Developing of a High Level Architecture based Beer Game Simulation in the ADOxx Environment

Ion Dan Mironescu
Faculty of Agricultural Sciences
Food Industry and Environmental Protection
University Lucian Blaga of Sibiu
Romania
ion.mironescu@ulbsibiu.ro

ABSTRACT: *The research presented in this paper has as goal the development of a simulation framework that should support the federated simulation of the classical example in the study of supply chains – the beer (distribution) game. The approach also integrates a systematically modeling methodology – Domain Specific Modeling for the development of simulation federates. By conforming to the High Level Architecture, the developed framework has a high degree of interoperability. Although only a proof of concept in scale the resulted framework was verified and validated trough testing with a beer game setting.*

Keywords: Beer Game, Federated Simulation, HLA, Adoxx, Metamodeling

Received: 11 October 2018, Revised 27 November 2018, Accepted 7 December 2018

DOI: 10.6025/jet/2019/10/1/15-21

1. Introduction

Problem

The beer (distribution) game [1] is a good tool for simulating the dynamics and problems arising from a multistage supply chain. The current implementations [2][3][4] are developed mainly as a support for playing the serious game in a learning context. Even when they include elements of Artificial Intelligence [5] they are concentrating on the exchange between the business entities interacting on the supply chain and the decision taking. They abstract and simplify the business processes inside the entities. More important, the model of the business processes is hardcoded in the simulation without the possibility of modifying them without recoding and recompiling the whole simulation. Consequently, the problem to which this research is trying to find a solution is how to connect independent developed models in a bigger simulation of the supply chain. Independently developed models can be more complex.

State of the Art

The problem of combining complex models developed by different teams in large-scale simulation has arisen when the US military has tried to simulate combined arms exercise. Each of the separate branches for air, naval and land power have developed their own modelling and simulation solution. Not only the organizational division but also the sheers size and complexity of the combined arms simulation have required the reuse of this solution as developing from scratch the complete system was unfeasible. The solution found was to assure the interoperability of these simulators. This interoperability assures that simulations performed on the individual simulators can be combined in a single simulation. This is done by assuring the synchronisation of the simulators - so that everything happens in the same time - and the exchange of events and data that describes the interactions between the systems. As the coupling between the models executed on different simulator is loosely in comparison with models on the same simulation, this is called a federated simulation. Two technologies have emerged and then where standardised. One is Distributed Interactive Simulation (DIS)[6] which focus on the tactical scale and real time and has a decentralised architecture and High Level Architecture (HLA) [7] [8] which has a more operational and strategically focus and a more centralised architecture and time management although more by implementation then by design.

A systematic approach to the modelling is trough Domain-specific modelling (DSM) [9]. The DSM is a systems engineering method that enhance the productivity of the modelling team teams by allowing the definition of a Domain specific languages (DSL) or a meta language for the description of models. The DSL is better comprehended and fits better with the needs of the domain experts. Consequently, they can build easier models that are more expressive and can obtain more relevant data trough simulation of these models. Numerous metamodeling platforms with their corresponding metamodeling languages were developed [9]. From the most popular ones like Generic Modelling Environment (GME), MetaEdit+, ADOxx, the later has an architecture that allows the development of virtual labs and the formation of communities that can share and exchange models and simulations.

Proposed Solution
The solution that we propose in this paper will allow the easy combining at the simulation time of business entities models running in different contexts by using an architecture that comply to the High Level Architecture.

Requirements
The proposed solution is a modelling and simulation system. This system should:

- Include a modelling solution that allows the definition of the business entities in a systematic way, using a DSL for modelling;

- Include components that perform automatic inclusion in the generated models of all the common definition and specification needed for the coupling of individual models;

- Generate executable models for different simulation context;

- Provide the interoperability of these simulation contexts by providing mechanism for synchronisation and data interchange.

2. Design

Technologies
The following technologies and tools were found to be most adequate for satisfying the requirements. The metamodelling platform ADOxx supports the DSM method. It provides the possibilities of developing domain specific modelling languages with graphical notation. The platform can generate also the editor for the graphical design of the model using this notation, so that the process of developing the model is less tedious. ADOxx allows also the definition of execution algorithm for the model, so that the generated models are executable. The platform generates and manages the simulator implementing these algorithms.

The executing environment for the generated tools and for the simulation is based on micro services, so that its flexible and extensible. Other applications can have direct access (if they are in the same execution context) or through web services (if they are in different execution contexts) to the functionality of the environment, so that they can interact with the running simulation.

The IEEE 1540-2010 (HLA Evolved) [7] is a standard of interconnectivity for simulation at runtime. It includes specifications for common data models and APIs and assures the cooperation for the simulation applications that implement this specification. Beside the conforming simulation application, a cooperative simulation needs a Run Time Infrastructure, which supports this cooperation. The simulation applications are communicating only with the RTI trough API calls. This allows the possibility of simply coupling and decoupling of applications as no communication between simulations takes place. The data exchange and the event propagation take place through a publisher subscriber mechanism managed by the RTI. The RTI also assures the synchronisation through callback function which access the simulation applications. As a RTI we used the poRTIco HLA RTI.

The software is open source and has good community support. It was used to implement a similar distributed application [10]. It implements the HLA Evolved standard including the HLA Web service API. This was important as we chosen to base our solution on Web variant of the API so that we have a high flexibility.

For the definition of needed web services, we used the Olive micro-service server from the OMiLAB[11] collaborative platform that integrates seamless with the ADOxx modelling and simulation platform.

Architecture

We designed a three layer architecture which include the simulation execution environment, an intermediary translation layer and the RTI

The models are defined using special connecting ports. The *in* ports have associated data types that are specifying the data on which the model is interested or is awaiting. The *out* ports have associated data types that are specifying the data that the model is producing.

The simulations of these individual models are running in their ADOxx based execution environment that has hooks corresponding to these ports that triggers services in the intermediary layer.

The services in the intermediary layer translate the triggering actions that are essentially data generation notifications and the associated produced data in HLA Web API requests. These requests are directed to RTI, which handles them as if they were coming from a HLA federate and translates them in event notification to the subscribing simulation federates. The subscribing federate receives the notification and the data.

At the receiving federates, the intermediary layer translates the receiving event in a data transfer to the receiving port and triggers the execution actions associated with this event.

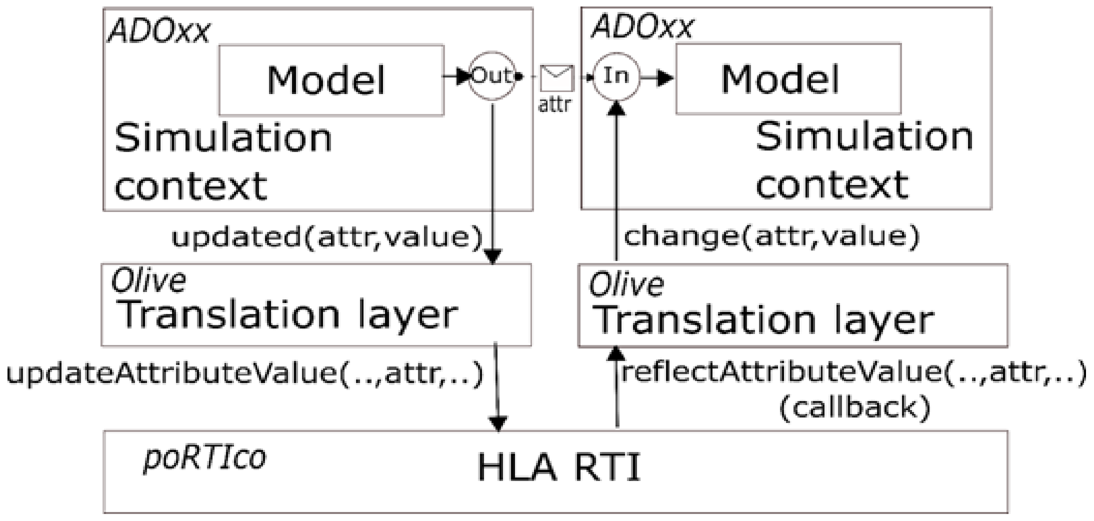

Figure 1. Architecture of the system

3. Implementation

We started from the BPMN implementation in ADOxx. Giving the dynamic of the studied system, the incoming and out coming messages where assimilated with the input and output ports. The specification of data associated with the message was included in the model specification textual description and in the graphical representation.

The translation layer was implemented as a collection of micro services in the Olive server. Some of these microservices are invoked by the local simulation context and they translate the calls in WSDL HLA API call to the RTI. The other is invoked by the RTI and transforms the call-back of the RTI in calls to the local simulation context.

The simulation execution environment was enhanced with menu entries that allow the starting of a new federated simulation, the inscribing of the model in an existing simulation, and the starting of federation simulation. These entries invoke the micro services that call the corresponding services from the federation management section of the HLAAPI.

The graphical simulation interface was modified to display all models that participate in the federated simulation and to allow the connection of *message out* and *message in* nodes that share the same exchange data model. A script was associated to the connecting action. The script calls the *publish attribute* for the model that contains the *message out* node and *subscribe attribute* for the model containing the *message in*. The unconnected *messages in nodes are* entry points with values that are introduced initially by the user. The values can be modified during the simulation, by stopping the simulation, introducing the value and continuing the simulation. Also, consequent with the philosophy of the original game, the user can modify only values for the local model and is not visualizing the progress of the complete simulation. A global view of all attributes of the game was implemented as a service that use direct RTI calls and is subscribed to all updates. This service was installed on the master system running also the master RTI component.

The code associated with the execution of the *message in* node calls the micro services that translate this in an *update attribute value* WSDL HLA service. The RTI then transmits the corresponding call back to all subscriber for this attribute.

In the translation layer we implemented, also, the micro services that respond to call-backs of the RTI from the time management section of the API. They manage the time advance of the local simulation in sync with the other simulation federates.

4. Testing

Use case

For testing purpose we used a standard 4 level configuration of a supply chain in the beer manufacturing industry. The configuration is depicted in figure 2.

For each entity, a model with its distinct simulation context was developed. A model template including the interfacing ports was developed for each level. To simplify the connection on the global model on each level, distinct exchange objects types for the ports were defined limiting the connection possibilities.

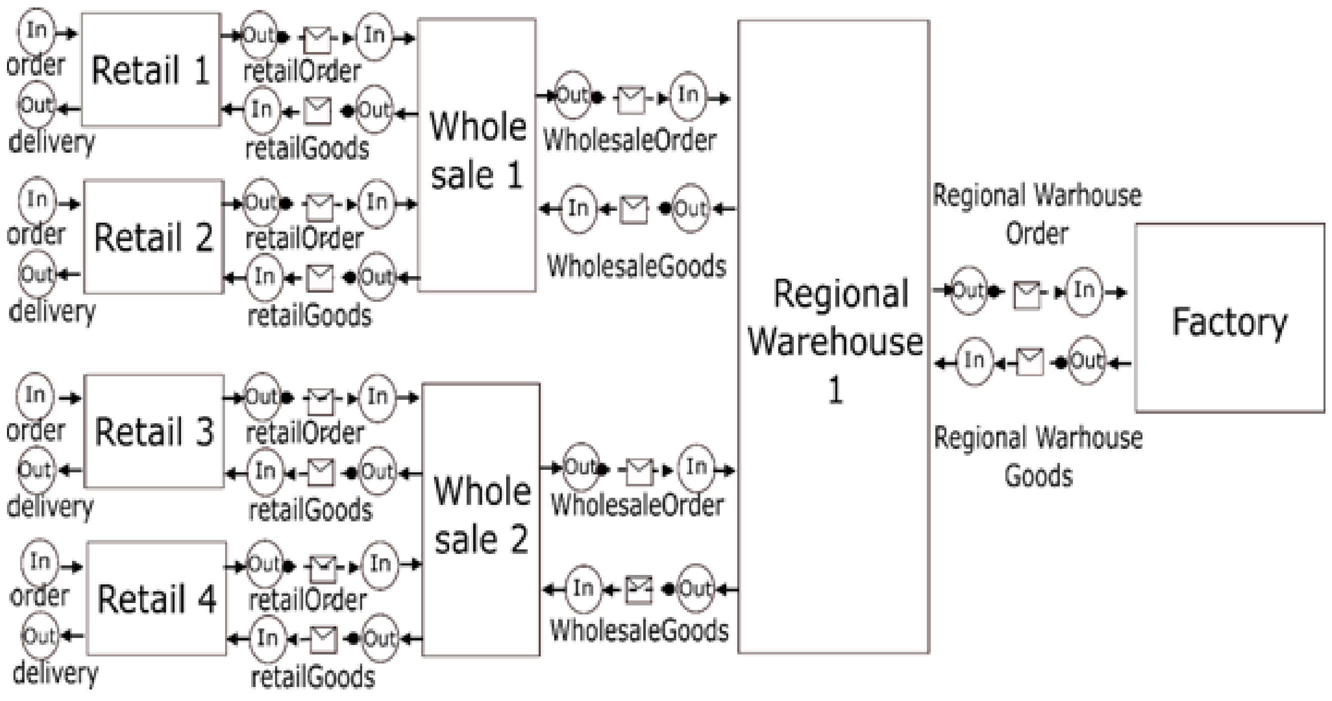

Figure 2. General configuration of the supply chain

The entities on the factory level are receiving the raw materials from suppliers and are delivering to the Regional warehouse based on the orders. For simplification, the suppliers models are included as part of the factory model. The models on this level expose two ports for connection with a model on the regional warehouse level: one *message in* for receiving the order and one *message in* for delivering the goods. An optional *message in* port for the user input of the produced quantity is also included in the standard template. The models have as parameters the production and the storage capacity. They expose the current stock of produced goods as state attribute for the global view.

The entities on the Regional warehouse level are receiving orders from Wholesale entities and deliver back to them. They are placing order to the Factories and are receiving goods from them. The template for the models of this level include the corresponding two *message in* and two *message out* ports, the optional *in port* for the user input of ordered goods, the storage capacity parameter and the current stock exposed state attribute.

The entities on the Wholesale level are receiving orders from entities on Retail level and deliver back to them. They are placing order to the Regional Warehouse and are receiving goods from them. The templates for the models are similar to the precedent level.

The entities on the Retail level are receiving order and delivering to the market and are placing order and receiving goods from the Wholesale level. Consequently, the structure of the template is similar to the precedent level, with the left side ports remaining potentially unconnected.

Using the templates, models were developed for each level. The model follows the game rules in introducing a delay (through a task node) in delivering the orders on the chain. This mimics the next turn processing of orders. It can be adequate if the decision is based on the whole day sales and stokes state at the end of the day. The model can be adapted also to model integrated supply chains were information is flowing more rapid and decision are taken more rapidly based on more frequently updated stock and sales values. Also, for simplifying and speeding up the testing process, no user input was provided. The model calculates the difference between the current stock and the order and tracks the points for penalties (for standing stock and unfulfilled orders). Then it try to minimise the resulting score trough a proportional algorithm. This mimics the uneducated approach to the game in educational context that conducts to oscillations and whiplash effect in the system.

To test beyond a linear supply chain, the models for Regional Warehouse and Wholesale where constructed with supplementary ports so that they can connect to two entities on their left side.

The final system used in each simulation configuration was a treelike structure with fore Retail stores, two Wholesale stores, one regional Warehouse and one Factory. The inputs of the Retails were linked (in the same model) with a generator node which was programmed with a step function.

To verify and validate the system and test the capabilities of mixing simulations, Java programs behaving similar to the simulations and using the same shared object model (FOM) were developed. Two variants of the Java programmes where produced: one accessing the micro services from the Olive server and one accessing directly the Java API of the RTI.

The simulation where performed in following configuration:

- Only ADOxx simulations (AD);
- Only Java using Java API (JJ);
- Only Java using the Web services (JW);
- Mixing JW and JJ simulations (JJW);
- Mixing AD, JW and JJ simulations (AJJW).

The tests were performed until all simulation give same results with identic input functions.

5. Results

The fact that the results are identical in all test variants verifies that the system fulfils the requirements. The results reproduce the whiplash effect of propagating the step perturbation introduced as a suddenly increase in the order at the Retail end of the chain. The perturbation is propagating one way as stock depleting and raise in unfulfilled orders. The system then catches up and the stocks and unfulfilled orders tend to decrease to 0. If the demand comes back to the level before the sudden increase, the perturbation propagates back to the Retail end in form of increasing stocks. This qualitatively validates the system.

Figure 3 presents the result of a simulation for which each retailer get a raise in order of about 15%. Each horizontal lane represents the plot orders versus turns for the entity named on the right side. The dotted line represents the orders received by the entity and the full line the orders given by the entity. For the factory, these are production orders and are equal to the produced quantity.

Although the results of the simulation are the same, the simulation speed is not the same with the JJ variant being the fastest, followed by the JW (7 time slower due to Web) and the AD variant being the slowest one (but only 1, 25 times slower than the JW variant). The mixed solutions were in the order of magnitude of the slowest component as the RTI waits for the slowest component to keep the simulation in sync.

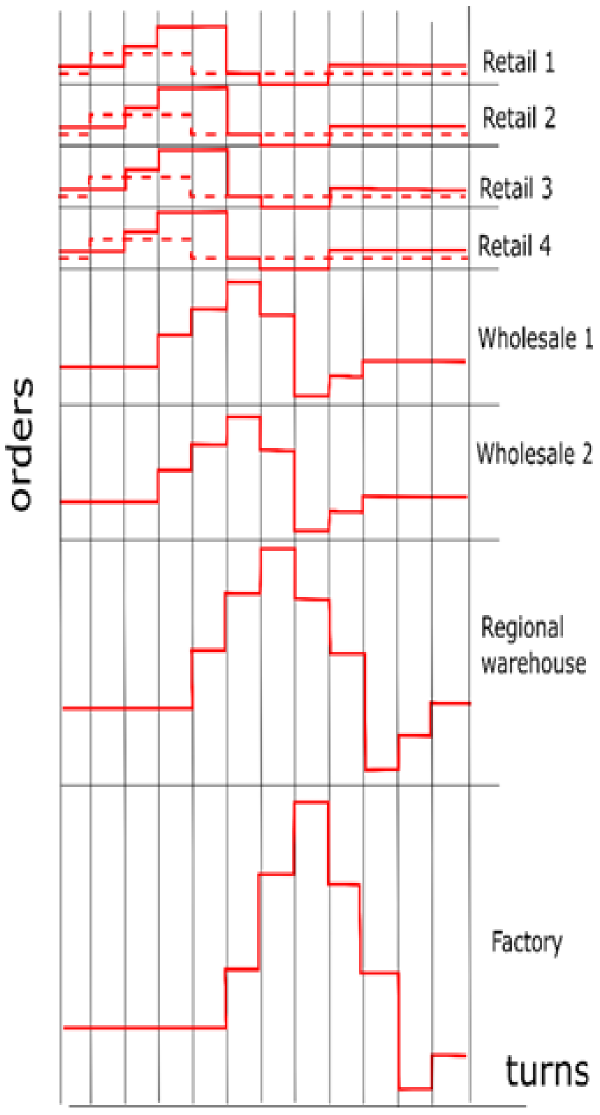

Figure 3. Result of a simulation for which each retailer get a raise in order of about 15%

6. Conclusion and Further Work

The results have demonstrated that we succeeded to develop a system that can simulate a multistage supply chain in a form similar to the beer distribution game. In comparison to other solutions, our approach allows the use of more complex models that can be developed using a Model based development method in an advanced collaborative environment.

At this stage, our approach is a proof of concept. We intend to perform more development and testing work to extend the system. The goal is to have a tool that permits the federated simulation of models developed in the ADOxx environment with simulation performed in different context (another basic meta-model in the ADOxx environment like Petri Net, or even a model defined in another modelling environment).

Another interesting direction is the development of automatic ways of generating a compiled variant of the simulation which respects the structure so that it can be integrated in federated simulations but has the advantage of speed.

References

[1] Martinez-Moyano, I., Rahn, J., Spencer, R., (2007). The Beer Game: Its History and Rule Changes. Draft paper Argonne National Laboratory https://www.researchgate.net/publication/238075204_The_Beer_Game_Its_History_and_Rule_Changes.

[2] Sarkar, B. B., Nabendu, C. (2012). A Distributed Retail Beer Game for Decision Support System, Procedia - Social and Behavioral Sciences,Volume 65, 2012, Pages 278-284.

[3] Riemer, K. www.beergame.org retrieved 10.01.2019.

[4] *** https://beergame.masystem.se/ retrieved 10.01.2019.

[5] Oroojlooyjadid, A., Nazari, M. R., Snyder, L., Takáè, M. (2017). A Deep Q-Network for the Beer Game: A Reinforcement Learning Algorithm to Solve Inventory Optimization Problems. Neural Information Processing Systems (NIPS), Deep Reinforcement Learning Symposium 2017.

[6] IEEE 1278.1-2012 - IEEE Standard for Distributed Interactive Simulation—Application Protocols.

[7] IEEE 1516-2010 - IEEE Standard for Modelling and Simulation (M&S) High Level Architecture (HLA)— Framework and Rules.

[8] Topçu, O., Oguztüzün, H. (2017). High Level Architecture. In: Guide to Distributed Simulation with HLA. Simulation Foundations, Methods and Applications. Springer, Cham.

[9] Karagiannis, D., Mayr , H.C., Mylopoulos, J., Domain-Specific Conceptual Modelling: Concepts, Methods and Tools, Springer Publishing Company, Incorporated, 2016.

[10] Tu, Z., Zacharewicz, G., Chen, D. (2011). Developing a web-enabled HLA federate based on poRTIco RTI. Proceedings - Winter Simulation Conference. 10.1109/WSC.2011.6147940.

[11] Götzinger, D., Miron, E.-T., Staffel, F.(2016). OMiLAB: An Open Collaborative Environment for Modelling Method Engineering. 10.1007/978-3-319-39417-6_3. In Dimitris Karagiannis, Heinrich C. Mayr, John Mylopoulos.