*Proceedings of the 2005 Winter Simulation Conference*
*M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.*

# FROM SIMULATION TO GAMING: AN OBJECT-ORIENTED SUPPLY CHAIN TRAINING LIBRARY

Alexander Verbraeck
Stijn-Pieter A. van Houten

Faculty of Technology, Policy and Managment
Jaffalaan 5
Delft University of Technology
Delft, 2628BX, THE NETHERLANDS

## ABSTRACT

The development of web-enabled interactive training simulations is far from easy, especially when all models have to be developed from scratch for each training game. Actually, one would like to be able to reuse parts of existing, off-line simulation models in an interactive setting. The challenge is how to set-up simulation models or simulation libraries that are developed for off-line simulations in such a way that they can be reused for on-line situations, and adapted for different educational settings. Using a supply chain context as an example, this paper shows how libraries of simulation components can be applied both for off-line simulation studies and for on-line training. The paper also describes the other functionality that is needed to create a generally applicable component library for supply chain training simulations.

## 1 INTRODUCTION

Due to the dynamic nature of supply chains, simulation is a natural and important instrument for the analysis and design of supply chains and supply chain management. Models, particularly those that offer good insight through visualization and graphs, can help companies to structure and simplify their complex and dynamic supply chains. The simulation models can help to structure, transform, condense and visually display data in such a way that managers can quickly grasp a situation and act upon the presented information (Boyson et al. 2004). Many of the supply chain models are, however, created uniquely for one specific situation, and it is hard to reuse the developed model or parts of the model for other situations.

Another aspect that complicates the modeling of supply chains is that they are heavily distributed. For the management of supply chains, there is not one central organization that has the authority to make decisions on behalf of the entire supply chain. All actors take their own decisions, based on limited information, as the other organizations do not share all their data with others. This has a severe effect on the models that are created for supply chain decision making; many decisions have to be taken based on assumptions of other organization's information or behaviour.

Many 'traditional' supply chain models disregard the fact that suppy chain information is spread over multiple organizations and that modern supply chains are highly dynamic. Trade exchanges, short-term contracts, and spot buy markets make that supply chains are not static entities, but highly dynamic networks that work largely on a pull basis rather than on a push basis. Many traditional modeling applications, therefore, deliver increasingly limited value to companies struggling with these conditions.

This is not to say that modeling is not useful in today's fast-paced supply chains. On the contrary, Boyson et al. (2004) state that modeling is more important and more needed than ever before. Because of the nature of supply chain dynamics, managers often do not have insight into the ripple effects of their decisions. Effects also can easily get lost in the overwhelming flood of data that crosses the supply chain manager's desk daily, weekly, or monthly. A rapidly changing supply chain with a continuous change of partners leads to different sets of decisions than a stable chain with long-term contracts.

Most real world logistic problems have an inherent dynamic character; objects operate in parallel, affect each other and change over time. Descriptions of the real world are moreover inherently subjective, dependent on the vision of the focal actor involved in the problem. Many modeling techniques fail to describe and design this type of chaotic and dynamic problem situations. Especially in complex supply chain problems, with many actors involved, and lots of complex technology, it is extremely important to be able to show the *different* views of different actors, and to allow for interaction with the model to

study the effects of parameter changes as seen by these actors. This may require distributing a model over more computers.

When we want to prepare students or managers for taking better decisions in dynamic supply chains, we of course want realistic interactions in the models that are used in these teaching situations. For logistics students, it is extremely instructive to study complex systems by varying parameters in different parts of the system and look at the effects on the most important performance indicators in the model. Figure 1 shows part of a supply chain model as we developed it for teaching purposes in the simulation language Arena (Kelton et al. 2002). Students can vary all kinds of parameters for the organizations in the supply chain, and study the outcome of the model, both during the run and after the run. Indicators such as inventory levels, production times, cycle times, and costs can be studied in detail per supply chain actor, as illustrated in Figure 2.

The problem with these types of teaching or training models is that it is very time consuming to develop them, it is hard to incorporate the dynamics of the supply chain itself as the structure of the model is ususally fixed, and the interaction with the model is very limited. Each student or group of students is able to make the assignments by carrying out a number of 'what-if' studies, but we are not able to demonstrate the effects of decision making on a more continuous basis. In this paper we present a Java-based architecture of supply chain components that overcomes the above mentioned problems. The components can both be used in a stand-alone 'what-if' mode and in highly interactive supply chain training games. In developing the architecture, we tried to develop the component libraries in such a way that the 'mode of use' (demonstration, what-if, interactive, competitive game) is not visible in the components themselves, but rather in additional services that are used to deploy the supply chain library for a particular mode of use. These additional services for gaming are also briefly introduced in the paper.

## 2 A BASIC SUPPLY CHAIN ARCHITECTURE

As stated before, one of the major requirements of the supply chain architecture is to be applicable in multiple settings, ranging from classical simulation to on-line gaming. In these settings, many different problem contexts may be represented, such as inventory problems (Shapiro 2001, ch.11), aliances and global integration (Simchi-Levi et al. 2003, ch. 6 and ch.8; Archibugi et al. 1999), the increasing role of customers (Fredenhall & Hill 2001), the effects of e-business on supply chains (Poirier & Bauer 2000), or the effect of portals on supply chain management (Boyson et al. 2004). Specific cases may demonstrate the effects of decisions on the bullwhip effect in supply chains (Lee et al. 1997), the effects of postponement on inventory levels (Chopra & Meindl 2001;

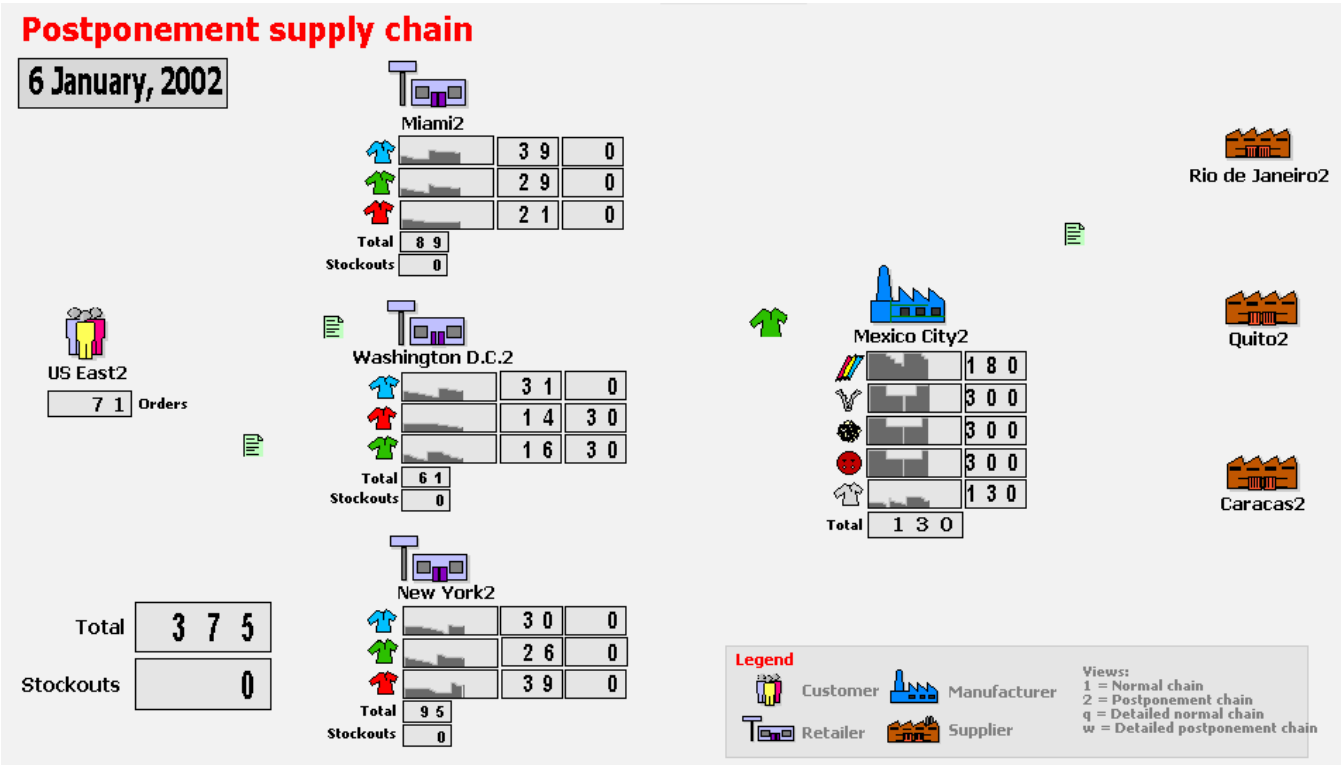


Figure 1:. Screen-Shot of a Supply Chain Teaching Game Developed in Arena

Hoek 2001), see also Figure 1, or the information that is needed for global sourcing and global sales (Waters 1999). Addressing all these different contexts asks for a set of supply chain and supply chain management building blocks that allow for flexible assembly.

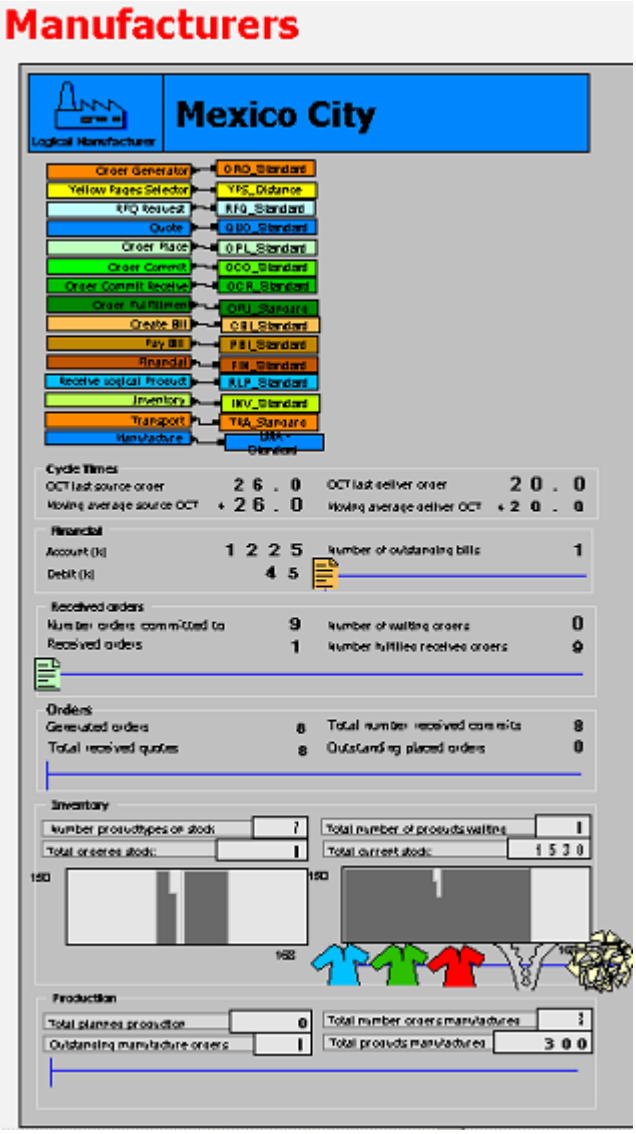


Figure 2: User Output of a SupplyChain Teaching Model

Supply chains consist of a series of companies that contribute to the production of the end product from raw material and add value to it in each step. The network structure consists of an arbitrary number of actors, which play different roles in the supply chain. In order to reach the needed flexibility, we chose for an object-oriented set-up of the supply chain library. To model the supply chain, an object model has been developed that can capture the supply chain generically. The object model focuses on the operational level; the actors within the supply chain, their roles, and the flows present between supply chain partners. Previous attempts have been made to model supply chains, but to capture the supply chain with its complexity and dynamics, the object model needs to consider the entire supply chain. This model is constructed with the goal to give a thorough description of supply chains, and secondly to form a framework for the development of a simulation library. With this simulation library, an arbitrary supply chain can be modeled and analyzed dynamically. Visualization can be put on top of the simulation to provide insight in the supply chain. The object model is made to define all relevant things in a supply chain by recognizing them as objects. All objects have a certain behavior, state and identity (Zobrist & Leonard 1997). Objects can interact using prescribed methods, and their behavior is dependent on their internal state, which can change. This state is described by the present data, which is stored in an object's attributes.

## 2.1 Requirements

The object model will be used as a framework to make a simulation library that will be used to specify different supply chain models. The products for which the simulation library is developed are analysis models, teaching models and models that will be used as the engine in supply chain portals. The object model should be suitable to describe any given supply chain for teaching models, analysis models, or decision support models.

Models for teaching purposes have different demands to the representation of supply chains than a real-life simulation model as used in the portal model and analysis models. For real-life situations and decision support, a realistic model is most important. This can imply that a company that wants to get insight in the working of its chain can visualize the flows of information and products in its supply chain The simulation of real-life supply chains will need real-life data, which is often difficult to obtain.

In a teaching environment more emphasis is placed on the outcome of the model. Concepts to be taught need to be clear to students. The visualization must emphasize the findings from their books. There is no need for a time-consuming search for real-life data. The network structure of the chain visualized however, can be extracted from real-life. This means that the object model must provide a generic model, which is suitable for both purposes.

The requirements to the object model are separated in different fields: requirements to the translation into a simulation library and requirements for the modeling of a variety of supply chains for analysis and for teaching purposes (Corver 2001).

### 2.1.1 Requirements for Modeling and Simulation

**R1. Clear Description**. To be able to make a one-to-one translation of the object model into a simulation library, the

object model must provide a clear description. This description needs the right amount of detail in order to be able to capture the supply chain dynamics and to provide a picture of the supply chain but without giving an overload of data.

**R2. Right Level of Detail.** A supply chain is difficult to comprehend because of its complexity. With a structured model it is possible to get better insight in the supply chain. The dynamics in the supply chain networks and relationships is a major part of this complexity. The object model must be able to capture this dynamics from a supply chain. In order to do this, it's necessary to understand the flows of information and products between actors. To get a controllable model, choices have to be made about the amount of detail. With a limited amount of detail the functioning of a supply chain must be described. There should also be enough detail to provide a description of a variety of supply chains. If the description is too detailed, the translation in a simulation library will be difficult and without enough details the modeling of supply chains in not possible.

**R3. Clear Structure.** The object model must give a description that provides the builder of the simulation library with a clear structure. This structure allows the one-to-one translation of the object model into a simulation library.

### 2.1.2 Requirements for Teaching and Training

**R4. Potential Link to Real Supply Chain Data.** Simulation models can be fed with real-life information in order to be able to display the current situation. To achieve this, a link will be made to the information systems and databases of the individual actors. In this connection also the desired data required in the supply chain model needs to be extracted. The object model should be prepared to handle these information systems. The object model must clearly state which information must be extracted from the information system. This information will consist of the objects, attributes and methods stated in the object model.

In a situation with dummy data, the simulation model must be able to function without extracting data from a real-life information system. For the object model there should not be a difference between the situation with or without a real-life information system.

**R5. Calculate Supply Chain Performance Indicators.** The object model must enable gathering of enough data to provide statistics. The object model must define the input data that is necessary to calculate performance measures. Using this data, model users can compare different scenarios, but also compare their organization to other companies in the same line of business. These performance measures should be taken over the entire supply chain but a drill down function will allow focusing on smaller sections of the chain. For displaying information to the users, choices should be made which data and how much data should be visible. Too much information will distract the user from the core of the problem. Insufficient information makes it impossible to make well-balanced decisions. The object model, however, should be able to provide all the desired information for different situations.

**R6. Flexible Scope.** The scope of the supply chain object model is to consider the entire supply chain if needed. When studying supply chain literature, one finds that some authors only look at the focal company or the focal company with some external links to first tier suppliers and customers. The object model must have the possibility for modeling the entire chain, but if desired, just a small part can be singled out. The reason for taking the entire chain into account is because the effect of events can go far beyond the first tier customer or supplier. For instance when the whole chain is taken into consideration the effects become clear of a transport strike on the delivery of end products to end-customers.

**R7. Multi-Actor Visualization and Performance Indicators.** In order to visualize the whole chain, different actors must be recognized. If only part of the supply chain is visualized, the focal actor and its suppliers and customers are described. Now it's important to look beyond these actors. In the supply chain literature the most commonly mentioned actors are: suppliers, manufacturers, distribution centers, retailers and transporters. These actors all have impact on the performance of the supply chain. The object model must be able to store the information about these actors.

**R8. Represent Relevant Objects.** To provide insight in the supply chain only relevant objects in the physical supply chain need to be visible. This includes the network structure and the different flows. Even though the object model should be prepared for real-life information, it is not desirable to model the actual information systems. The main reason is that for the visualization, the modeling of information systems will add nothing for the user of the display/control panel. The origin and the destination of the information flow are of interest, not the manner in which data is received. Training how to use an ERP system is usually offered as a separate course. The internal functioning of actors, including the processing of information, is also not relevant for the visualization.

**R9. Allow Capturing of a Variety of Supply Chains.** Most supply chains look like a tree where the branches and roots are the network of suppliers and customers. The focal company will deal with a number of suppliers and customers. Each of these links will have it's own suppliers and customers. The network structures will differ per product

or actor. It should be prevented that every industry needs its own template. A pharmaceutical product will require a different supply chain then for instance a car. Both supply chains should fit into the supply chain object model. The objects should be described on a generic and high level. Making the objects generic, will enable their use for different models. The aspects of a supply chain that can be recognized in every supply chain need to be described, not the specific instances for one supply chain.

**R10. Extensibility.** For specific situations, a supply chain might have to be described in more detail, so the object model must allow for extensions. An example is to allow for a more detailed description of actors. An example is that some internal processes in one version of the object model have not been taken into account. The black box, as the internal processes are now implemented, can be changed into a description with more detail but this has not been the primary goal of the object model. In preparation to extensions, several objects are present in the object model as placeholders for future extensions, such as advanced transportation modeling.

### 2.2 The Architecture and Object Model

The architecture has been set-up as a layered architecture, which captures the essence of supply chain operations and supply chain management. The first layer is formed by a set of definitions of *actors* that are able to *communicate* through *messages*.

The second layer is formed by a set of supply-chain specific actor definitions, such as `SupplyChainActor`, `Trader`, `Retailer`, `Manufacturer`, `Customer`, and `Supplier`. These actors exchange supply-chain specific messages such as `RequestForQuote`, `Quote`, `Order`, `Bill`, and `Payment`. When orders are accepted, the actors also exchange instances of a `Product` in the form of a `Shipment`. The business logic of the actors is implemented in a very flexible way. When receiving messages of a certain type, these messages are handled by instances of a message-type specific `Handler`. Each handler contains a parameterizable set of business logic that in turn can send out replies or other buisiness messages through the communication logic that is present in the actor layer. Handlers can be added or removed during the simulation run, which allows for flexible business logic. An example of a handler for handling an incoming `Bill` is the `BillTimeOutHandler`, which sends out an extra bill with a fine when the original bill has not been paid on time.

In addition, the second layer of the supply chain libraries contains reference implementations for financial management, demand generation, inventory management, production, and transportation.

After carrying out several projects with the library, it turned out that there was one very important component missing, which is not normally present in supply chain simulation libraries; the database of work on hand, promises, and history of information exchange. We dubbed this object the `ContentStore`. This object is close to the data that one could retrieve from an ERP system in a supply chain organization. The supply chain library we developed, keeps all messages that have been sent or received in the course of the simulation run in the `ContentStore`, thereby allowing the business logic to retrieve historical and status information for making decisions. For large simulations, an extension of the `ContentStore` has been defined, the so-called `LeanContentStore`, which deletes the references to messages after full delivery and payment of bills. If we would not do this for large simulations, memory would fill up fast. A possible extension that we have not yet developed would be the `DatabaseContentStore`, which would store the historical information on all exchanges with other actors in an actual database. After implementing the concept of a `ContentStore`, the implementation of business logic became much easier, because the references to all relevant information that is needed to handle incoming requests or orders, are present.

The third layer is the layer that extends the supply chain library for a certain *mode of use*. One mode of use is educational use by individual students. Individual users work with the supply chain models to be confronted with a certain aspect in supply chain management. Examples are models for experiencing the bullwhip effect, models for make-to-order supply chains versus make-to-stock supply chains, and models for comparing a normal strategy to a postponement strategy. These models can be used in pure *demonstration mode*, i.e. a teacher shows the models to a group, or in an *interactive training mode*, i.e. a student or a student team uses the model to answer some questions by carrying out a number of experiments involving different parameter settings. The layered architecture of the supply chain library is such that the individual training models should not lead to extra object classes. For all organizations (actors) in the model, several handlers are already available. If extra business logic is needed for the interactive mode (linking the handler to a user interface), the new handlers can easily replace the existing handlers in the use specific layer.

Another mode of use is the *gaming mode of use*, where some actors are controlled by the computer, and other actors are controlled by the students. Actually, actors are usually hybrid, where the students control some behavior of the organization while other business logic is carried out automatically by the algorithms in the handlers.

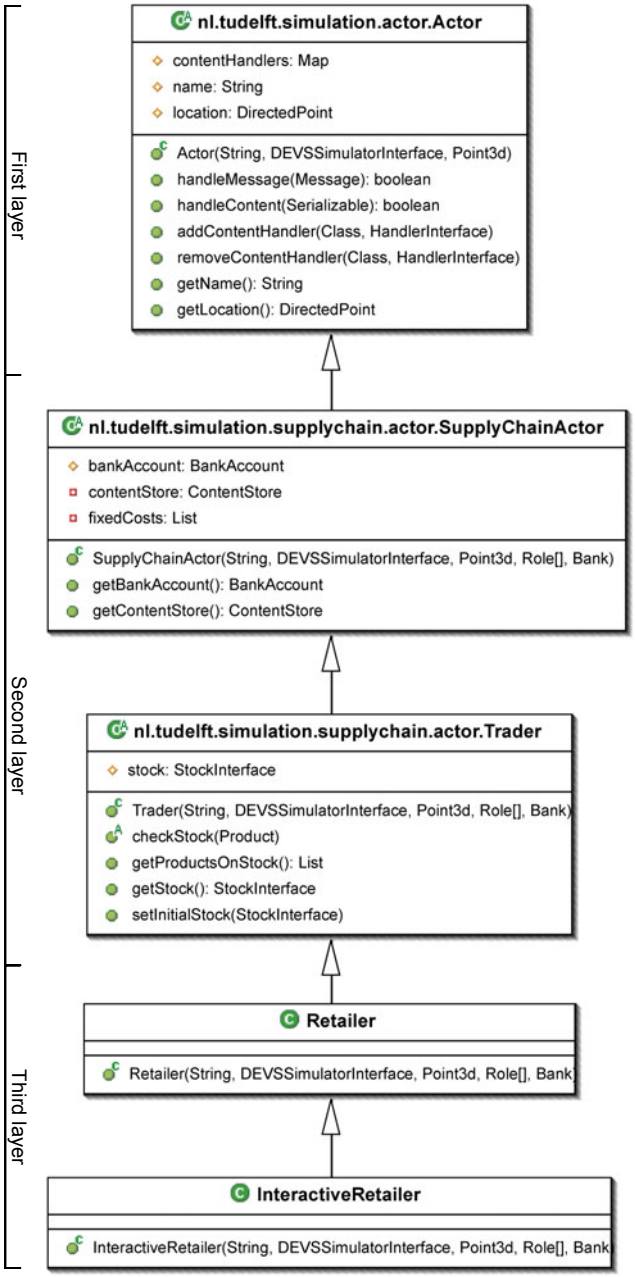


Figure 3: Example Hierarchy for the Actor Objects

It was an interesting question whether an interactive gaming mode of use could be easily designed on top of the supply chain layer. Because of the clear triggers of the business logic by messages, and the bundling of the business logic in the handlers, interactive and hybrid behavior was very easy to incorporate in the architecture. For those business messages where the students would have to make decisions themselves, the business logic is such that the relevant information for making the decision is first retrieved, for instance from the `ContentStore`. Instead of making calculations, making decisions, and send out new business messages as is the case in the fully automated handlers, the information is readied for display on the screen in the case of the interactive training mode. The player can react to the information, submit the answer in terms of the same variables that are calculated in the case of the automated handlers (e.g. price, order size, promised delivery dates), and based on that information, the `Actor` logic sends out the appropriate messages to one or more other actors, be it fully automated, hybrid, or fully human controlled actors. For an interactive mode of use, the implementation of the third layer would involve some specific handlers that are able to retrieve and prepare the information for the users, and process the fields that are based on the choices of the students, but again there is no need to change the core objects for the mode of use.

Concluding we can say that by staying close to a multi-actor, message-based supply chain architecture, we are able to use this architecture for multiple modes of use in training, without having to make time-consuming extensions to the existing architecture.

## 3 A REFERENCE IMPLEMENTATION IN JAVA

In order to test the architecture, an implementation was made in Java, on top of the Distributed Simulation Object Library DSOL, see Jacobs et al. (2002) for more information on DSOL. One of the main reasons for choosing for DSOL is that it is ready for distributed, web-based simulation, which is especially important for the individual training mode of use, e.g. portaled access to simulation models, and for the interactive gaming mode of use, where different players will have to interact with the model of the supply chain, or where the model of the supply chain is really distributed over the computers of the players, close to the structure of a real supply chain.

The three layer structure was implemented in three different projects in Java, using the DSOL environment as the simulation kernel. One project, called 'actor' is the implementation of the actor and messaging functionality. In Figure 3, an illustration is given to demonstrate how the `Actor` class is extended into an `InteractiveRetailer` class using the architectural choices as described in the previous section. The abstract `Actor` class is centered around communication (with `Message` objects), where each `Message` object has a certain `Content`. One is able to add or remove Handlers to or from `Actor` objects (a handler is defined by an object that implements the `HandlerInterface`) through the `addContentHandler` and `RemoveContentHandler` methods. Actors have a name and a geographic location (defined by a `LocationInterface`), so they are named, locatable objects that are able to communicate.

On the second layer of the supply chain libraries, the `Actor` class is extended into a `SupplyChainActor` class. The `SupplyChainActor` is an actor with a `BankAccount`.

and a ContentStore, the ERP-like data structure as explained in the previous section. The BankAccount illustrates that one of the core elements in trading is the exchange of money related to the exchange of products or services. The SupplyChainActor is still an abstract class, as we don't want users to directly instantiate this class, as the SupplyChainActor itself is still too abstract to carry any meaning in a supply chain teaching implementation.

Still in the second layer, the SupplyChainActor class is instantiated into a Trader class. A Trader is a SupplyChainActor that really owns products at a certain moment in time. In other words, the Trader has an inventory, represented by the StockInterface.

In the second layer, we also find recognizable reference implementations of actors that can really be recognized in the supply chain, for example the Retailer, Manufacturer, Customer, and Supplier. In the implementation, we could really see that the Retailer, as shown in Figure 3, did not need any extra attributes or methods to make it a retailer. The characterization of the retailer is done by giving it a specific set of handlers, and by leaving open another set of handlers to give the retailer its specific behavior.

```
public boolean handleContent(final Serializable
content)
{
  if (super.handleContent(content))
  {
    Bill bill = (Bill) content;
    try
    {
      bill.getSender().getSimulator().
        scheduleEvent(
        bill.getFinalPaymentDate()
        - bill.getSender().getSimulatorTime()
        + this.maximumTimeOut, this, this,
        "checkPayment", new Object[]{bill});
    } catch (Exception exception)
    {
      Logger.severe(this, "handleContent",
        exception);
    }
    return true;
  }
  return false;
}
```

Figure 4: Example of an Implementation of a Handler

Figure 4 illustrates how the core method of a handler can be implemented. In Figure 4, we see the handleContent method of the BillTimeOutHandler. This handler reacts to an incoming Bill. It calls its super.handleContent() method, which actually takes care of trying to pay the bill. This message in addition checks on the final payment date plus a timeout whether the bill has *actually* been paid, and schedules an event at the appropriate time to check this. At the timeout date, special measures can be taken for paying the bill anyhow, when the finances were apparently not such that the bill could be paid. One could think of business logic for borrowing money from the bank, or taking the money from a savings account to pay the bill on the timeout date to avoid a fine for paying late.

The reference implementation of the InteractiveRetailer in Figure 3, also lacks extra attributes and methods. The specific constructor of this class adds all kinds of interactive handlers to the InteractiveRetailer in addition to automated handlers for other business messages.

Please note that the implementations of the Retailer and the InteractiveRetailer are example, reference implementations. Using these classes as an example, any programmer can implement different types of supply chain actors with a different business logic in a very short amount of time.

During implementation we found out that in addition to the external messages that are sent by the other actors, we also need internal messages that are for instance the result of inventory levels dropping below a certain, pre-set value. An example is the InternalDemand message that the actor sends to itself when it might have to start ordering new products, which is handled by a certain InternalDemandHandler handler class. We also observed that in interactive gaming, other information was needed for the human players than the information present in the standard ContentStore. Therefore, a GameActorContentStore was created that stores the additional information needed for human players. As this store implements the same interface by extending the existing ContentStore, no changes in any other part of the code had to be made to accommodate this extended ERP system for the human players in the interactive game.

## 4 EXPERIENCES WITH THE SUPPLY CHAIN LIBRARY

We tested the supply chain library for the two modes of use as indicated in section 3, the demonstration / individual training mode and the interactive gaming mode of use.

For the individual use mode, three models were designed and built; exploring make-to-stock versus make-to-order strategies, comparison of normal supply chains with postponement supply chains, and a demonstration of the bullwhip effect as a result of a sales promotion. All three models could be instantiated in the DSOL-based Java library by setting the parameters for the reference actor implementations, such as the Retailer, Manufacturer, Customer, and Supplier, and by allocating the appropriate handlers, again with a number of parameters. No structural addition of object classes was needed to extend the supply chain library to specific individual training assignments.

For the interactive gaming mode, the so-called 'Distributor Game' was developed, which places students in the role of the logistics manager of a distribution center, and shows all the operational decisions that have to be taken to manage the distribution firm in terms of sales, purchases, inventory management, and financial manage-

ment. The game was implemented by instantiating the interactive reference implementations of the mode-of-use layer for the interactive players, and the standard implementations from the supply chain layer for the fully automated actors. Of course, the game implementation had to be extended with many other services that were specific for the gaming mode of use and for the game itself, but these did not lead to severe changes in the architecture in any of the three layers. The 'Distributor Game' was actually implemented and successfully used in a class of 32 MBA students in Spring 2005 at the Supply Chain Management Center at the R.H. Smith School of Business of the University of Maryland. More information about the game can be found on <http://www.gscg.org>.

## 5 DISCUSSION AND CONCLUSIONS

When matching the implementation with the requirements that were posed in section 2.1, we can see that we were able to meet most of the requirements. The supply chain library offers a clear description of supply chain concepts (R1) at the right level of detail (R2), which was shown in the interactive Distributor Game, where it was very easy to map real-world organizations on the supply chain actors as defined in the library, and to map their business logic on the developed handlers (R8). In that sense, the structure is also very clear (R3). Input (R4) and output (R5) could be easily linked to the supply chain models through the open Java-based structure of the DSOL simulation library. In addition, the `ContentStore` object, which acted as a kind of mini ERP-system, provided us with past, present and future information about the organization for which we wanted to calculate output statistics and visualization (R7). The scope of the models is flexible (R6), as there is no rigid structure with which modelers have to comply when building their supply chain models. The whole range from small supply chains with two or three actors to thousands of interacting organizations in a global network is supported with this architecture (R9). There is no lock-in on certain types of organizations, or types of products in this architecture. All classes are extensible, and interfaces are used wherever appropriate to allow programmers to extend or adapt the concepts for their specific supply chain needs (R10). When reviewing the two projects in the two different modes of use (individual training and interactive gaming) we can say that we were truly able to extend an object-oriented supply chain simulation library into a training and gaming library with all the flexibility that is needed for application in multiple contexts.

The implementation in three layers provided us with a very flexible and extendible framework for supply chain modeling with several modes of use. The advantage of the architecture chosen is that the core supply chain classes do not need to be changed for different modes of use. The separation between communicating actors on the first layer, supply chain specific actor extensions and behavioral representations on the second level, and use of specific classes in the third layer resulted in an intuitive and easy to maintain set of supply chain libraries for multiple modes of use.

## ACKNOWLEDGMENTS

The authors acknowledge support from the Supply Chain Management Center at the R.H. Smith School of Business of the University of Maryland, in particular prof. Sandor Boyson and prof. Thomas Corsi.

## REFERENCES

Archibugi, D. J. Howells, and J. Michie. 1999. *Innovation policy in a global economy*, Cambridge, UK: Cambridge University Press.

Bowersox , D.J., D.J. Closs, M.B. Cooper. 2002. *Supply chain logistics management*. New York: McGraw-Hill.

Boyson, S., T.M. Corsi, M.E. Dresner, and L.H. Harrington. 1999. *Logistics and the extended enterprise*. New York: John Wiley.

Boyson, S., L.H. Harrington, T.M. Corsi. 2004. *In Real-time: Managing the new supply chain*. Greenwood Publishers / Praeger.

Chopra, S. and P. Meindl. 2001. *Supply chain management: Strategy, planning and operations*. New York: Prentice-Hall.

Churchman, C.W. 1971. *The design of inquiring systems*, New York: Basic Books.

Corver, A. 2001. Supply chain visualization: Simulation as a means to gain insight in the supply chain. M.Sc. thesis, Delft University of Technology.

Fredenhall, L. and E. Hill. 2001. *Basics of supply chain management*, New York: St. Lucie Press.

Hoek, R. van. 2001. The rediscovery of postponement: a literature review and directions for research", *Journal of Operations Management*, 19(2), pp. 161-184.

Jacobs, P.H.M., Lang, N.A., Verbraeck, A. 2002. D-SOL: A distributed Java based discrete event simulation architecture. In P*roceeding of Winter Simulation Conference*, ed. E. Yucesan, C.-H. Chen, J. L. Snowdon and J. M. Charnes, 793-800, San Diego, California. Available via http://www.informs-sim.org/wsc02papers/102.pdf [accessed april, 15, 2005].

Kelton, W.D., R.P. Sadowski, and D.A. Sadowski. 2002. *Simulation with Arena*, 2nd Edition. New York: McGraw-Hill.

Lee, H.L., V. Padmanabhan, S. Whang, 1997. The bullwhip effect in supply chains. *Sloan Management Review*, 38 (3), pp. 93-102.

Poirier, C.C. 1999. *Advanced supply chain management*. San Francisco, CA: Berrett-Koehler Publishers.

Poirier, C.C. and M.J. Bauer. 2000. E-Supply Chain – Using the internet to revolutionize your business. San Francisco, CA: Berrett-Koehler Publishers.

Shapiro, J.F. 2001. *Modeling the supply chain*. Pacific Grove, CA: Duxbury.

Simchi-Levi, D., P. Kaminsky, and E. Simchi-Levi. 2003. *Designing & managing the supply chain*, 2nd Edition. New Yrok: McGraw-Hill.

Sol, H.G. and P.W.G. Keen. 2005. Decision support next generation. (forthcoming).

Waters, D. (Ed.). 1999. *Global logistics and distribution planning*. 3rd Edition. Boca Raton: CRC Press.

Zeigler, B.P., H. Praehoffer, and T. G. Kim. 2000. *Theory of modeling and simulation, second edition: Integrating discrete event and continuous complex dynamic systems*. Elsevier Science, San Diego.

Zobrist, G.W. and J.V. Leonard (Eds.). 1997. *Object oriented simulation: reusability, adaptability, maintainability*. New York: IEEE Press.

## AUTHOR BIOGRAPHIES

**STIJN_PIETER A. VAN HOUTEN** is a Ph.D. student at Delft University of Technology. His research is focused on gaming services for decision support environments.
His e-mail address is
s.p.a.vanhouten@tbm.tudelft.nl
and his webpage is
www.tbm.tudelft.nl/webstaf/stijnh.

**ALEXANDER VERBRAECK** is chair of the Systems Engineering Group of the Faculty of Technology, Policy and Management of Delft University of Technology, and a part-time full professor in supply chain management at the R.H. Smith School of Business of the University of Maryland. He is a specialist in discrete event simulation for real-time control of complex transportation systems and for modeling business systems. His current research focus is on development of generic libraries of object oriented simulation building blocks in C++ and Java.
His e-mail address is
a.verbraeck@tbm.tudelft.nl,
and his web page is
www.tbm.tudelft.nl/webstaf/alexandv.