An Agent-Based Formal Framework for Modeling and Simulating Supply Chains

Li Tan[1], Shenghan Xu[2], Benjamin Meyer[1], and Brock Erwin[1]

[1] School of Electrical Engineering and Computer Science
Washington State University, Richland, WA 99354
[2] College of Business and Economics
University of Idaho, Moscow, ID 83843

Abstract

We propose an open and extensible agent-based formal framework for modeling and simulating supply chains. Since structures and behaviors of supply chains can be very different based on underlying business models and markets, most of existing simulation and modeling tools are only applicable to specific subsets of supply chains. To improve extensibility, a distinctive feature of our approach is that it separates the functionalities of an element from its role and handles interactions among elements in an agent-based framework: elements are modeled as agents and their interactions decide the behavior of a supply chain. Our framework provides formal definitions for the syntax and semantics of an element. The framework separates internal behaviors of an element from its interface. These features make it easier to define new types of elements and customize their behaviors for a variety of supply-chain applications. The framework also gives rigid simulation-based semantics for a supply-chain model. The formalism it introduced helps an analyst understand and validate simulation results precisely and rigorously. The formal framework also facilitates automated formal analysis of a supply chain [7]. We discuss the implementation of our framework in context of SIMRISK*, a supply chain simulation and analysis tool we developed.*

1 Introduction

Supply chains concern the movement of merchandise from suppliers to customers' hands. A streamlined supply chain is a vital asset for many companies that compete in today's global market. To reduce cost and maintain profit margin, many companies engage themselves in the global supply chain expansion involving suppliers, distributors, retailers, and logistics providers across multiple continents [3]. For example, Costco operates its 544 warehouse stores in North America, South America, Asia, and Europe. It sources merchandise from all over the world [2]. With the world facing recession and businesses facing stiffer competitions, streamlining supply chains is increasingly important to a company's bottom line. Understanding the behavior of a supply chain is an important step in supply chain management and a precondition to supply-chain optimization.

Simulation remains an important tool for understanding the behavior of a supply chain. Compared with other methods such as stochastic analysis, simulation does not heavily tax one's theoretical background and analysis skills. Simulation results can be visualized and explained to executives and other stakeholders with limited or no background on supply-chain management. With the introduction of more powerful hardware, especially emerging multi-core architecture, there is renewed interest in recent years in desktop-based simulation tools for supply chains [8].

One of our motivations is to provide an agent-based approach for analyzing complex supply chain systems. With a growing number of businesses considering oversea suppliers as a way to cut cost, a global supply chain operation often consists of large numbers of facility nodes including suppliers, warehouses, and retailers, each of which can be seen as an (semi-)autonomous decision maker. Agent-based modeling has been successfully used to study a wide range of complex systems [1]. These systems typically consist of groups of autonomous agents. We adopt an agent-based approach because the challenge of simulating complex supply chains is the exact problem agent-based approach is prescribed for: although the behavior of each facility node may be simple to understand, the overall behavior of a supply chain as the result of interactions among these nodes is not. Our framework models elements in a supply chain as agents, which include facility nodes (e.g. warehouses) and other elements (e.g. routes). In our framework, an analyst defines the internal behavior of an element as well as its interactions with other elements via interfaces and messages. The simulation engine simulates the behavior of a supply chain by computing interactions among elements.

One goal of this research is to provide an open and extensible framework for supply-chain modeling and simulation. The structure and behavior of a supply chain are heavily influenced by the underlying business model and market. For example, Walmart's supply chain operation follows a more traditional setting. It contains multiple echelons of suppliers, distribution centers, and stores, whereas Dell uses a direct marketing approach that eliminates intermediate layers. These two supply chains not only differ in structure, but also in functionality and strategy at the element level. For instance, in Dell's "build-to-order" model, supplier sites also include manufacturing capabilities that can assemble custom-built computers, whereas for most retail chains, suppliers are simply warehouses of merchandise providers. Because of differences and complexity in supply chain operations, most supply-chain tools choose to target only a specific set of supply chains with limited extensibility. In contrast, our agent-based formal framework provides an analyst a higher degree of flexibility: he can add new elements by instantiating an existing element type, redefining an existing type, or even introducing a new type of element. From supply-chain elements, we abstract both their structural components such as ports, and their behavior factors such as internal merchandise transformation. To define a new type of element, an analyst just needs to supply the configuration of these components and factors.

Our agent-based formal framework gives formal definitions for the syntax and semantics of an element. It also defines formal simulation-based semantics for a supply chain. One problem with many existing supply-chain simulation tools is that they lack a formal definition of a supply-chain model. This makes it harder to validate a simulation result since the expected behavior of a supply-chain model is not rigidly defined. For the same reason, it is hard to tell bugs from legitimate features in tool implementations. The formalism we introduced has several benefits: it helps validate simulation results and also reduces errors in tool implementations; the formalism also facilitates formal analysis of a supply chain. For example, our formal framework supports a model-checking-based risk analysis approach in [7].

The rest of the paper is organized as follows: in Section 2 we discuss related works; in Section 3 we introduce our agent-based modeling framework. We formally define the interface of an element (Section 3.1), its internal and external behaviors (Section 3.2), and constraints (Section 3.3). In Section 4 we introduce our simulation algorithm and define a simulation-based formal semantics for a supply chain model. Section 5 discusses SIMRISK, a supply-chain modeling, simulation, and analysis tool that implements the proposed framework. Finally, Section 6 concludes the paper and discusses future research.

2 Related Works

Supply-chain modeling and simulation have attracted much research interest recently. In [4] Liu *et al* introduced a Java-based supply-chain simulation tool Easy-SC. In Easy-SC modeling environment, facilities were modeled as instances of pre-defined six enterprise nodes, and routes were implemented as connection arcs. in [8] Wang *et al* discussed a general business simulation environment (GBSE) developed by IBM China research lab. GBSE was a Java-based event-driven simulation tool built on top of the Eclipse platform. GBSE defined three types of facility nodes and one type of link. In contrast to these tools, our approach provides a higher degree of extensibility by allowing an analyst to define his own type of supply-chain element.

Agent-based approaches have been explored by other researchers for simulating supply chains. Swaminathan *et al* [6] proposed a multi-agent approach for modeling supply chains. They believed a multi-agent approach is "*a natural choice*" for modeling supply chains because "*supply-chain management is fundamentally concerned with coherence among multiple decision makers*". They modeled structural elements as agents, which interacted with each others using control elements. Our agent-based research follows the same line but one of our improvements is to formally define the syntax and semantics of an agent.

On the implementation side, Rossetti *et al* [5] used a objective-oriented framework to implement a Java package for supply-chain simulation. We also use an object-oriented type system in SIMRISK, but do so within our agent-based formal framework.

3 Agent Modeling

Agent-based modeling studies interactions among autonomous agents. Supply chains fit nicely into the profile of agent-based modeling: a supply chain usually has a large collection of facility nodes connected by routes. For example, a regional supply chain can have hundreds of retailers serviced by dozens of warehouses. Interactions among facility nodes make it very challenging to understand and predict risks embedded in supply chains. Our modeling technique is built based on agent-based modeling with special attention for extensibility.

In our framework, elements in a supply chain are modeled as agents. These elements include facility nodes and routes connecting them. Agent modeling includes physical modeling and behavior modeling. Our goal for agent modeling is to define a general and extensible model that can (1) model a variety of elements in a supply chain including facility nodes and routes and (2) provide a rigid syntax and semantics definition for an element. Definition 1 gives the formal definition of an element. Next, we model the interface of an element using ports (Definition 2), then proceed with behavior modeling including merchandise transformation (Definition 2), message sending (Definition 7), delivery decision (Definition 8). Finally we discuss constraints (Definition 9) imposed on an agent.

Definition 1 (Element) *An element of a supply chain is defined as a tuple $\langle P, U, f_t, f_m, f_d, C\rangle$, where P is a set of ports, U is a set of modes, f_t is a merchandise transformation function, f_m is a message sending function, f_d is a delivery decision function, and C is a set of constraints.*

3.1 Ports and Deliveries

The interface of an agent is defined by ports. Depending on the direction of its merchandise flow, a port is either an in-port or an out-port. A duplex port may be defined as a combination of an in-port and an out-port. Each port may buffer flow up to a given *inventory* size.

An element may consist of both in- and out-ports. These ports are the interface of the element via which the element receives and/or delivers merchandise. Definition 2 gives the formal definition of ports. We will use the following notations in the rest of this paper: we denote $a.p_i$ for port p_i of element a, and $a.p_i.inv$ for inventory at port p_i. We denote $a.P^+$ and $a.P^-$ for all a's in-ports and out-ports, respectively. We also denote Z for the set of all integers, and R for the set of real numbers. Z^+ and R^+ represent non-negative numbers of these sets, respectively.

Definition 2 (Ports) *A port may contain its own buffer. It is either an in- or out- port. Formally, the state of a port is defined as a tuple $p = \langle inv, dir\rangle$, where $inv \in Z^*$ is the amount of merchandise stacked at p and $dir \in \{+, -\}$ is the direction of the port. p is an input port if $p.dir = +$, or an output port if otherwise.*

Definition 3 (Deliveries) *Let p^- be an out-port of an element that is connected to input ports $p_1^+, \cdots p_l^+$ of some elements, a delivery from p^- is a vector $\hat{\delta}_{p^-} = \langle \delta_{p^-p_1^+}, \cdots, \delta_{p^-p_l^+}\rangle$, where $\delta_j \in Z^*$ is the amount of merchandise delivered to p_j^+ from p^-.*

By Definition 3, a delivery at an out port p^- is characterized by amounts of merchandise delivered to receiving in-ports. The total delivery received by an element a in a supply chain $\mathcal{S}$ is defined as,

$$\sum_{p^- \in \mathcal{S}.P^-} \sum_{p^+ \in a.P^+} \delta_{p^-p^+}$$

The total delivery received by a is defined as,

$$\sum_{p^+ \in \mathcal{S}.P^+} \sum_{p^- \in a.P^-} \delta_{p^-p^+}$$

3.2 Modeling Agent Behaviors

Definition 4 (Modes and States) *A state of an agent a is defined as a tuple $\langle p_1.inv, \cdots, p_l.inv, u\rangle \in (Z^*)^l \times U$, where $p_i.inv$ is the inventory of i-th port of a and U is the set of a's modes.*

An element has a finite set of modes, which are used to model states of its internal processes. A state of the element is defined by its mode and inventory at its ports. We use the following notations in the rest of the paper: given a state $s = \langle p_1.inv, \cdots, p_i.inv, \cdots, p_l.inv, u\rangle$, $s[p_i.inv \leftarrow p_i.inv']$ is a new state s' in which p_i's inventory is changed from $p_i.inv$ to $p_i.inv'$, i.e., $s' = \langle p_1.inv, \cdots, p_i.inv', \cdots, u\rangle$. Similarly, $s[u \leftarrow u']$ is a new state obtained by replacing mode u in s by u'.

In our agent-based framework, the behavior of an elements is defined by its internal merchandise transformation, message sending and processing, and delivery decisions.

Internal Merchandise Transformation Internal merchandise transformation depicts activities that produce and/or transport merchandise inside an element. For example, in case of a route, merchandise is moved from its receiving in-port to its out-port. In case of a manufacturing site, raw materials received at in-ports are transformed into finished products at out-ports.

Definition 5 (Merchandise Transformation Functions) *A merchandise transformation function of an agent a has form of $f_t : S \rightarrow S$, where S is the set of all the states of a. In addition, $f(\langle a.p_1.inv, a.p_2.inv, \cdots, a.p_l.inv, u\rangle) = \langle a.p_1.inv', a.p_2.inv', \cdots, a.p_l.inv', u'\rangle$ satisfies the following constraints,*

1. *$a.p^k.inv' - a.p^k.inv \leq 0$ if p^k is an out-port.*
2. *$a.p^k.inv' - a.p^k.inv \geq 0$ if p^k is an in-port.*

Definition 5 is general enough to describe a variety of internal merchandise transformation activities. For a manufacturing site a, $a.p^k.inv' - a.p^k.inv$ represents the amount of raw material consumed if $a.p^k$ is an in-port, and the amount of finished products produced if $a.p^k$ is an out-port. It should be noted that a merchandise transformation function does not specify types of raw materials and finished products. Raw materials and finished products can be different types of merchandise, and in reality, they often are.

For a route, $a.p^k.inv' - a.p^k.inv$ represents the amount of merchandise being transported. The transportation of merchandise does not have to be instantaneous. Transportation delay can be modeled in modes as well. This can be done by, for example, further decomposing modes and introducing finite queues. In such a case, the amount consumed at an in-port is not instantly transferred to out-ports; instead, it is pushed to a finite queue, and the amount transferred to out-ports is dequeue from the same queue. Since the size of an internal queue and its content are finite, the status of the queue can be encoded as a part of the modes.

Message Sending and Processing In our agent-based framework, elements are communicated through messages. Each message contains a receiving element and an action. An action is defined as a mode transition function of the receiving element. A receiving element processes a received message and changes the element's mode as the result.

Definition 6 (Actions and Messages) *An action α of an agent a is defined as a function $\alpha : U \rightarrow U$, where U is the set of a's modes. A message for agent a is a tuple $\langle a, \alpha \rangle$, where α is an action of a.*

Definition 7 (Message Sending Function) *Let $\mathcal{S}$ be a supply chain, a message sending function of an agent a in $\mathcal{S}$ is a function $f_m : S \rightarrow 2^M$, where S is the mode of a, and M is the set of messages generated in $\mathcal{S}$.*

Message sending function in Definition 7 describes how messages are generated by elements. Based on its mode, an element may send a set of messages and enter the next mode. Once generated, a message is routed to its destination. The set of messages sent by an element may also be empty, meaning that it does not send out any message at the current mode. Such a mode is called a *silent* mode.

Delivery Decision Delivery decisions made by an agent are defined by a delivery decision function. A delivery decision function decides how much merchandise an out-port shall deliver and how it shall be distributed to in-ports that are connected to the out-port. A delivery decision function makes its decision based on the current state. Definition 8 gives the formal definition of Delivery Decision Function. In other words, a delivery decision function has form of $f_d(s) = (\langle \delta_{p_1^- p_{11}^+}, \cdots, \delta_{p_1^- p_{1q_1}^+} \rangle, \cdots, \langle \delta_{p_m^- p_{m1}^+}, \cdots, \delta_{p_0^- p_{mq_m}^+} \rangle$, where $p_{i1}^+, \cdots, p_{iq_i}^+$ are in-ports connected to agent a's i-th out-port p_i^-, and $\langle \delta_{p_i^- p_{i1}^+}, \cdots, \delta_{p_0^- p_{iq_i}^+} \rangle$ is an outgoing delivery at p_i^-.

Definition 8 (Delivery Decision Function) *Let $\mathcal{S}$ be a supply chain and a be an element of $\mathcal{S}$ with m out-ports. A delivery decision function of agent a is a function $f_d : S \rightarrow (\hat{\delta}_{p_1^-} \times \cdots \times \hat{\delta}_{p_m^-})$, where S is the set of states of a, and $\hat{\delta}_{p_i^-}$ is an outgoing delivery at out-port p_i^-.*

3.3 Constraints

In our framework, an element may be imposed with a set of constraints. Constraints can be used to model physical or logical limitations imposed on an agent. For example, constraint can be used to model a facility a with limited inventory space V, in which case a constraint for a can be defined as $\sum_{p \in a.P^+ \cup a.P^-} p.inv \leq V$, where $a.P^+ \cup a.P^-$ is the set of all the a's ports.

Definition 9 (Constraints) *Let $\mathcal{S}$ be a supply chain. A constraint for element a is a predicate c over states of a, that is, $c : S \rightarrow \{true, false\}$, where S is the set of a's states. An execution ρ of $\mathcal{S}$ is invalid if an agent a can reach a state s such that $c(s)$ is false, where c is one of a's constraints.*

3.4 Special Cases

Our agent-based framework is general enough to define the structures and behaviors of a variety of supply-chain elements. On the top level, there are two categories of elements: facility nodes and routes. a route is a special kind of element that has precisely one in-port and one out-port. Its merchandise transformation function simply models transfer of merchandise from the in-port to the out-port with transportation delay.

Depending on the type of a supply chain, facility nodes can be further classified to several categories. For example, a retailer r has only in-ports. Its internal merchandise function models the consumption of merchandise at its demand rate. A warehouse w has both in- and out- ports. Its merchandise transformation function and delivery function shall satisfy the flow balance equation, that is, its inventory before an update plus incoming deliveries shall be the same as its inventory after the update minus outgoing deliveries.

An advantage of our framework is that an analyst can define his own type of elements for targeted supply chain operations. For example, in a traditional retail setting, a supplier has only out-ports, but for companies like Dell, a supplier's site also has manufacturing capability. In such a case, a supplier node has both in- and out-ports. Its merchandise function models how raw materials are consumed at in-ports and finish products are produced at out-ports.

4 Supply-chain Semantics and Simulation

We introduce semantics of a supply chain using its simulation semantics. That is, the semantics of a supply-chain model in our framework are defined by its simulation traces. Algorithm 1 defines our simulation algorithm.

Semantically a supply-chain model is a synchronous system extended with messages. Each iteration in Algorithm 1 simulates a clock update. A clock update is a basic time unit in supply-chain planning. Depending on planning horizon of underlying supply-chain operations, an update may represent a hour, a day, or a month etc. Each iteration starts with an internal merchandise transformation: lines 2-5 call the merchandise transformation function of each agent. As a result, an element enters its next state and sends out messages as defined in its message sending function. In general, whenever an element changes its state, its message sending function is called to check if the element needs to inform others of its change of state.

Algorithm 2 processes messages generated by elements. During each iteration it takes a message $\langle b, \beta \rangle$ from a message pool M and applies action β on receiver b. β may change the mode of b. b may generate messages at the new mode, or b's new mode can be a silent mode with no message being sent. Algorithm 2 uses a run-to-completion semantics, that is, it exits only after the message pool is empty.

Next, Algorithm 1 calls every element's delivery decision function to compute deliveries. To realize a delivery,

it subtracts inventory at an out-port and adds it to the connected in-ports. A delivery may also change the states of sending and receiving elements, which causes these agents to send out messages. Algorithm 2 is called afterward to process these messages.

Algorithm 1 simulate($\mathcal{S}$)

Require: a supply chain $\mathcal{S}$ with a set of agents A, where each agent $a \in A$ has a start state s_0^a.

1: **while** true **do**
2: **for all** $a \in A$ **do**
3: $s^a = f_t(s^a)$;
4: $M = M \cup f_m^a(s^a)$
5: **end for**
6: processMsg()
7: **for all** $a \in A$ **do**
8: *// Assume a has k out-ports* $p_1^- \cdots p_k^-$
9: $(\hat{\delta}_{p_1^-}, \cdots, \hat{\delta}_{p_k^-}) = f_d(s^a)$;
10: **for all** $p^- \in \{p_1^-, \cdots, p_k^-\}$ **do**
11: *// p^- is connected to* $p_1^+ \cdots p_q^+$
12: let $\langle \delta_{p^-p_1^+}, \cdots, \delta_{p^-p_q^+} \rangle = \hat{\delta}_{p^-}$
13: **for all** $p^+ \in \{p_1^+, \cdots, p_q^+\}$ **do**
14: // p^+ belongs to agent d.
15: $s^d = s^d[d.p^+.inv \leftarrow (d.p^+.inv + \delta_{p^-p^+})]$
16: $M = M \cup f_m^d(s^d)$
17: **end for**
18: $s^a = s^a[a.p^-.inv \leftarrow (a.p^-.inv - \delta_{p^-p_1^+} \cdots - \delta_{p^-p_q^+})]$
19: $M = M \cup f_m^a(s^a)$
20: **end for**
21: **end for**
22: processMsg()
23: **end while**

Algorithm 2 processMsg()

1: **while** $M \neq \emptyset$ **do**
2: $M = M - \{m\}$ *// Take a message m from M*
3: let $\langle b, \beta \rangle = m$
4: let $\langle b.p_1.inv, \cdots, b.p_n.inv, u^b \rangle = s^b$
5: $s^b = \langle b.p_1.inv, \cdots, b.p_n.inv, \beta(u^b) \rangle$
6: $M = M \cup f_m^b(s^b)$
7: **end while**

5 Implementation

We implement an initial prototype of the proposed agent-based formal framework in SIMRISK. SIMRISK is an integrated tool for supply chain modeling, simulation, and risk analysis. It implements a visual integrated development environment (IDE). Its IDE provides three different views of a supply chain: a hierarchical presentation of elements in the supply chain (tree view), a geographic view of elements (network view), and a property page for displaying attributes of a selected element (property view). SIMRISK IDE can visualize the on-the-fly status of a simulation, for example, it shows the animation of shipments during a simulation. Besides a normal simulation mode in which a user can start, pause, and stop a simulation, SIMRISK also provides a batch mode for numeral experiments. During the batch mode, all the non-essential status displays are disabled to reduce computational overhead. In the spirit of extensibility, the design of SIMRISK's graphic user interface also supports user-defined element types: SIMRISK provides a graphic handler to the user-defined type package, and the package can use the handler to display attributes of an element of user-defined type. Figure 1 shows a snapshot of SIMRISK's visual IDE. SIMRISK is written in Java.

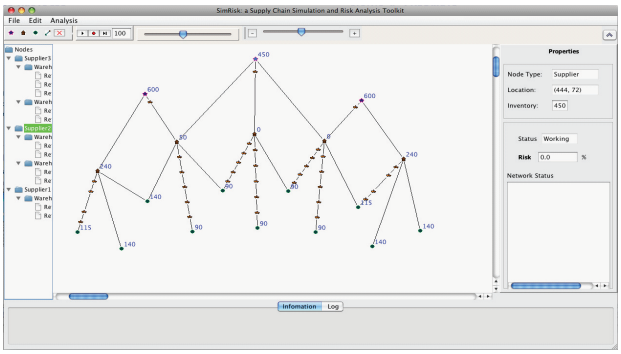

Figure 1. The Integrated Development Environment (IDE) of SIMRISK.

Figure 2 shows the architecture design of SIMRISK, presented as a UML component diagram. To achieve a higher extensibility, a key feature of SIMRISK is to separate the operational semantics of a supply chain from its topology. The requirement for defining the operational semantics of a supply chain is designated by an interface for *strategy*. A user can propose his own types of elements, as long as he supplies the necessary details as required by the strategy interface. The topology package stores the physical structure of a supply chain. They define, for example, the locations of nodes and how they are connected by routes. In other words, the topology package defines geographic locations of elements and the user-defined type package specifies their semantics as defined in Section 3. To test different supply chain policies on the same network structure, an analyst can switch between different strategies on-the-fly. SIMRISK implements an event-driven simulation engine as outlined in Algorithm 1.

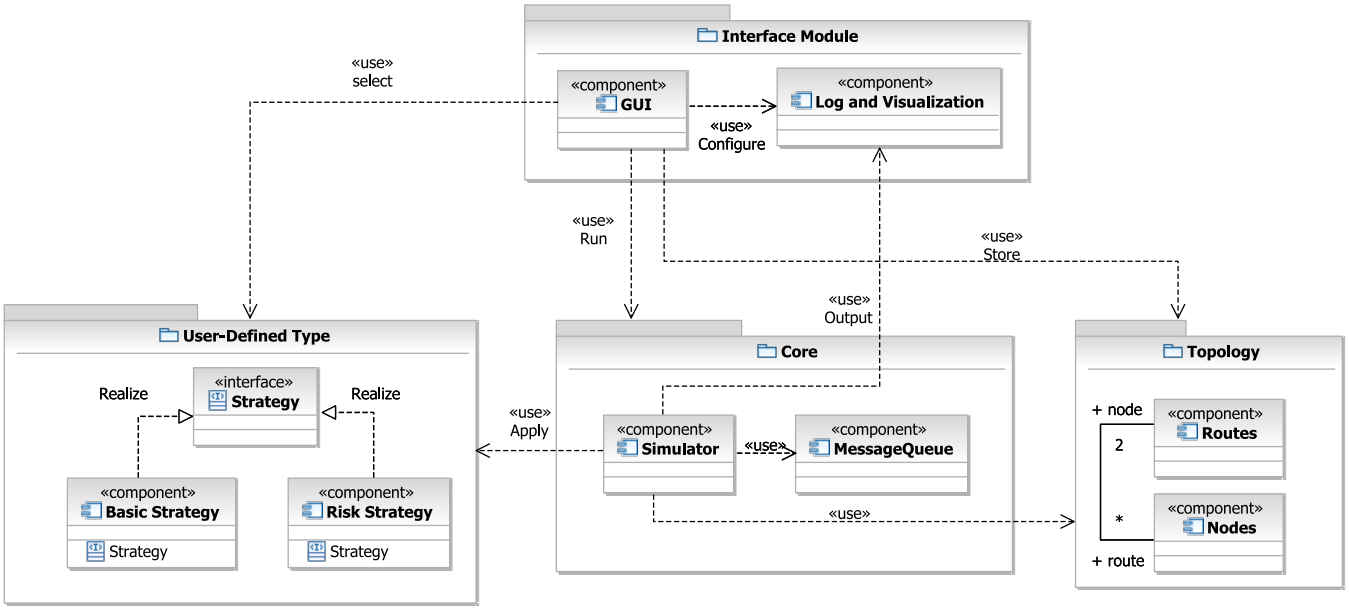

Figure 2. The architecture design of SIMRISK.

6 Conclusion and Future Works

We propose a novel agent-based formal framework for modeling and simulating supply chains. Our framework makes the following contributions to supply-chain modeling and analysis: first, our agent-based approach facilitates the study of complex system dynamics arising from interactions among different elements in a supply chain. Our event-driven simulation algorithm simulates system behavior by computing interactions among elements. Second, our framework is open and extensible. An analyst can customize it for targeted supply chain applications by introducing new types of element or redefining existing ones. Finally, we provide a formal definition for syntax and semantics of an element, and we introduce a simulation-based semantics for a supply chain model. The formalism we introduce helps precisely interpret simulation results and validate the tool's implementation. The formalism also facilitates automated formal analysis such as a model-checking-based risk analysis technique we introduced in [7].

There are several directions to extend this research. For instance, multi-core hardware could be an excellent platform for our agent-based framework, with elements running on different cores and communications expedited by shared memory and cache. we want to optimize our simulation algorithm and its implementation on multi-core hardware. Another possibility is to provide a tighter integration between simulation-based approach and formal analysis approach. Currently SIMRISK can translate a supply-chain model to an extended Markov decision process for formal analysis. In the future, we want to extend it to support user-defined element types. We also plan to implement more features of SIMRISK as described in this paper.

References

[1] R. Axelrod. *The Complexity of Cooperation:Agent-Based Models of Competition and Collaboration*. Princeton University Press, 1997.

[2] Costco Wholesale Corporation. Costco wholesale annual report 2007. Year ended Sept 2, 2007.

[3] J. Ferrer and J. Karlberg. Achieving high performance through effective global operations. Technical report, Accenture, 2006.

[4] J. Liu, W. Wang, Y. Chai, and Y. Liu. Easy-SC: a supply chain simulation tool. In *Proceedings of the 2004 Winter Simulation Conference*, volume 2, pages 1373–1378, Dec. 2004.

[5] M. D. Rossetti, M. Miman, V. Varghese, and Y. Xiang. An object-oriented framework for simulating multi-echelon inventory systems. In *Proceedings of the 38th conference on Winter simulation*, pages 1452–1461, 2006.

[6] J. Swaminathan, S. Smith, and N. Sadeh. Modeling supply chain dynamics: A multiagent approach. *Decision Sciences*, 29(3):607–632, Summer 1998.

[7] L. Tan and S. Xu. Model check stochastic supply chains. In *Proceedings of the IEEE 2008 international conference on Information Reuse and Integration*, pages 416–421. IEEE, 2008.

[8] W. Wang, J. Dong, H. Ding, C. Ren, M. Qiu, Y. Lee, and F. Cheng. An introduction to ibm general business simulation environment. In *Proceedings of the 2008 Winter Simulation Conference*, pages 2700–2707, Dec. 2008.