simulation features, so that, it is validated if the simulation features match with what was expected.
- Result analysis. According to the experiment goals, the results must be evaluated in order to have conclusions. Sometimes, this evaluation may involve the simulation of other interesting experiments in order to check how the system works in different conditions. Even if the experiment planning is thorough detailed, many conditions can become in interesting to modify after watching the emergent behaviours, so that, new simulation experiments may be carried out.

IV. CASE STUDY

The case study presented in this section concerns prospective experiments that are developed for studying the human flows on a shopping centre from the point of view of the people amount. In order to run these experiments, the proposed framework can be used and customised to run this kind of experiments. For this, a metamodel, behaviours and models are necessary to be developed. This simulation framework will work as data miner which is fed by a data store where the statistical data concerning historical values of people flows for shopping centres are stored. Furthermore, this simulation will create new data stores allowing for the analysis of the results.

However, as the aim of this case study is not to provide truthful results, but testing the framework for complex system simulation, a non-contrasted hypothesis has been used for simulating the human flows. The hypothesis we have designed behind the human flows going shopping is expressed in the list below. Therefore, the social agents will be programmed according to the information that is contained in a data store.
A summary in natural language of the content of this data is expressed below:

- From 9.30 to 10.00, around 60-70 workers arrive to the Shopping Center, leaving it between 16:00 and 16:50.
- From 15.30 to 16.00, around 80-90 workers arrive to the Shopping Center, leaving it between 22:00 and 22:50.
- From 10.00 to 16.00, around 1000-2000 customers will arrive to the Shopping Center, leaving it between 20 and 180 minutes later.
- From 16.00 to 22.00 around 2000-4000 customers will arrive to the Shopping Center, leaving it between 20 and 180 minutes later. If such leaving time reaches the 22.00 limit, the leaving time will be 22.00.

The next sub-sections focus on the steps for developing a simulator from the scratch for running this kind of experiments. The first step is the metamodel development where it will be defined which elements are going to be used in the simulation models. Later on, the behaviours are developed which will address the way of acting of each model element. Once the metamodel and behaviours are implemented, the Profiler tool will be used for generating a huge scenario which will contain many agents and one shopping centre. Once the model is ready, the simulation is executed in order to gather the results and to be able to develop an analysis.

A. Metamodel development

For this experiment, the metamodel development is easy since it is only composed by three elements: Shopping Centre, Seller and Buyer. Note that the metamodel design is one of the most critical processes since its good design will allow having flexibility enough for adding or modifying elements without affecting the whole structure. In figure 5, the metamodel for this complex system is presented.

In the scene part, there is only one element: the Shopping centre. The topology is empty in this case and the population has two elements: the seller and buyer agent. From the point of view of the experiment context, both do the same: go in and out from the Shopping centre. However, they are conceptually separated in order to provide conceptual clarity. When running experiments, it is important to keep the concepts which exist in the reality in the metamodel. The next XML code represent the description of every metamodel element:

```xml
<class name="ShoppingCentre" parent="Location">
  <feature name="Address" type="string" default="">
    Address where the shopping centre is
  </feature>
  <variable name="personCount" type="int" default="0">
    Number of persons inside
  </variable>
</class>
```

```xml
<class name="Seller" parent="Business">
  <variable name="shoppingCentreId" type="string" required="true">
    Id of the shopping centre where the seller works
  </variable>
  <context name="shoppingCentre" type="ShoppingCentre">
    Shopping Centre where the seller works
  </context>
</class>
```

```xml
<class name="Buyer" parent="Business">
  <context name="shoppingCentreList" type="ShoppingCentre" replicated="true">
    A list containing the Shopping Centres where the buyer uses to go
  </context>
</class>
```

Fig. 5. Case study’s metamodel

B. Simulation development

The simulation development is guided by the presented Simulation life cycle which concerned four main steps. At this moment, all the ideas about how the experiment is wanted to be developed must be clear in order to decide how to proceed the different processes.

Since the objective is to test the easiness to build simulations one experiment will be developed. This one is a simple scenario where there is only two shopping centres. In this case, all the buyers of the simulation are related to these shopping centres.

1) Data processing: Since this kind of scenarios where many elements are going to be represented are hard to develop by hand, the use of an automated tool for generating simulation models is necessary. In this case, Profiler is fed by the data store where the statistical information concerning the human flows in the shopping centre is stored. Based on this information, a model that represents a concrete scenario of the people flows is generated. At this point, and through the use of the Profiler tool, the scenario can be modified as necessary according to the experiments design in order to test how the people flows would react concerning variables variation.

The table into which the hypotheses were translated contains the information about the population of the simulation model. Getting into the details of the table presented, the next columns can be observed: Agent type, Start time, Duration time, Amount and Shopping Centre. The agent type column can be observed: Agent type, Start time, Duration time, Amount and Shopping Centre. The agent type column can be observed: Agent type, Start time, Duration time, Amount and Shopping Centre. The agent type column can be observed: Agent type, Start time, Duration time, Amount and Shopping Centre.
The buyers or customers have this column empty since they can go to any shopping centre. The asterisk symbol at the last row indicates that the accomplishment of this duration time is conditioned to the gates closing in the shopping centre which has been established at 22 pm.

2) **Model development:** Using the output of the previous step the simulation model development can be addressed.

Listing 6. Simulation model where the scenario is represented containing two different Shopping Centres and many sellers and buyers. Note how behaviours programmed to accomplish the tasks of going in/out at concrete times.

3) **Result analysis:** The results incoming from the simulation are stored in a new data store allowing to be analysed. At this point, the use of the KPIs designed in the first stages of the iteration would be used so that new information can be incorporated to the decision making process. An example of KPI may be the people count according to the time.

V. CONCLUSIONS AND OUTLOOK

Future developments require the use of prospective experiments in order to find out information that may be helpful in the decision making process. In many cases, these prospective experiments cannot be fulfilled in reality since every experiment may be costly. As an alternative to this issue, simulations may be helpful in order to run these prospective experiments. In this paper, a framework based on Model Driven Engineering has been presented as a solution to create data mining simulators where prospective experiments can be designed and executed. Such simulators must be considered as data miners since they are fed by information that is considered as an input for the prospective experiments.

Based on this input, data mining simulators can represent the prospective experiment so that new conclusions can be extracted from the simulation results. The use of this framework in a decision making process can be helpful as the experiments can be rapidly performed and customised providing results which feedback the decision making process.

VI. ACKNOWLEDGMENT

This work has been partially supported by European Regional Development Fund (ERDF/FEDER) and Agencia Canaria de Investigacion, Innovacion y Sociedad de la Informacion (ACIISI) of Canary Islands Autonomic Government through the project whose reference is SolSub200801000137, and also through the ACIISI PhD grant funding to José Évora with reference TESIS20100095.

REFERENCES