Data Pipeline Interactive Simulation Using XML

Dr. S. Nirmala[1], Ms. Pavithra A[2], Ms. S. Parvathy[3], Ms. Apurva.P[4], Ms. Deepashree.N[5]

[1]Professor, Department of CSE, AMC Engineering College, Bangalore, Karnataka, India
[2,3,4,5]Assistant Professor, Department of ISE, AMC Engineering College, Bangalore, Karnataka, India

---******************---

ABSTRACT

Enhancing software that is efficient, flexible, reusable and easy to work which is a hard task for simulation developers. In this paper we propose the use of XML and its related tools (e.g. JAXB, XQuery, XSLT, and Native XML Database) for the implementation of a technology unified data pipeline targeted to interactive simulation. We introduce a technology-independent conceptual data model as the basis for every simulation framework. We show that XML is a well suited technology to be used in that context. We propose a data modeling methodology that takes its roots from Model-Driven Engineering (MDE). We would be showing a sample implementation that uses XML for transmitting data over the entire simulation loop.

INTRODUCTION

Nowadays, simulation is used extensively by scientists and engineers for designing complex systems or for under-standing intricate phenomena. Typically, "batch run simulations" are exploited for extracting knowledge from vir-tual experiments. Many batch run simulations lead to wasted computation time because errors in simulations or poor choice of simulation parameters are only discovered after a run, something that could have been avoided should interactive simulation be exploited. For being considered as interactive, a simulation should take a reasonable amount of time to execute, typically a few seconds to a few minutes.

Interactive simulation can be seen as the process of steering a simulation while it is executing. Many ap-proaches have been proposed for steering simulations interactive simula-tion is a 3-step process. Firstly, a model of the simulation needs to be built in the Simulation Modeling Module. This step consists of defining the actors participating in the simulation, the properties of each actor and the interactions between actors leading to the desired behav-iors. The scope of this step depends on the architecture of the simulator and on the level of detail required for the models. In addition to the definition of the actors and the interactions, a scenario includes the initial values for the parameters of the actors and the scheduling of "outside" events that will occur during the simulation and whose effect may be, among other things, to modify behaviors of the actors.

At the second step, the Simulation Execution Module accepts the scenario and models that were designed at the modeling stage. It is clear that the Execution Module must be structured so as to understand both the models and the scenario in order to execute the simulation properly and to maintain a coherent internal state. Then, the Execution - Module runs the simulation and updates its internal state accordingly while taking into account the events scheduled in the scenario. The "simulation state" is defined as the set of variables and parameters describing the totality of a si-mulation at a given time step. This simulation state, which is available in a given data format, is sent periodically to the Simulation Analysis Module. The latter module must provide the user with interfaces that allow him to visualize what is actually occurring in the simulation, and with tools that allow him to perform different types of analyses on the data such as statistical analysis or data mining. A major difference between interactive simula-tion and batch run simulation is that, for interactive simulation, the user "closes the loop" by providing input to the models and to the Execution Modulebased on his interpretation of the results.

We propose a design methodology that facilitates the implementation of user interaction with new and existing simulators. The methodology, presented as a conceptual framework that is a formal way of thinking of the XML based conceptional model.

In this paper, we propose a design methodology that facilitates the implementation of user interaction with new and existing simulators. The methodology, presented as a conceptual framework that is a formal way of thinking, is generic and does not make any assumption on the architecture of the simulator. The paper demonstrates that the data pipeline must be designed carefully to ensure successful implementation of interactive simulations. It intro-duces a conceptual framework for this data pipeline, which comprises a data model and a generic data flow.

CONCEPTUAL FRAMEWORK

This section presents a conceptual framework that sets up the foundation of a methodology for facilitating interaction with simulations. Figure 1 shows the key building blocks of the conceptual framework. Large rectangular boxes represent storage units, whereas rounded boxes represent processing units. Storage units encapsulate the state of data at a particular stage in the conceptual framework, whereas processing units transform input data and outputs the results.
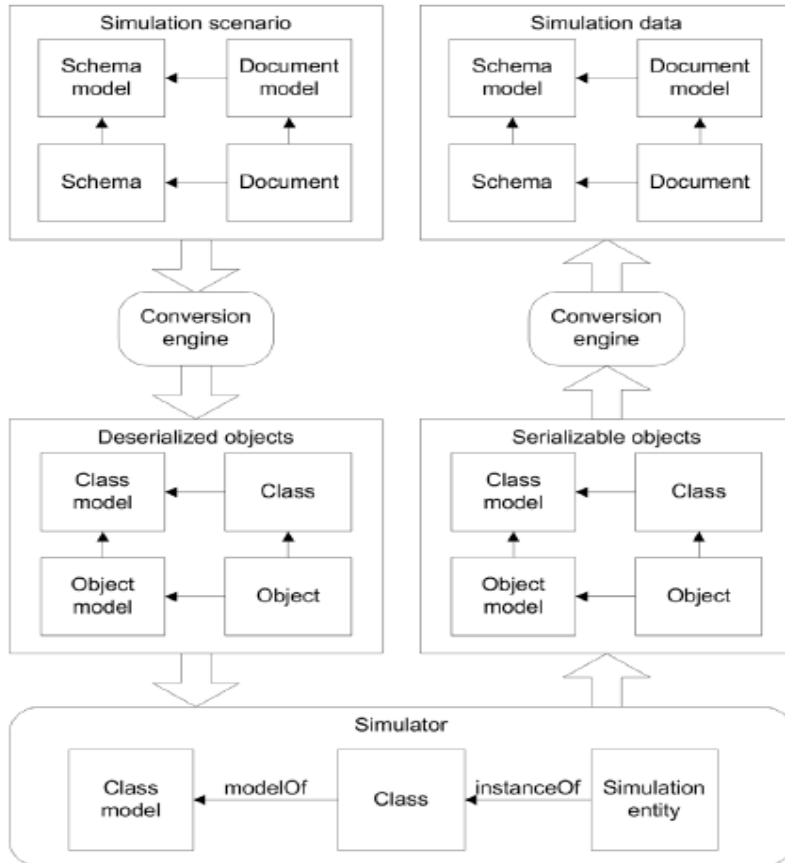

Fig 2: Technology independent conceptual framework

Data model of the conceptual framework

The first block of the conceptual framework (Figure 1) is the simulation scenario, which contains the elements described(e.g. actors, models, interactions, scheduled events). The tools that are used for building the scenarios may range from sophisticated GUIs to simple text file editors. The "simulation scenario" box, exploded in Figure 1, illustrates the data model the scenario must comply. The scenario "document," which contains the data, is an instance of a "document model." On the other hand, the "document" validates against a "schema," a model defining the document syntax. The schema itself is an instance of the "schema model," a meta-model defining the content of a schema. Once the scenario document is built and validated against its schema, it must be converted to a format that is understandable by the simulator.

For that purpose, a "conversion engine" links the si-mulation scenario and deserialized objects, which are in-termediate storage elements, by performing appropriate processing. The reason for feeding the simulator with deserialized objects instead of the scenario as such is that we want to keep the framework generic and independent of the architecture of the simulator. This genericity constraint has a direct impact on the technologiesthat need to be selected for ensuring smooth integration of the deserialized objects and the simulator. Therefore, the addition of the interactiv-ity feature to a simulator should be transparent to its inter-nal modules. It is worth noting that a single scenario usu-ally generates several deserialized objects, each being an instance of a "class."

As shown at the bottom of Figure 1, we assume that simulation entities exist in the internals of a simulator, regardless of its architecture and implementation. The simulator executes interactions between entities, which generates data that must be serialized and converted back to a format that a user understands in order to be available for analysis.

The serializable objectsare data containers in which the simulator writes the simulation state. They own the same data model as deserialized objects. A conversion engine translates serializable objects to simulation data, which shares the same data model as the simulation scenario.

Generic dataflow in the conceptual framework

Figure 2 shows a generic dataflow that is suitable for inter-active simulation. The simulation modelingstep shows that a user exploits a scenario editorto produce a scenario document. This document is usually a computer file. The elements contained in the scenario document are converted to deserialized objects and affected to the simulation state. The simulation engine performs calculations on elements composing the simulation state and updates involved enti-ties of the simulation state accordingly. Then, the simula-tion state is copied to serializable objects.

We implemented a mechanism to meet the interactive simulation requirements; serializable objects can be saved to checkpoint files, which act as simulation state containers that a user can modify and reload back to the simulator. On the other hand, the developer of the simulation modeling module of an interactive simulator needs to interface existing code with deserialized and serializable objects in order to load and save the simulation state.

A user is able to control the flow of a simulation with the simulation analysis step, which includes the following modules:

• A **data manager** whose role is to communicate with the simulation for retrieving simulation data into a stream, to manage incoming data, and to process user interaction;

• A **database** that stores selected information in-coming from the simulator data streams;

• A **data stream query** filter that selects, accord ing to the user's needs, information relevant to the simulation analysis;

• A **data analysis** module that is exploited by the user for exploring the simulation and acquiring knowledge of the phenomenon under study.

In summary, we propose in this section a framework that simulation practitioners should implement in order to convert existing software to an interactive simulator. The proposed framework allows the use of traditional simulation methodologies such as batch simulation because simulation data is stored in a database for later consultation.

XML AS A UNIFYING TECHNOLOGY

It is claimed that XML is a technology that is well adapted for the implementation of the conceptual framework described. It shows how the generic structure shown in Figure 2 can actually exploit XML technol-ogy to implement the conceptual framework. The suggested XML-based implementation makes the assump-tion that the simulator is implemented in object-oriented technology. We aim towards a design methodology that could be used by simulation practitioners to decrease the development effort when building an interactive simulator from existing software.

Originally, XML was developed as a subset of SGML, intended for web applications. It now describes data in several application areas such as semantic web, mathemat-ics, biological simulations, and military decision making. Some authors propose guidelines to follow, so that researchers use XML technology only where it belongs.

Many tools exist in order to parse and validate XML files, bind XML entities to objects of different programming languages, store XML data in databases, visualize XML documents and schemas, transform XML documents and query XML files (Wikipedia 2008). An advantage of XML over other data formats is its self-description. XML describes its structure, field names and values. The integra-tion of metadata in an XML stream is also straightforward. The resulting plain text is human- and machine-readable and fully portable on different system architectures.

On the negative side, XML is verbose, which results in a waste of bandwidth when transmitted over a network. However, some binary XML formats compress data, making it less redundant and more efficient for processing. Al-so, every piece of data is a string, eliminating the intrinsic data type support that is available in most programming languages. For the conversion between XML and common data types, the marshalling operation transforms common data types (e.g. double, float, integer) to XML strings, whereas the unmarshalling operation transforms an XML string to common data types. Nonetheless, these conversions require considerable amount of processing time and need an XML schema that defines the node types and structure of a given XML file.

SAMPLE USE OF THE METHODOLOGY

The proposed methodology was applied successfully to the implementation of an interactive simulator using existing open-source software. The system is described below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Entity" type="Entity"/>
  <xsd:element name="InitialScenario" type="InitialScenario"/>
  <xsd:element name="Position" type="Position"/>
  <xsd:element name="Random" type="Random"/>
  <xsd:element name="SimulatedEntity" type="SimulatedEntity"/>
  <xsd:element name="Simulation" type="Simulation"/>
  <xsd:element name="Terrain" type="Terrain"/>
  <xsd:complexType name="Entity">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="InitialScenario">
    <xsd:sequence>
      <xsd:element ref="Terrain"/>
      <xsd:element    maxOccurs="unbounded"    minOccurs="0"
name="entitiesList" type="Entity"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Position">
    <xsd:sequence>
      <xsd:element name="x" type="xsd:int"/>
      <xsd:element name="y" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Random">
    <xsd:sequence>
      <xsd:element name="Seed" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="SimulatedEntity">
    <xsd:sequence>
      <xsd:element name="lifeLeft" type="xsd:double"/>
      <xsd:element name="position" type="Position"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Simulation">
    <xsd:sequence>
      <xsd:element name="timeStep" type="xsd:int"/>
      <xsd:element ref="Random"/>
      <xsd:element ref="InitialScenario"/>
      <xsd:element    maxOccurs="unbounded"    minOccurs="0"
name="simulationEntitiesList" type="SimulatedEntity"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Terrain">
    <xsd:sequence>
      <xsd:element name="width" type="xsd:int"/>
      <xsd:element name="height" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Figure 3. Sample XML Schema

Pythagoras as a simulator

Pythagoras is a free, open source, agent-based simulator that models agent entities having behaviors as well as sev-eral properties (e.g. life left, position, side color) and evolving on a terrain having its own properties. This software was part of Project Albert, which aimed at using high performance computing in order to "understand theunexpected" in a military context. Pythagoras employs brute force computing in order to investigate the proposed problems. It also includes a utility for batch simulation, but does not offer facilities for interactive simulation steering. Pythagoras is implemented in Java and already exploits XML binding technology to load the scenario into its kernel.

Applying the methodology

Pythagoras lacks several features that are needed to make it interactive. Using Pythagorasoriginal source code and through a thorough reverse engineering process, we im-plemented various functionalities, in compliance. However, since the simulator was already programmed, the approach of reverse engi-neering from source code to UML, then UML to XML schema, would have slowed down the design process. Hence, the XML schema was edited by adding essential elements for dumping a simulation state to a file. The to XML in the form of streams or documents. The check-point operation becomes trivial; it consists of writing the entire simulation state to an XML document.

CONCLUSION

We showed that the methodology presented in this paper can be successfully applied for developing a specific appli-cation. However, it can be improved in several ways. First, the data modeling framework could be fully automated. In fact, using a stereotype on appropriate classes, the UML diagram could be converted to XSD, the XSD compiled to source code and methods that copy data to/from objects automatically generated. This process is relevant for a new simulator design and one that was reversed engineered. Also, we are currently designing and implementing a generic visualization environment that will allow its users to manipulate data in an immersive virtual reality environment. We plan on integrating our entire data pipeline, so that multiple simulation instances can be visualized simultaneously. Finally, the transformation of Pythagoras from a batch run type of simulator to an interactive simulator is the beginning of a long term project. We plan on modifying several additional simulators and experiment whether or not users perform better in the understanding f a complex sys-tem model using the interactive version.

REFERENCES

[1].   S.Balachandar , R.Chinnaiyan (2018), Centralized Reliability and Security Management of Data in Internet of Things (IoT) with Rule Builder, Lecture Notes on Data Engineering and Communications Technologies 15, 193-201.

[2].   S.Balachandar , R.Chinnaiyan (2018), Reliable Digital Twin for Connected Footballer, Lecture Notes on Data Engineering and Communications Technologies 15, 185-191.

[3].   S.Balachandar , R.Chinnaiyan (2018), A Reliable Troubleshooting Model for IoT Devices with Sensors and Voice Based Chatbot Application, International Journal for Research in Applied Science & Engineering Technology,Vol.6,Iss.2, 1406-1409.

[4].   M. Swarnamugi ; R. Chinnaiyan, "IoT Hybrid Computing Model for Intelligent Transportation System (ITS)", IEEE Second International Conference on Computing Methodologies and Communication (ICCMC), 15-16 Feb. 2018.

[5].   M. Swarnamugi; R. Chinnaiyan, "Cloud and Fog Computing Models for Internet of Things", International Journal for Research in Applied Science & Engineering Technology, December 2017.

[6].   G Sabarmathi, R Chinnaiyan (2019), Envisagation and Analysis of Mosquito Borne Fevers: A Health Monitoring System by Envisagative Computing Using Big Data Analytics, Lecture Notes on Data Engineering and Communications Technologies book series (LNDECT, volume 31), 630-636. Springer, Cham

[7].   S. Balachandar, R. Chinnaiyan (2019), Internet of Things Based Reliable Real-Time Disease Monitoring of Poultry Farming Imagery Analytics, Lecture Notes on Data Engineering and Communications Technologies book series (LNDECT, volume 31), 615- 620. Springer, Cham

[8].   M Swarnamugi, R Chinnaiyan (2019), IoT Hybrid Computing Model for Intelligent Transportation System (ITS), Proceedings of the Second International Conference on Computing Methodologies and Communication (ICCMC 2018), 802-806.

[9].   G. Sabarmathi, R. Chinnaiyan (2016) , Big Data Analytics Research Opportunities and Challenges - A Review, International Journal of Advanced Research in Computer Science and Software Engineering, Vol.6 , Issue.10, 227-231

[10].   G. Sabarmathi, R. Chinnaiyan, Investigations on big data features research challenges and applications, IEEE Xplore Digital LibraryInternational Conference on Intelligent Computing and Control Systems (ICICCS), 782 – 786.
        .