

# Multi-Agent Supply Chain Modeling – A Re-Usable Component-Based Framework

Ramakrishna Govindu

Department of Industrial & Manufacturing Engineering, Wayne State University

(Note: This is a Working Paper)

---

A re-usable component-based framework is proposed for modeling and simulation of multi-agent supply chain systems. The framework facilitates study of both intra- as well as inter-organizational issues simultaneously across a supply chain (SC) paying due attention to the issue of private as well as public information. It utilizes a pre-developed library of organizational primitives, supply chain primitives, agent behaviours, and policy objects in order to configure a SC of interest. The framework architecture integrates several software tools and technologies for implementing the framework. It utilizes *Java* and *Java Agent DEvelopment Framework (JADE)* for implementing various agent components, behaviours, and policy objects, *Protégé* and *Bean-generator* for SC domain ontology development, *PostgreSQL* for database back-end support, *Apache Ant* for Java code compilation, and *Eclipse* as an Integrated Development Environment (IDE) for facilitating integration and development. With the help of the libraries developed the user can quickly put together a multi-agent system (MAS) for a given supply chain and perform simulation and study coordination issues. The utility of the proposed framework is demonstrated through multi-agent simulation of Tamagotchi SC. The results of the study are presented and some of the research extensions being pursued are discussed.

Keywords: Supply Chain Modeling, Multi-Agent Simulation, Supply Chain Ontology, Java Agent DEvelopment Framework (JADE), Interaction Protocol

---

## 1. Introduction

Efficient management of Supply Chain (SC) is a key enabler providing competitive advantage to organizations. Individual firms in the market place are no longer competing as independent enterprises but rather as integral parts of SCs (Lambert et al., 1998; Min and Zhou, 2002). As more and more firms embrace supply chain management (SCM), efficient SCM is becoming a fundamental pre-requisite for survival rather than offering any significant competitive leverage. This in turn is compelling organizations to continuously innovate and implement creative practices both with and without the help of latest technologies in the business processes within a SC. Multi-Agent System (MAS) framework is one of such approaches becoming increasingly popular for SC modeling due to its ability to model the distributed and autonomous features of various entities comprising a SC in a natural and realistic way.

For over a decade, MASs have been offering the promise of becoming the next generation modeling paradigm for modeling complex systems such as supply chains. The process of MAS development still remains quite involved and extremely time consuming. Despite a few a real-

world commercial applications (Darley and Sanders, 2004; Himoff et al., 2005; Dorer and Calisti, 2005; van Aart et al., 2002), adoption of MASs mostly remains within the confines of the technology enthusiasts and visionaries needing traction to cross the chasm between visionaries and vanguard pragmatists for realizing mass market adoption (Wagner et al., 2005). Just a few of the reasons for the same include: lack of mature development methodologies, still evolving standards, and not-so layman-user-friendly “work-in-process” development tools and toolkits. This is more-so with respect to developing multi-agent supply chain systems (MASCS).

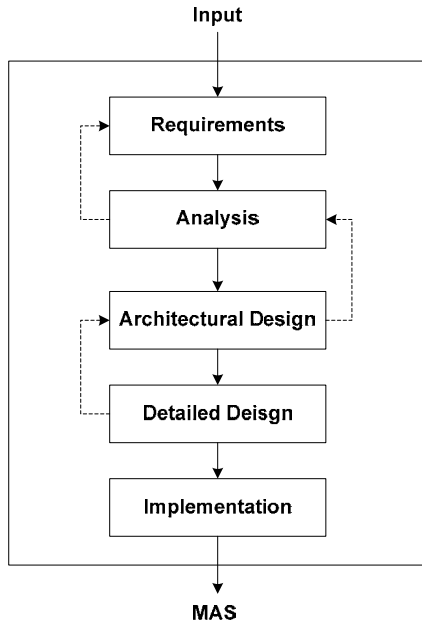


Figure 1: Phases in Multi-Agent System Development

Our research focuses on how to simplify the process of MASCS development. MAS development, or for that matter any software development process (as illustrated in fig.1), includes the phases of requirements analysis, conceptual analysis, system design (both architectural & detailed) and implementation phases applied sequentially and when-required iteratively. In our earlier research, we developed a methodological framework, MASCF (Multi-Agent Supply Chain Framework), for the analysis and design of MASCS by integrating the Gaia methodology and SCOR (Supply Chain Operations Reference-model). MASCF focused more on the conceptual analysis and the design phases of MASCS development.

This paper elaborates our research leading to the development of a re-usable component-based framework that focuses more on the design and implementation phases. We provide the details of our framework, discuss the development of component library, and show how the framework can be used to model and simulate Tamagotchi SC as described by Higuchi and Troutt (2004). In the following section, we review the literature pertinent to multi-agent supply chains. Section 3 introduces the re-usable component-based framework. We also present in detail the solution architecture and elaborate on how we conceive our framework to operate. In Section 4, discusses the implementation of the framework on the popular Tamagotchi case study for modeling the dynamics of its SC. Section 5, concludes and summarizes the research and presents some of the extensions.

## 2. Supply Chain Modeling & Multi-Agent Systems

This section reviews the literature pertaining to multi-agent based modeling and SCM, some of the agent-development toolkits, and wraps up with a discussion. Before doing so however, a brief note on multi-agent systems is provided.

### 2.1 Agent & Multi-Agent Systems

Literature presents numerous definitions for what an “agent” is. An agent is a computational entity such as a software program that perceives, acts upon its environment, and is autonomous in its behavior (Weiss, 1999). In a generic sense, an agent is an entity (either computer, or human) capable of carrying out goals and has two key properties: partial autonomy, and part of a community in which mutual influence occurs (Hayes, 1999). Weiss (1999) specifies two reasons for the popularity of multi-agent systems (systems with multiple interacting agents): (i) modern computing and information environments are distributed, large, open, and heterogeneous, and (ii) multi-agent systems have the capacity to play an important role in developing and analyzing models and theories of interconnectivity in human societies. In terms of the application potential, they are best suited and hold a great promise for a large spectrum of complex real-world systems, in particular, supply chains. Sycara (1998) identifies their characteristic features as: Each agent has incomplete information, capabilities, thus a limited viewpoint; There is no global system control; Data is decentralized; and Computation is asynchronous.

### 2.2 Multi-Agent Supply Chain Modeling & Management

Researchers have been exploring multi-agent systems in order to better model various supply chain problems and this sub-section presents some of the ongoing research. Fox et al. (2000) investigate the construction of intelligent agent-based software architecture for managing supply chains at the tactical and operational levels. They develop an “agent building shell” that provides generic, reusable, and guaranteed components and services to support cooperative work perturbed by stochastic events. An overview of MASCOT (multi-agent supply chain coordination tool) – a reconfigurable, multi-level, agent-based planning and scheduling architecture – is presented in Sadeh et al. (2001). The key architectural elements for real-time support in finite capacity scheduling and the development of new coordination protocols are discussed. Wagner et al. (2003) show how TAEMS agents, when equipped with coordination mechanisms, automate and manage a distributed dynamic supply chain. They demonstrate that agents increase flexibility, and enable the supply chain to be more responsive through producer/consumer negotiation and reasoning. Lin and Shaw (1998) propose a multi-agent information system (MAIS) approach for reengineering the order fulfillment process (OFP) in supply chain networks (SCN). A multi-agent simulation platform, SWARM, was enhanced to conduct simulated experiments to help in the reengineering efforts, and to identify and evaluate potential improvement strategies. Nissan (2001) presents intelligent supply chain agents that conduct business on behalf of product users, buyers and vendors. In the context of integration in a major enterprise, an agent-based supply chain process design, its structure, and the agent federation behavior (developed using Agent Development Environment (ADE) built on an expert system shell G2) are discussed.

In one of the earliest and most widely cited papers, Swaminathan et al. (1998) present a multi-agent framework for developing supply chain simulation models of appropriate fidelity with minimal time and effort. It involves composing models from a library of reusable, domain

specific, primitive software components representing supply chain agents, control element objects, and their interaction protocols. They discuss a cross-docking prototype, and compare multi-agent and conventional modeling approaches. Although the authors mention a full-scale application at IBM, they neither present the complete details/results of the system nor discussed the system details. A framework that integrates various elements of a supply chain represented in a unified, intelligent, and an object-oriented fashion is proposed by Julka et al. (2002a). It is designed to model, monitor, manage, and help analyze business policies within a supply chain. They demonstrate its application through a prototype decision support system, PRISMS (developed using ADE on G2), to study the effects of internal policies, exogenous events and plant modifications of a refinery. Jiao et al. (article in press) propose an agent-based multi-contract negotiation system for global supply chain manufacturing coordination. The system is implemented using Java Agent DEvelopment Framework (JADE) based on blackboard architecture at a leading mobile phone manufacturing company. A flexible agent system for supply chains that can adapt to transaction changes brought about by new products or trading partners was presented in Ahn et al. (2003). Their approach was demonstrated with the help of a PC supply chain application prototype. Gjerdrum et al. (2001) apply multi-agent modeling techniques to simulate and control a simple demand-driven supply chain network system. They utilize Java Agent Template Lite (JATLite) for modeling the multi-agent system and optimize its manufacturing component using general batch scheduling system (gBSS).

Although multi-agent technology has been in existence for sometime and gaining popularity, we hardly came across any literature that refers to industrial strength applications in general, and supply chain applications in particular. With the exception of a few papers (e.g., Fox et al., 2000; Swaminathan et al., 1998) that discussed the importance of development of generic components and reusability aspects, most of the applications seemed to offer a specific modeling solution to a particular problem. Articles like Sadeh et al. (2001) and Wagner et al. (2003) provide a discussion on the architectural issues, but the focus has been on specific aspects like coordination or real-time scheduling. From the literature, it appears that most of the applications are research oriented in nature. There were a few industrial prototypes that utilized a programming language (e.g., Java), commercial software (e.g., ADE, G2), or a freeware/open-source toolkits (e.g., Swarm, JATLite, JADE) for development. Most of the literature, having research oriented focus, does not seem to consider explicitly the system analysis and design (the most important aspects in the development of industrial strength applications) but seem to go directly from pre-stated requirements to implementation. We have not come across any literature that views MASCS development as a process. As a step in that direction, researchers have started developing generic methodologies, and toolkit-based implementation frameworks MAS development. In our research, we are focusing on how to simplify the development of MASs for SC modeling and decision support.

### **2.3 Agent & Multi-Agent Development Toolkits**

An interesting aspect is that while agent-based systems are becoming increasingly well understood, the development of multi-agent systems is not (Wooldridge and Jennings, 1999). They suggest devoting more effort to understand the pragmatics and the reality of MAS development. In some sense the problem of lack of commercial applications cannot be attributed to the lack of software tools for modeling MASs. In fact, there exist numerous toolkits, both commercial and open source, available in the market place. AgentLink (2002) report reviewed thirty-six software products. We argue that, the toolkit problem is more to do with the limited functionality, lack of

maturity, not-so-good documentation and methodology support, and either lack of or limited graphical-based user interface support for development to a large extent owing to still evolving standards. Some of the toolkits however are reaching a stage of maturity where they are getting used for commercial applications. A few popular toolkits are: Jack and Agent Builder among the commercial ones, JADE and Zeus among open-source software.

Developed by Telecom Italia and distributed under LGPL (Lesser General Public License Version 2), JADE is a software development framework aimed at developing MAS and applications. It simplifies the implementation of MAS through a middle-ware that complies with the FIPA (Foundation for Intelligent Physical Agents) specifications. JADE includes both an agent platform and a package to develop Java agents. It provides a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same operating system) and the configuration can be controlled via a remote GUI. JADE has been fully coded in Java language and is made of various Java packages, giving application programmers both ready-made pieces of functionality and abstract interfaces for custom, application dependent tasks.

There exist quite a few limitations with JADE, for example: it not-so easy to learn quickly and write programs unless some one is very good in object-oriented programming concepts especially Java, limited and high level documentation for a typical user/practitioner, unavailability of an official development methodology specific to the toolkit, almost non-existent user interface support for graphical description and model development. In spite of all these limitations, JADE is the most popular among the open-source toolkits and is finding its way into a few commercial applications as well. The popularity of JADE can be gazed by looking at numerous plug-ins and extensions being developed to improve upon the basic functionality offered by the JADE agents. It is precisely some such reasons led us to choose JADE platform for our research.

## **2.4 Discussion**

We strongly believe and argue in favor of treating the development of multi-agent systems as software engineering projects. Considering them as such, would lead to the development of systematic and structured processes that facilitate development of industrial strength applications. Wooldridge et al. (2000) argue that for agents to realize their potential as a software engineering paradigm, it is necessary that specific techniques tailored to them be developed. Clearly defining, simplifying, and speeding up the process of development are essential for the widespread adoption of multi-agent paradigm by the industry, be it for supply chain modeling or otherwise. In this research, we are attempting to do address some of such issues. Our framework attempts to define and develop library of components that can be used to configure to model a supply chain of interest. In the following sections, we offer more details of our framework, discuss its implementation on Tamagotchi SC.

## **3. Re-Usable Component-Based Framework**

The framework is designed to facilitate analysis and study of both intra- as well as inter-organizational issues together across a supply chain (SC) paying due attention to the issue of private as well as public information. It utilizes a pre-developed library of organizational primitives, supply chain primitives, agent behaviours, and policy objects in order to configure a SC of inter-

est. Our framework is best explained with the help of a graphic (see fig. 2). It consists of two layers:

- An infrastructure layer that consists of several software products required for developing, modeling and running a MASCS, and
- A modeling layer consisting of different components of a pre-defined library using which MAS of any SC would be configured.

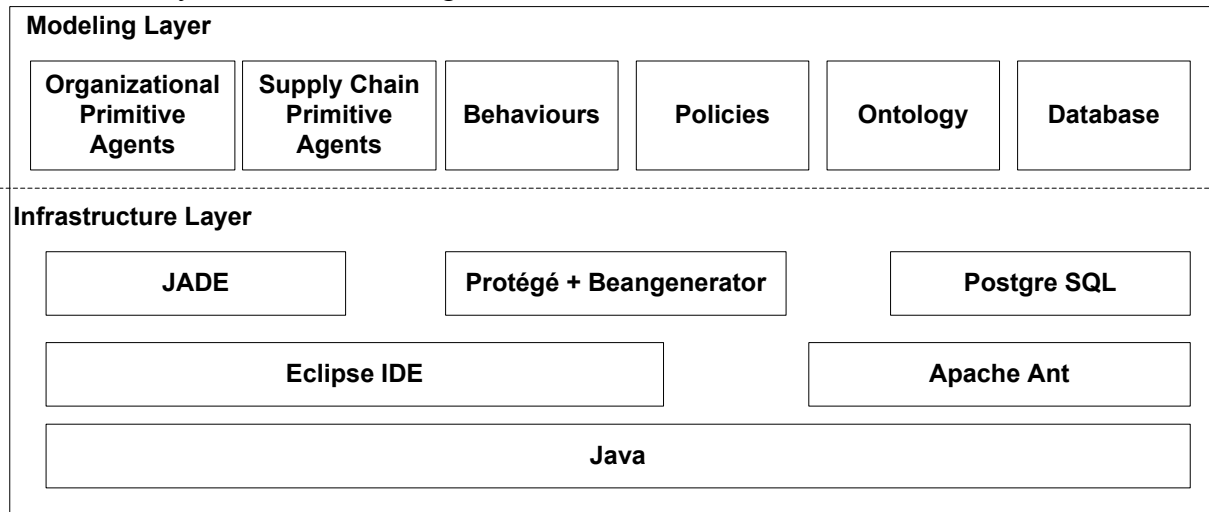


Figure 2: A Re-Usable Component-Based Framework

### 3.1 Infrastructure Layer

As indicated in the fig. 1, the basic foundation on which the system runs is the Sun Microsystem's Java Platform. As we had to integrate multiple toolkits and softwares for making the solution work, we chose the Eclipse platform to be the IDE for the solution framework. Eclipse is an extensible, open source java-based IDE donated by IBM, brings together all of the tools a developer needs to be successful at application development. It uses Apache's Ant in order to compile the Java code written for the system. As already indicated earlier, the agent-development toolkit being adopted for implementing the framework is JADE which is completely implemented in Java. JADE provides, pre-developed standard templates, objects, methods that can be extended to develop the code for any MAS. The fundamental characteristic of MAS is that individual agents communicate and interact. JADE provides three ways to implement communication between agents:

- Using strings to represent the message content
- Using Java serialized objects
- Defining ontology objects to be transferred as an extension of predefined classes that JADE can encode and decode messages in a standard FIPA format

This framework adopts the ontology approach, and in order to implement the same, the popular Ontology development toolkit Protégé developed by Stanford's Medical Informatics is selected along with the Beangenerator plug-in developed by Acklin BV. Protégé+Beangenerator tool allows the user to define domain ontology using graphical user interface of Protégé and the JADE template of the Beangenerator. Then using Beangenerator the user can automatically generate ontology source code in JADE acceptable format. At runtime, JADE agents take the help of the developed ontology objects for communication and coordination. Using this approach greatly

reduces the burden on the developer to carefully program ontology in JADE acceptable format. The final part in the infrastructure layer consists of database software, PostgreSQL. PostgreSQL is one of the world's most advanced open source databases. It is integrated into the infrastructure layer with the help of JADE (Java Database Connectivity). Using the database component helps persist the data associated with important model variables at runtime and greatly relieves the developer of the burden of storing the data in predefined output file formats for later use and analysis. Once the data is stored in the database, database tables can be accessed and the data imported into excel files for further analysis. At a later time, we will extend our framework to incorporate data analysis features into agents to eliminate this burden on the model builders/users.

## 3.2 Modeling Layer

The modeling layer consists mainly of the component categories of the Library developed. The framework utilizes several software tools and technologies for implementing the various component categories. It utilizes *Java* and *Java Agent DEvelopment Framework (JADE)* for implementing various agent components, behaviours, and policy objects, *Protégé* and *Beangenerator* for SC domain ontology development, *PostgreSQL* for database back-end support, *Apache Ant* for code compilation, and *Eclipse* as an Integrated Development Environment (IDE) for facilitating integration and development. With the help of the libraries developed the user can quickly put together a multi-agent system (MAS) for a given supply chain and perform simulation and study coordination issues.

### 3.2.1 Agent Components

Our framework conceives a SC system to be consisting of entities at multiple and hierarchical levels. It is natural to conceive any SC network to be consisting of multiple SC entities/organizations. There would be Retailers, Distributors, Manufacturers, Suppliers, and Raw Material Suppliers in a SC network. All of the above entities are obvious in and shall remain the same irrespective of model resolution – detailed or concise. The only exception to the above logic occurs when a particular SC network having a Market entity representing all the possible customers for a product/s. This type of representation would be sufficient if one were conducting a study at an aggregated level rather than detailed. For most of the systems involving Strategic and Tactical Level studies such a representation would be sufficient and in some cases most desirable. However, Operational level and Real-time studies would require systems with much more detailed representation in which case, the ultimate Customers (multiple) will have to be represented explicitly and aggregated representation would be insufficient and may even be undesirable. Fig. 3 shows how we envisage the framework gets used.

In the real-world, SC entity organizations usually would consist of multiple functional organizations within. For example: A retailer will have a sales function, an inventory function, and a procurement function, ... etc. Likewise, a Manufacturer may include in addition to the above, a production function, facility function, ... etc. Our library is designed to consist Organizational Primitive Agents, for example: Sales Agent, Inventory Agent, Production Agent, Procurement Agent, Facility Agent, ... etc. In addition, the library also includes SC Primitive Agents, for example: Market Agent, Retailer Agent, Manufacturer Agent, Supplier Agent, ... etc. Our SC Primitive Agents are enabled through configuration the ability of generating the right resolution of the model – either concise or detailed. If the concise option is chosen at the time of configuration then only the SC primitive agent would be instantiated. If, instead, the detailed option is chosen then not only the SC primitive agent gets instantiated, but also all the organiza-

tional primitives under it. In addition, through configuration SC primitive agent will be provided with an ability to select only a subset of organizational primitives for instantiation. This option would be quite useful if for some reason let us say a manufacturer wants to focus on the downstream operations but not upstream, then perhaps instantiation of procurement organizational primitive agent may not be necessary and therefore should not be instantiated. Depending on the need configuration file can be initialized.

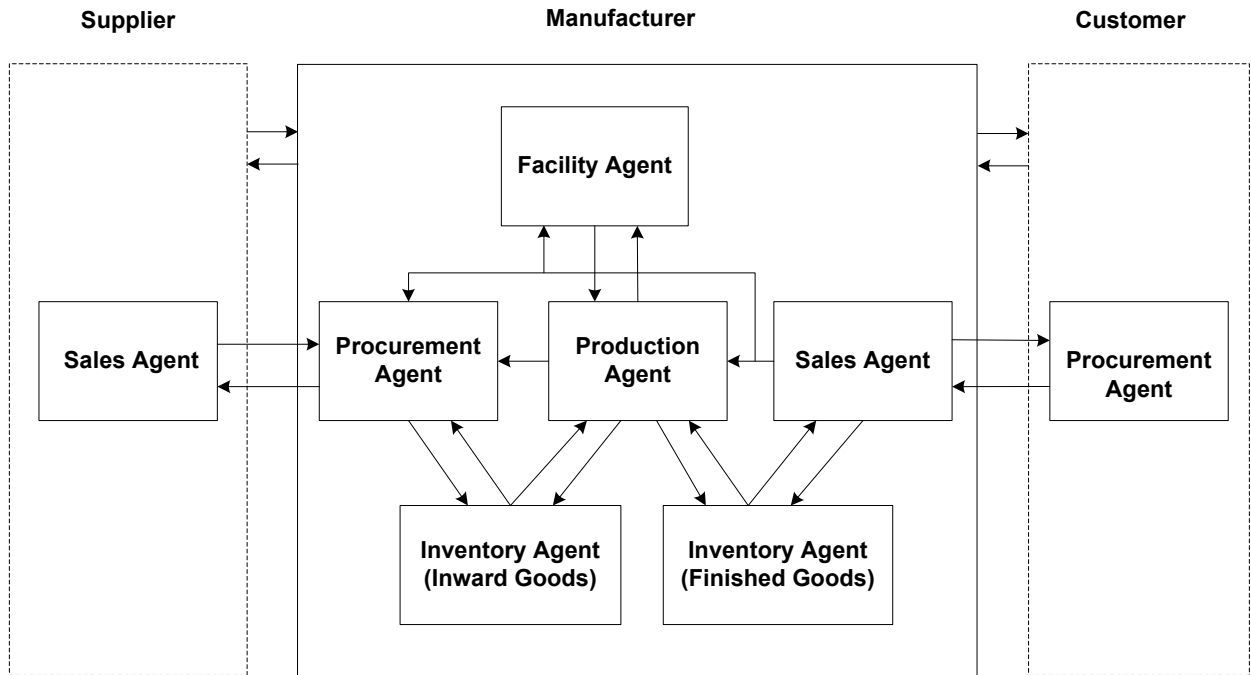


Figure 3: Modeling intra- as well as inter-organizational dynamics using the Framework

Having discussed the basic features of the agent components, we now elaborate on the reusability aspect of the framework. The framework by design conceives the organizational primitive agent components being used in multiple SC primitives. For example: an organizational primitive “Sales Agent” may be used in “Retailer”, “Manufacturer”, and “Supplier” SC primitives. This makes sense even from the real-world situation point of view. The basic functionality of the sales agent usually remains more or less the same irrespective of to which SC primitive entity it belongs to. For example: a Sales Agent usually receives customers’ orders and fills them based on certain pre-defined organizational policy, forecasts the likely demand for a future time period and informs other concerned organizational agents about it, and so on and so forth. The agents with whom this particular sales agent corresponds could be slightly different depending on the SC entity to which it belongs to. Such issues under this framework will be taken care of by the acquaintances list but doesn’t make a Sales Agent under one SC entity drastically different from another. Likewise, when we define a SC primitive agent let us say a Retailer, usually one could see that at the conceptual level different retailers could be the same and hence can be represented using the same SC entity called retailer. Of course, the internal working processes could be slightly different but, organizational entities would usually be the same. Even if they differ, the framework provides for adjustments through configuration at the time of model initialization.

The agent components both organizational primitive agents and SC primitive agents consist only of agent shells with only communication ability. The agents have the option of one of the

two communication architectures: either peer-to-peer or SC entity-to-entity. It should be obvious that if a detailed model is being configured, the communication between two organizational primitive agents within the same SC entity will be peer-to-peer. However, communication between two organizational primitive agents of different SC entities will have both the options with one of them to be chosen at the time of configuration. If SC entity-to-entity option is chosen, then the message from one organizational primitive agent to another organizational primitive agent inside a different SC entity will be routed through their corresponding SC primitive agents. The nature of communication between two SC primitive agents shall remain peer-to-peer irrespective of the model being concise or detailed. In order to take care of constraints in communication, rather to prevent it, the framework includes the generation of an acquaintance list from the model configuration file at the time of model instantiation. Having discussed the Agent components, our discussion now shifts to focus on the other components behaviours, policies, and SC domain ontology in the following sub-sections.

### **3.2.2 Behaviours**

JADE, like several other MAS toolkits, conceives agents to offer some services and these services are implemented by certain “behaviours”. JADE for example provides a diverse range of behaviour templates ranging from simple to composite to Finite-State-Machine (FSM). A research group from University of California at Santa Cruz developed even more powerful Hierarchical State Machine (HSM) as an add-on to the basic JADE behaviour functionality. Agents’ capabilities are defined by extending some of such behaviour templates available in JADE. The framework provides for defining re-usable behaviours that can be used across different agents. To give an example, an inventory agent could include the behaviour “inform inventory level”. The steps involved with informing inventory level behaviour remains usually the same. Some of the things that could possibly change from one inventory agent to another are: how often inventory level is informed, to whom all it is informed. It is pretty obvious that such information can be parameterized and can be defined at the time of instantiation. Another example could be the “inform demand forecast” behaviour of a sales agent. It is obvious from the above examples that most of the behaviour objects likewise can perhaps be defined in advance leading to reusability.

### **3.2.3 Policies**

Similar logic can be applied to policy objects. While in operation real-world supply chains will be utilizing certain pre-set organizational while making decisions. A few examples include: forecasting policy, order filling policy, procurement policy, production policy, capacity planning policy, facility expansion policy ... etc. It is obvious that different SC entities would be adopting different policies. However, from time to time organizations would be changing or carrying out various studies to understand the impact of a change policy ... etc. The execution logic for some of these policies remains the same. For example: exponential smoothing based forecasting policy logic remains same wherever it is used. The only likely changes are the parametric values and the input data. This allows for developing policy objects that can be reused across organizations. The only factor is that the potential list of such policy objects would be large.

### **3.2.4 Ontology**

The next aspect to be considered in our framework is the development of SC Domain Ontology. There exist numerous definitions for the term “Ontology” is. In the computer world “Ontology” means machine readable vocabulary that is specified with enough precision to allow differing terms to be precisely related. It includes computer-usable definitions of basic concepts and property slots of the domain of interest, the relationships among the various slots and concepts. On-

tology holds an enormous potential in making software more efficient, adaptive, and intelligent by facilitating sharing of common understanding, separation of domain and operational knowledge and reuse of domain knowledge. Ontology is being projected as one of the areas expected to bring the next breakthrough in software development. In our framework we are utilizing ontology mainly for agent communications.

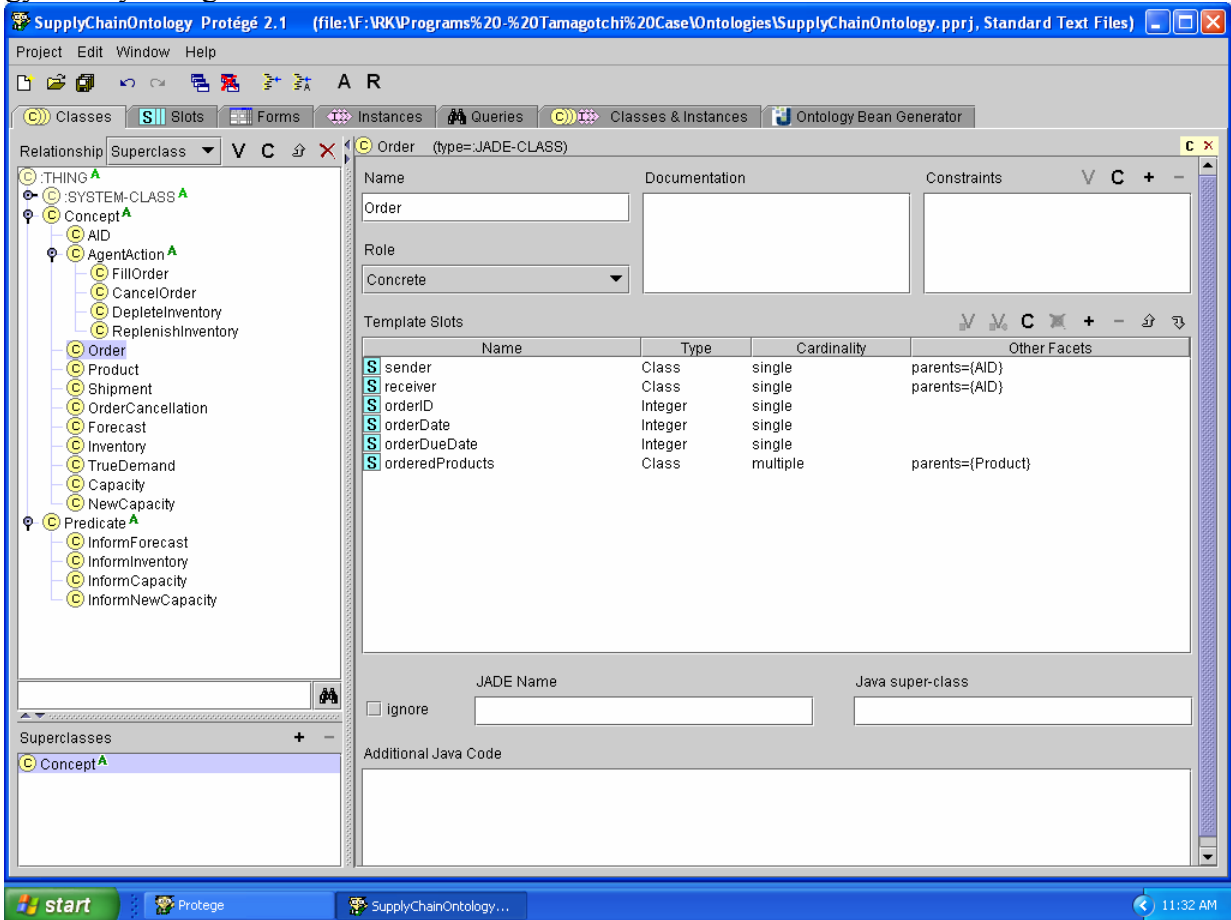


Figure 4: A partial view of the Supply Domain Ontology

The process of domain ontology generation is quite creative, involved and there exists no one single method of specifying the ontology. Developing SC domain ontology promotes structuring and modeling of information used in SC modeling and promotes reuse. Fig. 4 illustrates, partial details of the SC domain ontology developed for the implementation using the framework.

## 4. Case Study

This section illustrates how the framework and the developed libraries can be applied in practice. We demonstrate how the various components of the library can be used to configure the Tamagotchi SC influenced by real-world issues as outlined in Higuchi and Troutt (2004). Also presented are a few results of the multi-agent simulation runs. Before doing so however, a brief introduction to the case is provided.

## 4.1 Introduction to Tamagotchi Case Study

Tamagotchi was the first of the virtual pet games, introduced to the market in 1996 by Bandai Co., the Japanese toy manufacturer. Bandai estimated that this toy had the potential to be a big hit; however, they could not accurately forecast the demand and its shift. Although no advertisements were placed in the mass media, the effect of word of mouth was much stronger than expected with the demand boom outpacing the ability to meet the demand. Finally, they had to expand their manufacturing facilities to produce 2–3 million units per month despite the high risk of overstocking and excess capacity involved. After expanding their manufacturing capability, it met with a sharp decline of demand leading to huge unsold inventory that resulted in an after-tax losses of 16 billion yen (US\$123 million at US\$1 = 130 yen) in fiscal 1998. To illustrate what happened to Bandai and to demonstrate how they might have avoided these tremendously unfortunate effects, Higuchi and Troutt (2004) built a simulation model using system dynamics approach. The objective of their study was to show that such a modeling approach would be helpful to decision makers and planners faced with similar short-life-cycle product introductions. This case provides a good example illustrating the problems that can arise from the interactions between capricious demand, boom or bust, and capacity decisions in the very short product life cycle setting. We utilized the information provided in Higuchi and Troutt (2004) to setup our MASCS. The three stages of Tamagotchi system dynamics model were identified as the three SC stages – the manufacturer, the retailer, and the market. The entire system dynamics model logic was split among these three stages. The logic was then projected into process level logic for the agents using our practical experience in industry. In the absence of the actual requirements, the derived information was assumed to be the system requirements and was used in the development of the MAS. In addition, instead of considering the continuous time as in the system dynamics model, we assume the time to be discrete of weekly intervals and run the multi-agent model for a predetermined number of weeks.

## 4.2 Designing & Configuring the Tamagotchi SC using the Framework

The first step in the process is to conceive the Tamagotchi SC system as a collection of SC entities. From the details of the model provided, it is appropriate to consider three SC entities corresponding to the three stages of the SC – the market, the retailer, and the manufacturer. The framework offers the flexibility to implement the system either by a single decision maker, or multiple decision makers, or even by multiple organizations if they have an understanding to do so. A decision on the above would determine if the system will be implemented on a single machine or a single platform across multiple machines or on multiple platforms spread across multiple machines connected over some network. For purpose of testing the framework, we have implemented the system on one single machine. However, since JADE is proven to offer support for MAS implementation spread across multiple machines and platforms, one could take up such implementation without any major issues. Higuchi and Troutt (2004) considered the market level demand generation to be an aggregated level activity since; the purpose of the supply chain model is to assist in planning and not real-time operations. Our model also treats the market level demand generation at an aggregated level, and considers it to be the responsibility of market-level sub-organization. It is also possible to consider multiple markets, multiple retailers, and multiple manufacturers. However, in order to keep the model simple following the aggregated approach of Higuchi and Troutt (2004), our model also considers only one organization at each level.

### 4.3 Multi-Agent Simulation Results of Tamagotchi SC

The simulation model is run based on the information provided in Higuchi and Troutt (2004). Fig. 5 is a screen shot of the JADE model of the Tamagotchi SC system under execution. It shows various messages relating to interaction protocols getting exchanged between the agents comprising the system.

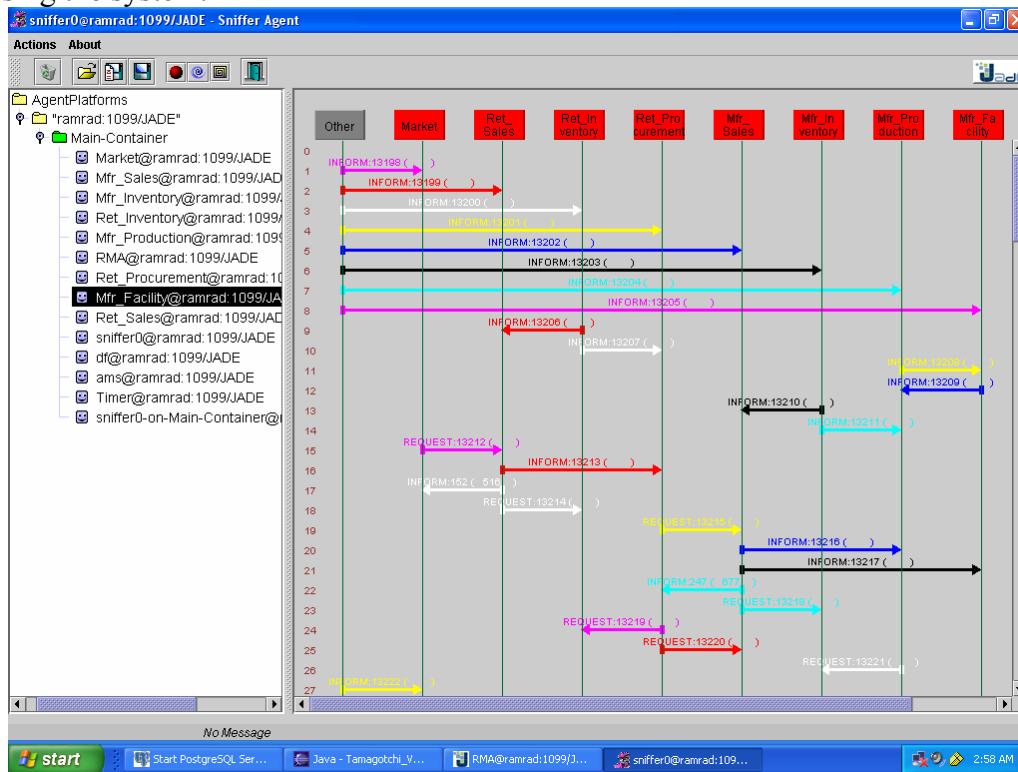


Figure 5: JADE Screenshot showing Message Exchanges in a Multi-Agent Simulation Run

Fig. 6 presents some of the results of the multi-agent simulation exercise. Fig. 6(a) indicates boom and bust phenomenon that relates total demand and manufacturing capacity. The capacity had its peak enhanced by the over estimate of the demand. As the capacity increased to large levels, the demand dropped steeply leading to losses due to over investment. Fig. 6(b) shows the inventory levels at the manufacturer and the retailer. As the manufacturer builds more and more capacity after reaching a certain peak, the demand vanishes and the manufacturer would be left with too much of inventory largely due to excess capacity additions driven by boom and bust. Impact of multiple diffusion speeds was studied and the results are plotted in Fig. 6(c). The plot indicates the total demand and the periodical demand for three diffusion speeds. It is clear from the figure that as the product diffuses at a much faster rate into the market, the risks associated with bullwhip and boom and bust also will be much higher. Therefore, it is extremely important that short life cycle product planning has to consider a lot of alternatives and options unlike the traditional stable demand products. Fig. 6(d) shows the benefit to the manufacturer in terms of reduction in inventory when the retailer shares the sales information. Incidentally, it can also be seen that the results generated out of the multi-agent model are comparable to those of Higuchi and Troutt (2004).

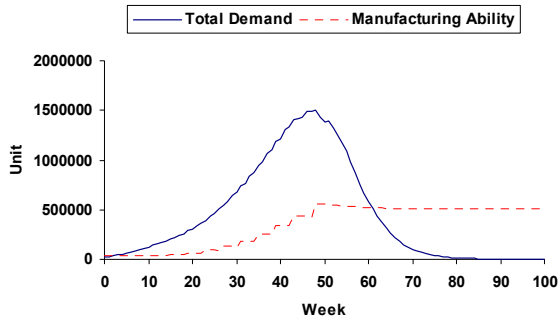


Fig. 6(a) Total Demand vs. Mfg. Ability

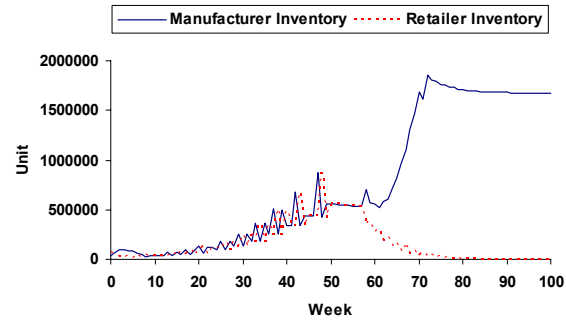


Fig. 6(b) Inventory – Manufacturer vs. Retailer

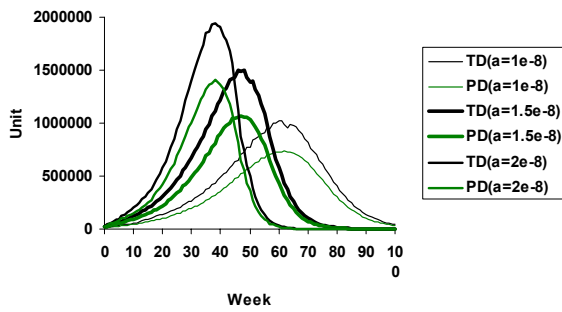


Fig. 6(c) Impact of Diffusion Speed

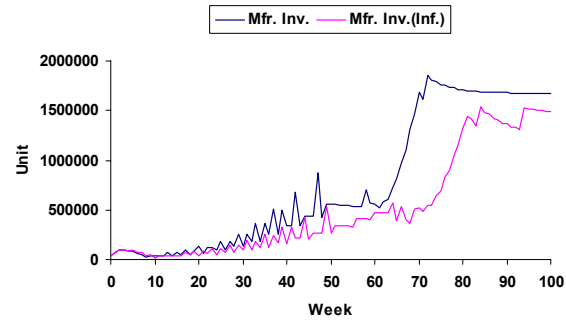


Fig. 6(d) Impact of Information Sharing on Total Demand and Periodical Demand

Figure 6: Simulation Results of Tamagotchi supply chain

## 5. Conclusions & Summary

Multi-agent systems introduce a new paradigm for modeling complex systems like supply chains. Although they are gaining popularity, their implementation is confined to academic research to a large extent. This is true in particular for SC systems. This paper argued that in order for the real-world industrial strength multi-agent applications to proliferate, it is extremely important to simplify the development process of multi-agent systems. Although numerous toolkits for developing MASs are available, the support they offer in terms of functionality, methodology and documentation, easiness of modeling varies to a great extent. Our research focuses on simplifying the development of MASs for SC modeling. This paper captured the details of a reusable component-based framework developed in our research. Also discussed are the details as to how we conceive the framework to be used in practice. In order to test the framework, we populated the components the library. These components were in turn used to model the Tamagotchi SC. Multi-Agent Simulation runs of the model were carried out brief results ate presented. Having proved its utility, we are confident that the framework would serve its intended purpose well.

We end the discussion with likely extensions of this research work. The implementation of the framework is being taken up in phases. Having successfully completed the first phase development and testing, in the next phase we are extending the ability of the generic components especially relating SC primitive agents. Once this is done, we are going test the ability of library

components to quickly configure and simulate SC network of reasonable size at different levels of resolution. In addition, further population of the libraries is to be taken up to incorporate additional SC entities (e.g.: logistics service providers, distributors), organizational primitive, behaviours, policies, and further expansion of SC Domain Ontology. Having established the framework and working infrastructure, further research is being planned to simplify the model building process incorporating graphical model descriptors and code-generation options. In addition, making the agents more powerful by incorporating intelligence and reasoning logic is also being planned. We argue that such options would further speed up the development and offer more potential for MASCS applications.

## Acknowledgments

I am extremely grateful to my advisor, Dr. Ratna B. Chinnam, Associate Professor in the department for his valuable guidance, encouraging support throughout the process of my doctoral research of which this particular work is one of the outcomes. It would have been impossible for me to carry out the research but for the financial assistance provided by the department through research and teaching assistantships. I am indebted to all those faculty members of the department who made the decision to support me financially at different points of time.

## References

- AgentLink. 2002. Review of Software Products for Multi-Agent Systems. Report.
- Ahn, H. J., H. Lee & S.J. Park. (2003). A flexible agent system for change adaptation in supply chains. *Expert Systems with Applications*, 25, 603-618.
- Darley, V., D. Sanders. 2004. An agent-based model of a corrugated-box factory: the trade-off between finished goods stock and on-time-in-full delivery. In H. Coelho and B. Espinasse, editors, Proceedings of the Fifth Workshop on Agent-Based Simulation, 2004.
- Dorer, K., M. Calisti. 2005. An adaptive solution to dynamic transport optimization. In M. Pechoucek, D. Steiner, and S. Thompson, editors, Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems: Industry Track, pages 45–51. ACM Press, 2005.
- Fox, M.S., M. Barbuceanu & R. Teigen. (2000). Agent-Oriented supply chain management. *The International Journal of Flexible Manufacturing Systems*, 12, 165-188.
- Gjerdrum, J., N. Shah & L.G. Papageorgiou. (2001). A combined optimization and agent-based approach to supply chain modeling and performance assessment. *Production Planning & Control*, 12(1), 81-88.
- Hayes, C.C. (1999). Agents in a nutshell – A very brief introduction. *IEEE Transactions on knowledge and Data Engineering*, 11(1), 127-132.
- Higuchi, T., M. D. Troutt. 2004. Dynamic simulation of supply chain for a short life cycle product - Lessons from the Tamagotchi case. *Computers & Operations Research*, 31, 1097-1114.
- Himoff, J., P. Skobelev, M. Wooldridge. 2005. MAGENTA technology: Multi-agent systems for industrial logistics. In M. Pechoucek, D. Steiner, and S. Thompson, editors, Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems: Industry Track, pages 60–66. ACM Press, 2005.

- Jiao, J. (Roger), X. You & A. Kumar. (Article in press). An agent-based framework for collaborative negotiation in the global manufacturing supply chain network. *Robotics and Computer-Integrated Manufacturing*.
- Julka, N., R. Srinivasan & I. Karimi. (2002a). Agent-based supply chain management – 1: framework, *Computers and Chemical Engineering*, 26, 1755-1769.
- Lambert, D.M., M.C. Cooper & J.D. Pagh. 1998. Supply chain management: Implementation issues and research opportunities. *The International Journal of Logistics Management*, 9(2), 1-19.
- Lin, F. & M.J. Shaw. (1998). Reengineering the Order Fulfillment Process in Supply Chain Networks. *International Journal of Flexible Manufacturing Systems*, 10(3), 197-229.
- Min, H. & G. Zhou. 2002. Supply chain modeling: past, present and future. *Computers & Industrial Engineering*, 43(1-2), 231-249.
- Nissan, M.E. (2001). Agent-based supply chain integration. *Information Technology and Management*, 2, 289-312.
- Sadeh, N.M., D.W. Hildum, D. Kjenstad & A. Tseng. (2001). MASCOT: an agent-based architecture for dynamic supply chain creation and coordination in the internet economy. *Production Planning & Control*, 12(3), 212-223.
- Swaminathan, J.M., S.F. Smith & N.M. Sadeh. (1998). Modeling supply chain dynamics: A multiagent approach. *Decision Sciences*, 29(3), 607-632.
- Sycara, K. P. (1998). Multiagent systems. *AI Magazine*, 19(2), 79-92.
- van Aart, C. J., K. van Marcke, R. F. Pels, J. L. F. C. Smulders. 2002. International insurance traffic with software agents. In F. van Harmelen, editor, Proceedings of the Fifteenth European Conference on Artificial Intelligence. IOS Press.
- Wagner, T., V. Guralnik & J. Phelps. (2003). TAEMS agents: enabling dynamic distributed supply chain management. *Electronic Commerce Research and Applications*, 2, 114-132.
- Wagner, T., L. Gasser, M. Luck, J. Odell, T. Carrico. 2005. Impact for agents. In M. Pechoucek, D. Steiner, and S. Thompson, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems: Industry Track*, pages 93–99. ACM Press.
- Weiss, G., (Ed.). (1999). *Multiagent systems: A modern approach to distributed artificial intelligence*. The MIT Press.
- Wooldridge, M.J. & N.R. Jennings. (1999). Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing*, May/June 1999, 20-27.
- Wooldridge, M., N.R. Jennings & D. Kinny. (2000). The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3, 285-312.

