MULTI-AGENT SYSTEMS FOR SUPPLY CHAIN MODELING: METHODOLOGICAL FRAMEWORKS

by

RAMAKRISHNA GOVINDU

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2006

MAJOR: INDUSTRIAL ENGINEERING

Approved by:

_____

Advisor Date

_____

_____

_____

_____

DEDICATION

DEDICATED TO:

My Parents,

Sri. G.V.K.S. Sarma & Smt. M. Ramalakshmi

and

everyone who influenced my life in a major way

ACKNOWLEDGMENTS

I express my sincere gratitude to my advisor, Dr. Ratna Babu Chinnam for his outstanding advise, inspiration, constant encouragement, and relentless support in this research. But for his able guidance, throughout the entire process, this dissertation would not have seen the light. I am indebted to my dissertation advisory committee members, Dr. Kenneth R. Chelst, Dr. Leslie F. Monplaisir, Dr. George C. Jackson, and Dr. Sanjay E. Ramaswamy for their critical review, valuable suggestions, and able counseling. Their outstanding navigation along with my advisor helped improve the quality of this dissertation research substantially. I would like to thank the Engineering Management Masters Program (EMMP), all its faculty, and directors (current and previous), and the department for having supported me with financial assistance throughout this entire journey. I greatly appreciate all the help and guidance I received from Dr. Francis E. Plonka in particular during the initial stages of my doctoral studies.

Thanks are due to my cousin Mr. Vishist Mandapaka for assuming the role of a '*consultant*' and helping me sort out software technology and programming related glitches. Thanks are also due to my wife Radhika M. Manda for not only her caring support but also programming/other related assistance. Without their help and support, it would have been impossible to validate the frameworks, and run simulations successfully in this research. I am grateful to my parents; my brother and sister and their families; and my brother-in-law for having stood by me through thick and thin. I appreciate my daughter's assumed role in making this otherwise arduous journey much more pleasant. I also take this opportunity

to thank all my relatives and well wishers who had contributed towards my development in a major way in some form or other.

Thanks are due to my friends, in particular Dr. Rajesh Jugulum. For he not only being instrumental in helping me to come to the USA but also getting me an assistantship. Thanks are also due to Mr. Nitin Garg for all his help in numerous ways on several issues from time to time. I also appreciate Dr. B. K. Rai, Dr. Bimal Nepal, Mr. Pundarikaksha Baruah, and Mr. Gangaraju Vanteddu, for their help, support, and making my stay quite pleasant.

I appreciate the help extended to me by the support and secretarial staff in the department on all the administrative/official matters. I would like to point out and thank in particular Ms. Rita Coyne, Mr. Terry Fisher, and Dr. Kirk Cornell for all the help they extended to me.

TABLE OF CONTENTS

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

CHAPTER 1

INTRODUCTION & PROBLEM STATEMENT

Efficient and innovative supply chain management (SCM) practices have become essential enablers providing competitive advantage to enterprises in the market place. Individual firms today no longer compete as independent enterprises but rather as integral parts of supply chains (SCs) (Lambert et al., 1998; Min and Zhou, 2002). It follows that the ultimate success of an enterprise is not derived independent of, but coupled with the destiny of its SC. Hence, the survival of an enterprise is dependent on its ability to efficiently integrate and coordinate the intricate network of business processes among its SC partners (Drucker, 1998; Lambert and Cooper, 2000). The sheer scope of integrated business processes from supplier's supplier to customer's customer (Gartner-Dataquest Report, 2005; Chopra and Meindl, 2006; Christophor, 1998) is what makes efficient management of SC a complex task, but potentially highly rewarding at the same time. As more and more firms embrace efficient SCM practices, such initiatives are becoming a fundamental pre-requisite for survival in lieu of offering any significant competitive leverage. This, in turn, is compelling enterprises to continuously innovate and make better decisions in their SC business processes for achieving substantial benefits. One of the approaches they resort to is to develop and utilize better, powerful, and much more realistic models for SC decision making.

Traditional SC modeling practices involve application of: optimization models (e.g., Arntzen et al., 1995; Goetschalckx et al., 2002; Lee and Billington,

1995; Beamon, 1998), mathematical models (e.g., Cachon, 2003; Cachon and Fisher, 2000; Anupindi and Bassok, 1999; Lee and Whang, 1999), simulation (e.g., Holweg and Bicheno, 2002; Bhaskaran, 1998; Terzi and Cavalieri, 2004; Chan et al., 2002; Petrovic, 2001), simulation-optimization (Padmos et al., 1999), system dynamics (e.g., Higuchi and Troutt, 2004; Sterman, 1989; Angerhofer and Angelides, 2000), and others. Traditional modeling techniques are quite suitable for modeling SC problems within a single enterprise. However, they are constrained by certain major limitations. For example, although simulation is the most pragmatic modeling approach for detailed analysis and evaluation of SCM alternatives (Swaminathan et al., 1998); it takes a long time for its development, models tend to be very specific and have limited reuse, and they lack optimization functionality. In general, the modeling representations adopted by the traditional techniques tend to be unrealistic. They mostly employ a centralized decision-making treatment, typically involve a single comprehensive model, operate under total information symmetry (every bit of information is known to every one else or at least available to the model builder/decision maker), and they lack adequate representation of uncertainties (deterministic and system dynamics models, for example). Although analytical models are capable of providing precise closed-form solutions, they cannot handle the complexities (Julka et al., 2002a) associated with SC models; they require constraining and unrealistic assumptions in order to solve even small-scale problems limiting their practical utility in problems of real-world sizes. Traditional SC modeling techniques, in general, call for considerable modeling effort (Swaminathan et al.,

1998) and they lack the ability to integrate various SC decision making processes (Julka et al., 2002b). Despite these limitations, organizations have been applying these techniques for several decades benefiting in terms of higher efficiencies, and they are expected to be utilized in particular for intra-enterprise applications even in the near future. Given that intra-enterprise modeling helped improve the efficiencies a great deal, further large-scale improvements have to essentially accrue from modeling inter-enterprise issues. Since, most of the SCs involve enterprises with independent ownerships (requiring the ability to model information asymmetry and distributed/decentralized mode of controls); applicability of the traditional modeling techniques is quite limited and indeed unrealistic. Based on the research literature we identify crucial gaps in traditional SC modeling that we attempt to partially fill in this dissertation research. Literature presents a clear need for:

- rapidly (re-)configurable, scalable, and high fidelity SC models
- realistic models incorporating:
    - distributed decision-making
    - information asymmetry
    - right level of abstraction
    - ability to support planning at different hierarchical levels

Recent trends in computing and object-oriented technologies facilitate modularization and reusability in SC modeling (e.g., Bagchi et al., 1998; Biswas and Narahari, 2004; Feigin et al., 1996; Lin et al., 2002). In conjunction with

Graphical-User-Interface (GUI) based technologies, they play a major role in simplifying the model description, speeding up development, and helping solve bigger and more realistic models. Latest advancements in the modeling technologies have led to the evolution of a promising new approach – agent-based/multi-agent modeling. Its popularity is attributed to a large extent to its ability in: promoting a natural form of modeling (Klugl et al., 2002); involving the key notion of autonomous agent/s (Bond and Gasser, 1988) having the ability to learn (Shen and Norrie, 1999; Nissan, 2001; Honaver, 1999); and incorporating information asymmetry, decentralized decision-making over distributed environments, and process integration (Yuan et al., 2001; Lau et al. 2002; Gan et al., 2000; Umeda and Jones, 1998; Davis, 2001). Multi-Agent Systems (MASs) are becoming increasingly popular due to their inherent ability to model the distributed and autonomous features (including information asymmetry) of various entities constituting a SC in the most natural and realistic way. Evolved as one of the two primary areas of “Distributed Artificial Intelligence (DAI)” (Bond and Gasser, 1988), MASs are being developed with a long-term goal of making the agents interact as well as human beings do, or even better (Weiss, 1999). They are expected to pioneer a revolutionary paradigm shift (Zambonelli and Parunak, 2003; Jennings et al., 1998), and can be used to model any phenomenon, scientific or behavioral, in order to study the underlying dynamics. In terms of their application potential, MASs are best suited and hold a great promise for modeling complex and adaptive real-world systems, in particular, SCs. They have the ability to integrate within the well known decision modeling

techniques (e.g., simulation, optimization, …) and diverse fields of knowledge (e.g., game theory, computational intelligence, behavioral sciences, cognition …) leading to potentially one of the most powerful modeling paradigms known ever. Table 1.1 provides a comparison of various modeling techniques and is self-explanatory. It clearly illustrates some of the superior modeling capabilities of MAS-based modeling – e.g., modeling across groups of firms, information asymmetry, easier migration from modeling to control, most natural representation, superior modeling abstractions based on human practical reasoning, distributed control, ability to adapt, learn, and incorporate intelligence.

Table 1.1: Comparison of Modeling Techniques

(a) System Dynamics vs. Agent / Multi-Agent Based Simulation

Feature	System Dynamics Simulation	Agent / Multi-Agent Based Simulation
Information asymmetry [5]	Does not support	Supports Information Asymmetry
Modeling across groups of firms [5]	Not possible	Allows naturally groups of firms to conduct joint modeling exercises
Natural unit of decomposition [5]	Equation Modularization crosses boundaries among individuals	Individual Modularization follows boundaries among individuals
Suitability and appropriateness [5]	Purely physical processes Most naturally applied to systems that can be modeled centrally Dynamics dominant by physical laws rather than information processing	Most naturally represent “Business Processes” with features: step-by-step processes, conditional decisions, discrete decisions Most appropriate for domains with high degree of localization, distribution, discrete decisions

Feature	System Dynamics Simulation	Agent / Multi-Agent Based Simulation
System representation [5]	System is represented as a set of equations that relate observables to one another No explicit representation of behaviors	System is made up of interacting individuals with corresponding behaviors Relationships among the observables are an output of the process not an input
Migration from simulation model to adaptive control [5]	Not straightforward	Much more straightforward, since one-to-one correspondence between: Agent – Individual, and behavior – real behavior Simulated agent can be used for automated control handling routine interactions
Model dynamics [5]	Observables explicitly drive model's dynamics	Outcome of interactions among individuals
Model features [5]	Have no intrinsic model of space Partial differential equations provide parsimonious model of physical space but not interaction space	Make it easier to distinguish physical space from interaction space Easier to construct
Validation feasibility [5]	Only at system level	System Level and Individual Level
Unit of analysis [6]	Structure of system	Agent's Rules
Structure of system [6]	Fixed	Not Fixed Rules can be adaptive
Modeling analogy [6]	Modeling the Forest	Modeling the Trees of a Forest
Modeling purpose [4]	Understand the dynamics of System interconnectedness	Framework for modeling components based on other modeling techniques, or provide agent models embedded into larger systems
Model usage [4]	Extremely useful for identifying important variables and causal	Useful for combining other techniques through "model

Feature	System Dynamics Simulation	Agent / Multi-Agent Based Simulation
	linkages in a system and for structuring many aspects of model development	blending”

[4] (Macal and North, 2005); [5] (Van Dyke Parunak et al., 1998); [6] (Schieritz and Grobler, 2003)

(b) Discrete Event vs. Agent / Multi-Agent Based Simulation

Feature	Discrete Event Simulation	Agent / Multi-Agent Based Simulation
Structure preservation [3]	“No” Close match between entities of reality and model, and simulated software	Close match between entities of reality and model, and simulated software
Distributed computation [3]	Not straightforward	Supports in a very natural way Allows better performance and scalability
Support for extremely dynamic simulation scenarios [3]	Does not support	Enables add or remove agents at runtime without interrupting simulation Possible to swap an agent for corresponding simulated entity (e.g., a real person during simulation)
Modeling abstraction [3]	Low Level	Very High Level (e.g., in terms of BDI) Easier for non-programmers to understand and participate in the development process
Level of pro-activeness [3]	None	Proactive and fully autonomous
Modeling purpose [4]	Understand the effects of uncertainty in a process	Framework for modeling components based on other modeling techniques, or provide agent models embedded into larger systems
Model usage [4]	Taking a process view of the system and for dealing with stochastic	Useful for combining other techniques through “model blending”

Feature	Discrete Event Simulation	Agent / Multi-Agent Based Simulation
	uncertainty	

[3] (Davidson, 2000); [4] (Macal and North, 2005)

(c) Object Oriented vs. Agent / Multi-Agent Based Simulation

Feature	Object Oriented Simulation	Agent / Multi-Agent Based Simulation
Nature of communication [3]	No Communication Languages – Simple Signals (e.g., Procedure Calls)	Based on Full Agent Communication Languages
Spatial explicitness [3]	No notion of Space	Each entity assigned a location in Simulated Physical Space / Different Machines
Mobility [3]	All entities are Stationary	Mobility in Simulated Physical Space / Different Machines
Modeling concepts [3]	Traditional	Usage of Mentalistic concepts, such as Beliefs, Desires, and Intentions (BDI)
Primary unit [1,2]	Object	Agent
Unit behavior [1,2]	Modular	Modular
Unit state [1,2]	Internal	Internal
Unit invocation [1,2]	External (Message)	Internal (Rules, Goals)
Functionality [2]	Can be made Intelligent, but still remain inferior to Agents	Can be made Intelligent more naturally through reasoning-based inference and neural networks
Support for business concepts [2]	Not directly supported	Supports concepts such as: Rules, Constraints, Goals, Beliefs, Desires, and Responsibilities
Adaptivity [2]	Completely Static Entities	Entities can learn Autonomously / Dynamically
Structure preservation [3]	Close match between entities of reality and model, and simulated	Close match between entities of reality and model,

Feature	Object Oriented Simulation	Agent / Multi-Agent Based Simulation
	software	and simulated software
Level of pro-activeness [3]	Purely reactive	Proactive and fully autonomous

[1] (Odell, 2002); [2] (Ilgar, 1998); [3] (Davidson, 2000)

For more than a decade, MASs are projected to be the next generation modeling paradigm. Researchers have already begun to explore MASs for better SC modeling and control (e.g.: Fox et al., 2000; Sadeh et al., 2001; Wagner et al., 2003; Lin and Shaw, 1998; Nissan, 2001; Swaminathan et al., 1998). Despite a few a real-world commercial applications in the recent past (refer Jennings et al., 2000; Belecheanu et al., 2006), multi-agent supply chain literature is scant in terms of richness of applications and implementations. The process of MAS development still remains quite involved and extremely time consuming, hindering their wide-spread adoption in industrial-strength applications including SCs. We identify based on our extensive literature review (provided in each of the subsequent chapters as well) three fundamental issues for this state of affairs as:

- Lack of detailed and “mature” development methodologies – in fact, to the best of our knowledge there exists no generic methodology for modeling SCs using MASs
- “Work-in-process” development toolkits lacking maturity, functionality, and user friendliness
- No standards / still “evolving” standards

Given that their development is quite involved and extremely time consuming, the necessary infrastructure support plays an extremely important role in the large-scale adoption of MASs. We argue that such a support simplifies the model building process potentially leading to the proliferation of real-world implementations. Accordingly, our research focuses on simplifying MAS development, in particular, for SC applications. In our research we bridge the gap on the first two aspects listed above partially, by developing methodological frameworks for modeling SCs through MASs.

1.1 Problem Statement

We now formally make our research problem statement as follows.

The overall goal of our research is:

> ***"To develop methodologies, tools, and infrastructure necessary to facilitate rapidly (re-)configurable, scalable, and high-fidelity SC models for accuracy, efficient knowledge acquisition, distributed decision-making, and information asymmetry."***

While addressing this over all goal, we identify specific objectives that we set out to achieve through this dissertation research as:

- ***Design and Develop Multi-Agent Frameworks for:***
  - ***Analysis and Design of Multi-Agent Supply Chain Systems***
  - ***Implementation of Multi-Agent Supply Chain Systems***
  - ***Rational Agency through BDI-reasoning***
- ***Validate the developed frameworks through a short-life-cycle product SC case "Tamagotchi" (Higuchi and Troutt, 2004)***

1.2 Dissertation Overview

Having formally made our research statement in the previous section, we now present an overview on how this dissertation is organized. As indicated in Fig.1.1, MAS development like any other software development process involves the phases of requirements analysis, system analysis, system design (both architectural & detailed), and implementation phases. The first of the three frameworks developed and validated in this research, MASCF, focuses on the conceptual analysis and design of MASCSs. Where as the implementation framework, software agent-component based framework, has somewhat less focus on the design but has a predominant focus on implementation of MASCSs. The third framework, rational agent framework, is an extension of the software agent-component based framework that incorporates BDI-reasoning based

rationality. All the frameworks are discussed in detail separately in the following chapters.

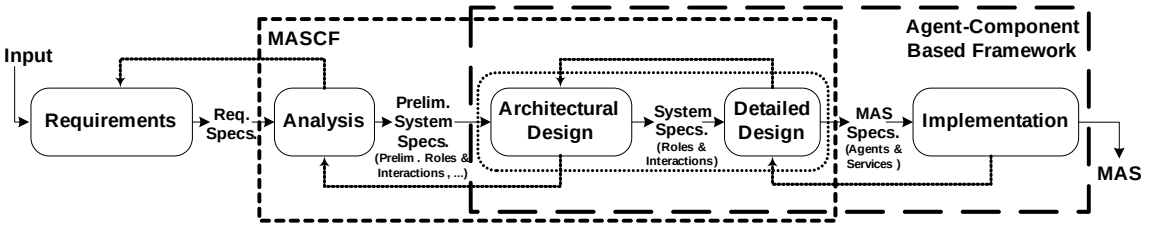


Figure 1.1: Methodological Frameworks for MASCS – Research Focus

The first framework titled “Multi-Agent Supply Chain Framework (MASCF)” is presented in Chapter 2. It provides a generic methodology for analysis and design of MASCSs that creatively adopts Supply-Chain Operations Reference-model (SCOR) to the popular generic MAS development methodology, Gaia. The framework takes requirements specifications of a MASCS as an input and generates MASCS software specifications in the form of Agents and Services models along with their detailed schemas as an output. The framework is validated by analyzing and designing the Tamagotchi case. Although we used Java Agent DEvelopment Framework (JADE) for implementation in our research in order to validate the framework, the output of MASCF can be implemented using any MAS development toolkit.

The details of a framework for implementing MASCSs, titled “Software Agent-Component Based Framework”, are presented in Chapter 3. Given the software specifications for a MASCS to be implemented as an input, this framework provides necessary infrastructure support in the form of integrated tools and technologies; and modeling support in terms of pre-developed reusable

components (agents, behaviours, and policy objects), reducing SC modeling into a matter of quick configuration in order to perform multi-agent simulations and/or study coordination issues. The framework is validated by implementing the Tamagotchi case.

The details of a rational agent framework that extends the software agent-component based framework and utilizes BDI-reasoning capable agents for modeling MASCSs are presented in Chapter 4. The first two frameworks developed in this research have a predominant focus on agent externalities. What gives power to an agent is its internal architecture, and that is where this framework plays an important role by introducing the BDI-reasoning ability into the agents. In order to validate the framework, multi-agent simulations are performed by incorporating elementary rational investment decisions by the Facility Agent of Tamagotchi case.

Finally, Chapter 5 summarizes the research work carried out, and outlines some of the potential future research extensions.

CHAPTER 2

MASCF: A GENERIC PROCESS-CENTERED METHODOLOGICAL FRAMEWORK FOR ANALYSIS AND DESIGN OF MULTI-AGENT SUPPLY CHAIN SYSTEMS

Multi-agent systems (MASs) are becoming popular for modeling complex systems such as supply chains (SCs). However, the development of MASs remain quite involved and extremely time consuming. Currently, there exist no generic methodologies for modeling SCs using MASs. In this research, we propose a generic process-centered methodological framework, Multi-Agent Supply Chain Framework (MASCF), to simplify MAS development for SC applications. The framework introduces the notion of process-centered organization metaphor, and creatively adopts Supply Chain Operations Reference (SCOR) model to a well-structured generic MAS analysis and design methodology, Gaia, for multi-agent supply chain system (MASCS) development. The popular Tamagotchi case was designed and analyzed using MASCF. The validity of the framework was established by implementing MASCF output of Tamagotchi SC using Java-Agent DEvelopment Framework (JADE).

2.1 Introduction

Supply chain management (SCM) has emerged as an invaluable strategic initiative providing competitive advantage for enterprises in the market place. Gartner-Dataquest Report (2005) defines SCM as a business strategy of

integrating business processes from a supplier's supplier to a customer's customer that creates and fulfills the market's demand for goods and services, and optimizes the flow of products, services and information. Chopra and Meindl (2006) define SC to be the dynamic involving constant flow of information, products, and funds between different stages that perform different processes while interacting with the other stages. Christopher (1998) defines SC as a network of organizations linked through upstream and downstream processes that add value to the ultimate customer through products and services. It is clear from these definitions that the sheer scope of a SC makes its efficient management a complex task. In today's global marketplace, as more and more firms embrace SCM, individual firms no longer compete as independent enterprises but rather as integral parts of SCs (Min and Zhou, 2002; Lambert et al., 1998). The constant battle for supremacy is no longer between an enterprise and its competitors, but between the SC of the enterprise and those of its competitors (Taylor, 2003; Baatz, 1995). The success of any enterprise, accordingly, depends on its ability to integrate and coordinate the intricate network of business processes among its SC partners efficiently (Lambert and Cooper, 2000; Drucker, 1998). It follows that the ultimate success of an enterprise is not derived independent of, but coupled with the destiny of its SC.

Traditional SC modeling and management involves the application of: optimization (e.g., Arntzen et al., 1995; Goetschalckx et al., 2002; Lee and Billington, 1995; Beamon, 1998), mathematical models (e.g., Cachon, 2003; Cachon and Fisher, 2000; Anupindi and Bassok, 1999; Lee and Whang, 1999),

simulation (e.g., Holweg and Bicheno, 2002; Bhaskaran, 1998; Terzi and Cavalieri, 2004; Chan et al., 2002; Petrovic, 2001), system dynamics (e.g., Higuchi and Troutt, 2004; Sterman, 1989; Angerhofer and Angelides, 2000), and others. These approaches usually employ a centralized decision-making treatment, and typically involve a single comprehensive model, under the assumption of information symmetry (every bit of information is known to every one else or at least available to the model builder/decision maker). Another trend in this area that is gaining prominence is the usage a combination of tools in decision modeling, for example: simulation-optimization (Padmos et al., 1999). Latest developments in the computing and object-oriented technologies facilitate modularization and development of reusable objects leading to rapid development of models (e.g., Bagchi et al., 1998; Biswas and Narahari, 2004). In addition, Graphical-User-Interface (GUI) based technologies available today simplify model description through drag-and-drop features. All these advancements play a crucial role in developing and solving bigger and more realistic models quickly. Traditional modeling techniques are quite suitable for modeling SC decisions within a single enterprise. Organizations have been applying these techniques for several decades leading to higher efficiencies. Given that intra-enterprise modeling helped improve the efficiencies a great deal, modeling of inter-enterprise issues for SC integration are crucial for further large-scale improvements. Considering the fact that most of the SCs involve enterprises with independent ownerships (requiring the ability to model information asymmetry and distributed/decentralized mode of controls),

applicability of the traditional modeling approaches is quite limited and indeed unrealistic. The latest developments in the modeling technology, agent-based systems, and MASs for example, are quite promising for such modeling situations. They are best suited to handle issues of information asymmetry, decentralized and distributed decision-making, and modeling inter-enterprise issues.

Agent technologies are an offshoot of “Distributed Artificial Intelligence (DAI)” that has a long-term goal of developing mechanisms and methods that enable agents to interact as well as human beings, or even better (Weiss, 1999). Autonomous agents and MASs represent a new way of analyzing, designing, and implementing complex software systems (Jennings et al., 1998). They are expected to pioneer a revolutionary paradigm shift in software systems modeling and engineering (Zambonelli and Parunak, 2003). MASs can be used to model any phenomenon, scientific or behavioral, in order to study the underlying dynamics of complex systems such as SCs very effectively. Agents can be modeled to represent organizations, functions, resources, and even human beings. They have the ability to incorporate within, some of the existing modeling approaches (e.g., optimization, simulation, game theory), making them more powerful. They can also be made to learn with the help of artificial intelligence tools and techniques, leading to “intelligent” agents. This, however, does not mean that “agents” are the panacea for all the modeling issues. Wooldridge and Jennings (1998; 1999) while emphasizing that intelligent agent and MASs can potentially play a significant role in complex and distributed systems engineering,

warn of avoiding potential pitfalls in engineering industrial strength agent-oriented software systems. Multi-agent SC literature is scant in terms of richness of real-world applications and implementations. While this is expected due to being a relatively new and upcoming field, there exist several other reasons too. For example, the modeling standards are still being evolved and the infrastructural support in terms of tools and methodologies are still in their nascent stage of development. Given that the development of MASs is quite involved and time consuming, the necessary infrastructure support plays an extremely important role in their large-scale adoption. We argue that such a support simplifies the model building process leading to the proliferation of real-world implementations.

Our research accordingly focuses on simplifying MAS development, in particular, for SC applications. We propose Multi-Agent Supply Chain Framework (MASCF), a generic process-centered methodological framework, towards this goal. MASCF is designed to facilitate and simplify the analysis and design phases of the development. Instead of developing yet another methodology afresh from ground zero, we design our framework around already well established model/methodology that become its key elements. The framework introduces the notion of process-centered organization metaphor, and creatively adopts a generic process-standard for SC description (Supply-Chain Operations-Reference model, SCOR) to a well-structured generic methodology for MAS development (Gaia). Since SCOR and Gaia are the major elements of MASCF, our framework is a generic tool widely applicable, and practical for modeling SCs through MASs. This particular chapter is structured as follows. Pertinent literature

is reviewed in the following section 2.2. Section 2.3 describes the specific elements of the framework. MASCF is introduced in section 2.4 along with its scope and limitations. The operating mechanics of the framework are discussed in detail in section 2.5. The framework is validated with the help of a case study, Tamagotchi, and its details are provided in section 2.6. Finally, section 2.7 summarizes the contribution of the research detailed in this chapter and identifies some of its extensions.

2.2 Supply Chain Modeling, Management & Multi-Agent Systems

This section reviews the literature pertaining to multi-agent based SCM, some of the more prominent agent-oriented methodologies, process-centered SCM, and wraps up with a discussion on some of the research gaps. Before doing so however, a brief note on MASs is provided here. 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 a couple of reasons for the popularity of MASs (systems with multiple interacting agents): (i) modern computing and information environments are distributed, large, open, and heterogeneous, and (ii) MASs 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, SCs. 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.

2.2.1 Multi-Agent Supply Chain Modeling & Management

Researchers have been exploring MASs in order to better model various SC 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 SCs at the tactical and operational levels. They develop an “agent building shell (ABS)” 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 SC. They demonstrate that agents increase flexibility, and enable the SC 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 SC agents that conduct business on behalf of product users, buyers and vendors. In the context of SC integration in a major enterprise, an agent-based SC 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 SC 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 SC 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 discuss the system details. A framework that integrates various elements of a SC 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 SC. 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. (2006) propose an agent-based multi-contract negotiation system for global SC 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 SCs 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 Personal Computer SC application prototype. Gjerdrum et al. (2001) apply multi-agent modeling techniques to simulate and control a simple demand-driven SC network system. They utilize Java Agent Template Lite (JATLite) for modeling the MAS 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 SC 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 exist 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. The literature 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 directly take up implementation from pre-stated requirements. We did not come across any literature that views MASCS development as a generic process. As a preliminary step in that direction, researchers have begun developing generic methodologies, and toolkits for MASs.

2.2.2 Agent-Oriented Methodologies

An interesting aspect is that while agent-based systems are becoming increasingly well understood, the development of MASs is not (Wooldridge and Jennings, 1999). As agents are autonomous, the development of MASs differ from others, requiring valuable contributions from software engineering methodologies (Cossentino, 2005; Odell, 2005). Henderson-Sellers and Giorgini (2005) compile ten most prominent agent-oriented methodologies and state that most of them are being proposed by academic researchers and are still in an early stage of maturity. They range from extensions of existing object-oriented methodologies to new agent-oriented techniques utilizing new modeling abstractions (O'Malley and DeLoach, 2001). These methodologies differ in software development phases, support for inter- and intra-agent support aspects, and modeling the environment in which the system operates (Weiss, 2001). A

brief review of some of the more popular agent-oriented methodologies is presented below.

One of the earliest and generic multi-agent methodologies, Gaia, is proposed by Wooldridge et al. (2000). It views MAS as a computational organization consisting of various interacting roles, dealing with both the macro-level (societal) as well as micro-level (agent) aspects. Zambonelli et al. (2003) extended the methodology by incorporating environment and several other key organizational abstractions. “Gaia” refers to the extended version by Zambonelli et al. (2003), from here on. Using the analogy of human-based organizations, Gaia facilitates the interaction of both a developer and a non-technical domain expert (Henderson-Sellers and Giorgini, 2005). Gaia is one of the most promising approaches as far as the analysis and design of large, distributed, open systems are concerned (Cernuzzi et al., 2005; Karim, 2004). Inspired by the organizational concepts, unlike the existing development (structured or object-oriented) methodologies, Castro et al. (2002) propose Tropos. Its development is based on two key ideas: methodology should cover the early requirements analysis phase and notions like agents, and use goals and plans from early analysis to actual implementation (Bresciani et al., 2004). Giorgini et al. (2005) develop a formal goal model for the requirements analysis phase in order to make the goal analysis concrete based on forward and backward reasoning. Deloach et al. (2001) describe a general-purpose methodology, MaSE – Multi-agent Systems Engineering, for developing heterogeneous MASs. Organized into seven well-defined steps, it describes system goals, behaviors, agent types,

and agent communication interfaces with the help of a number of graphical based models. The methodology is supported by a graphical, fully interactive tool, agentTool (DeLoach and Kumar, 2005) for implementation. DeLoach (2005) extends the methodology to include organizational modeling concepts like goals, roles, agents, capabilities, and the assignment of agents to roles. PASSI (Process for Agent Societies Specification and Implementation), is a requirement-to-code methodology for designing and developing multi-agent societies (Cossentino and Sabatucci, 2004; Cossentino, 2005). PASSI characterizes the system development using five process components (models) that are divided into phases and described using UML diagrams. Chella et al. (2006) develop an agile version (Agile PASSI) that exploits the features of reusable patterns. In order to facilitate implementation, a PASSI ToolKit (PTK), was developed as a plug-in for IBM's commercial tool Rational Rose.

We strongly believe and argue in favor of treating the development of MASs 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. Most of the existing agent-methodologies are generic but highly conceptual (e.g., Gaia, Tropos). Some of the methodologies (e.g., MaSE, PASSI) offer tool-based support for implementation. Most of the methodologies otherwise are weak in generic conceptual level system analysis and design. Odell (2005) summarizes

the agent-oriented methodology state of affairs in a nutshell as “each methodology has its own unique perspective; however, no one single methodology is useful in every situation.”

2.2.3 Process-Centered Supply Chain Management

Traditionally enterprises are organized in a function-oriented mode with various specialized functions working together to provide products and services to customer. With the advent of new technologies, some of them started reaping huge benefits by integrating the functional operations in a process-oriented mode. A few visionary enterprises began to virtually integrate the processes across enterprise boundaries to derive enormous benefits. The association between Wal-Mart and Procter & Gamble (Huang et al., 2002; Hammer and Champy, 1993) is the most often cited example. Such industrial implementations resulted in research that later came into existence as: Business Process Reengineering (Hammer, 1990; Hammer and Champy, 1993; Davenport, 1992; Davenport and Short 1990; Davenport, 1995), and Process-Centric Enterprises (Hammer, 1997; Hammer and Stanton, 1999). The dissemination of research knowledge fueled further widespread adoption of process-centered framework. The next logical extensions for both the research and industrial implementations are obviously to integrate processes across SCs (Hammer, 2001; 2003).

Among five process-oriented frameworks for SCM only two, the Global Supply Chain Forum (GSCF) framework and the Supply-Chain Operations Reference-model (SCOR), are detailed enough and include business processes

to achieve cross-functional integration (Lambert et al., 2005). The GSCF framework (Cooper et al., 1997; Lambert, 2004; Lambert et al., 1998) includes eight SCM processes with Customer relationship management and supplier relationship management forming the critical links with the rest. Each of the processes is cross-functional/cross-firm, and decomposed into a sequence of strategic and operational sub-processes described by a set of activities (Lambert et al., 2005). Developed by the Supply-Chain Council (SCC), SCOR offers a generic process reference model and has become a cross-industry standard for SCM (SCC, 2005). Through standard process descriptions, SCOR helps for gaining a unified understanding and comparing the operations of different SCs. At the highest level in SCOR, SCs are represented by five macro level processes – Plan, Source, Make, Deliver, and Return. In the subsequent levels, each of these processes is decomposed and described in increasing details. Enterprise specific process descriptions are documented at level four and below and are beyond its scope however.

2.2.4 Gaps in current research

The available literature does not present any generic methodology that is applicable for modeling SCs using MASs. Clearly defining, speeding up, and simplifying the process of development are essential for the widespread adoption of multi-agent paradigm by the industry, be it for SC modeling or otherwise. However, for such an adoption to materialize in practice, development methodologies have to offer much more precise and standard descriptions that

aid in the conceptual analysis process. Standard way of expressing various components and their attributes will help speed up MAS development. For this to happen, however, there needs to be in place standards that are acceptable to a wide range of industries and organizations. In the area of SCs, SCOR provides the standards that define and describe SC operational processes of any organization in a standard way. In this research, we propose MASCF that creatively adopts SCOR to the Gaia methodology in order to carry out the analysis and design phases of MASCS development.

2.3 Key Elements of MASCF

A brief description of some of the key elements of the MASCF is presented in this section. The focus here is on: SCOR-based SC modeling, Gaia-based analysis and design of MASs, and the notion of process-centered organization metaphor. How these elements complement each other, and work inside the framework, are meant for discussion in later sections.

2.3.1 Supply Chain Modeling using SCOR

Much of the material presented here to briefly describe SCOR and its role in SC modeling is adopted from SCC (2001; 2005). Since its introduction, SCOR has gained widespread acceptance in both the practicing as well as research communities world-wide. It is a process reference model that provides: a standard description of SC management processes, a framework of relationships

among the standard processes, standard metrics for measuring process performance, best-in-class management practices, and standard alignment to feature and functionality. SCOR is designed to be a standard language that helps management focus on both intra and inter-company SCs through effective communication among SC partners. It is used to describe, measure, and evaluate SC configurations in order to achieve competitive advantage.

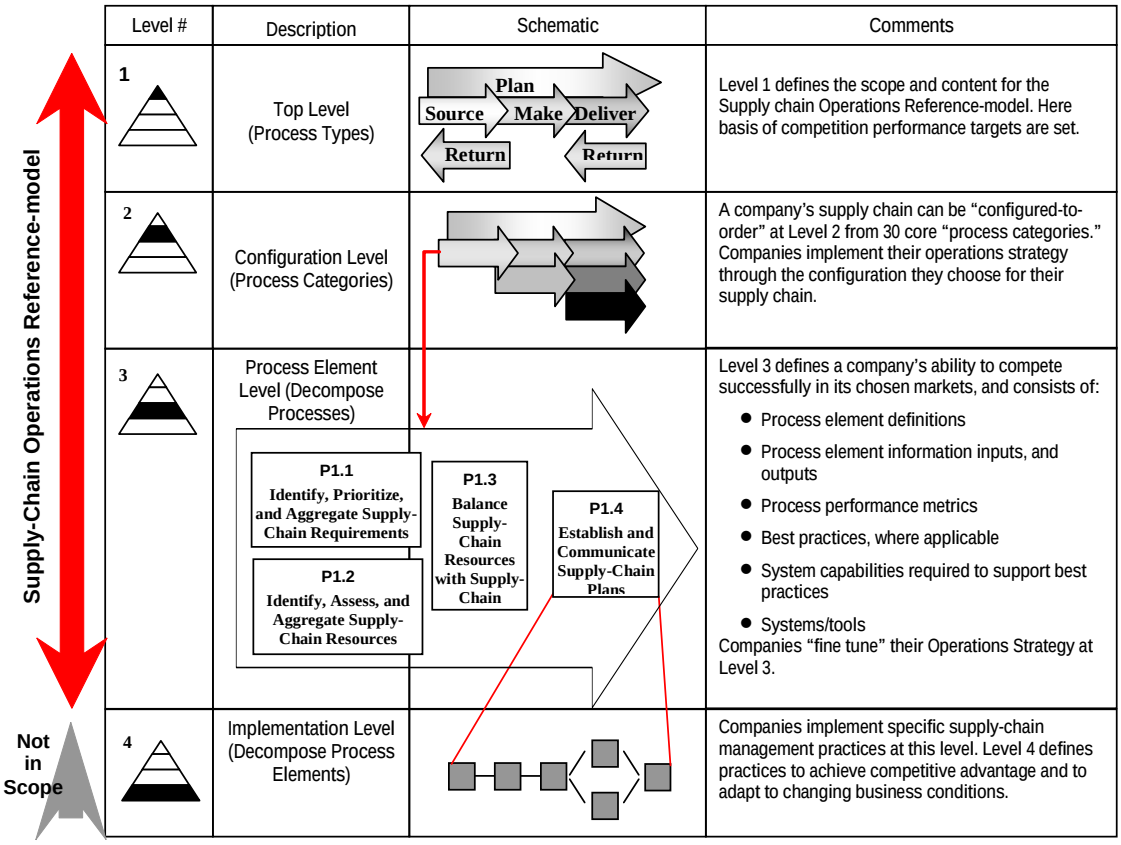


Figure 2.1: Supply-Chain Operations Reference-model (SCOR)
(source: SCC, 2005)

As described in Fig. 2.1, SCOR contains three levels of process detail that help integrate a SC from supplier's supplier to customer's customer:

- The process definitions level (Level 1) provides a macro-level description of the entire SC operations. It defines the scope and content for the SC model with the help of five distinct management processes – plan, source, make, deliver, and return.
- At the process categories level (Level 2), a SC is configured using a set of core process categories defined by the relationship between SCOR level 1 processes and process types relating to planning, execution, and enabling. This level of analysis helps in defining as-is or the ideal state of operations of every organization in the SC. At this level SCOR is applied for configuring SC threads and developing process maps to understand each distinct thread.
- At the process elements level (Level 3), each of the level 2 process categories is configured using a set of process elements. It is at this level that we can see more details of the process element logic in terms of process flows, sources, and destinations of inputs and outputs.

The implementation of company specific SCM practices occurs at decompose process elements level (Level 4) and below and is beyond the scope of SCOR. It is within these levels, each process element is described by classic hierarchical process decomposition. These practices provide competitive advantage and help a company adapt to changing business conditions. The model's logic supports horizontal process integration, with each basic SC being a chain of source, make, and deliver execution processes. These execution

processes transform/transport materials and/or products with each process being a customer to the previous one and a supplier to the next. Planning processes manage these customer-supplier links between two execution processes and balance the SC. Configuring a SC “thread” illustrates how SCOR configurations can be used to describe, measure, and evaluate SC configurations. Since SCOR is developed to be a process reference model, it can model any SC using a set of standard process descriptions. It neither deals with any specific modeling technique like optimization or simulation nor focuses on multi-agent modeling. MASCF creatively utilizes the details of SCOR to improve the precision of multi-agent methodology to better model operational dynamics.

2.3.2 Analysis & Design of Multi-Agent Systems using Gaia

By design, Gaia methodology is generic, comprehensive, and neutral with respect to the target domain and agent architecture, and appropriate for the development of large-scale real-world applications. In this sub-section its details are presented briefly. For a detailed explanation of the methodology, readers are suggested to refer Wooldridge et al. (2000), Zambonelli et al. (2003; 2005), and Cernuzzi et al. (2004). Gaia encourages the developer to think of building MASs as a process of organizational design. Using a water-fall based approach, it allows an analyst to go systematically from requirements to design. Early requirements analysis and the actual development phases are beyond its scope, however. As illustrated in Fig. 2.2, the Gaia methodology is organized into

analysis, architectural design, and detailed design phases. The output of Gaia is sufficiently detailed for implementation using any generic multi-agent toolkit.

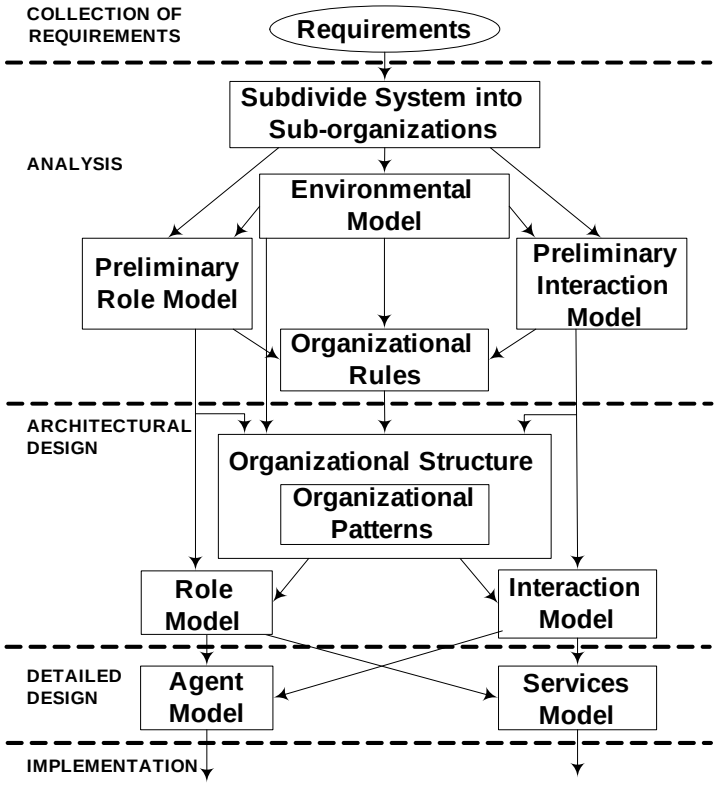


Figure 2.2: Gaia Methodology (source: Zambonelli et al. 2003)

2.3.2.1 Analysis Phase

The primary goal of the analysis phase is to organize specifications and requirements of the system-to-be into an environmental, preliminary role and interaction models, and a set of organizational rules; for all sub-organizations comprising the overall system. Such conceptual analysis involves the following tasks:

- The first step is to determine whether the system has to be decomposed into multiple MASs based on system specification, real-world structures, modularity, and efficiency considerations.
- It is very difficult to provide generic modeling abstractions without considering a specific application and the underlying technology. In order to develop a reasonably general approach, Gaia treats environment in terms of abstract computation resources, such as variables or tuples, made available to agents for sensing (reading), effecting (changing) or consuming (extracting).
- Preliminary roles are identified from the “basic skills” and their basic interaction needs of the system. A role is represented by two main attributes – permissions, and responsibilities.
    - Permissions identify legitimately usable resources (typically, information), and their limits in realizing the responsibilities (e.g., a role might have the permission to generate/read/modify certain information/variables associated with it). In general, permissions relate agent roles to the environment, and also to the information received from the other roles.
    - Responsibilities determine the expected behavior and have two properties: liveness, and safety. Liveness properties identify the various activities and interactions. The “life-cycle” of a role is specified using a liveness expression that has activities, and protocols as its atomic components. Activities correspond to a

unit of action that the role of an agent performs on its own without involving the other agents. Protocols, on the other hand, are the activities that do require interaction with others. Safety properties are invariants that define the requirements and constraints to be ensured over the variables identified in a role's permission attribute.

- The preliminary interactions model captures the dependencies between the various roles, in terms of one protocol definition for each type of inter-role interaction. Gaia views a protocol as an institutionalized pattern of interaction, focusing on the essential nature and purpose, rather than the precise sequence of execution steps and message exchanges. A protocol definition includes the attributes: protocol name, initiator, partner, inputs, outputs, and description.
- Organizational rules capture the general relationships between roles, between protocols, and between roles and protocols. Gaia considers organizational rules as responsibilities of the organization as a whole and categorizes them as: safety and liveness organizational rules. Liveness rules define the sequence of execution of roles or protocols and relate to the way different roles play specific activities. While, safety rules define time-independent global invariants for the organization (e.g., a role must be played by only one entity, or that two roles can never be played by the same entity).

The output of the analysis phase consisting of the above models forms the input to the design phase. The design phase of Gaia is logically decomposed into the architectural and detailed design phases.

2.3.2.2 Architectural Design

The architectural design phase involves the definition of the system's organizational structure and completion of preliminary role and interaction models. Organizational structure is defined in terms of its topology and control regime. Consideration is given to various aspects such as the organizational efficiency, real-world structure, and the need for rules enforcement in designing an appropriate organizational structure. Topology could range from the simplest to the most complex: a single-member, collections of peers, hierarchy, multi-level hierarchy, and hybrid organizational networks. On the other hand, control regimes range from workload partitioning, workload specialization, to market models. Once the organizational structure is finalized, the role and interaction models can be completely defined. Which involves filling the remaining attribute specifications for the roles and interactions identified in their preliminary models; specifying the attributes in their entirety for the new roles and interactions originating out of the organizational structure definition; as well as the attributes due to interactions between the new roles and the ones identified in the analysis phase.

2.3.2.3 Detailed Design

The detailed design phase is responsible for defining the agent model and the services model that act as guidelines for the actual implementation of agents and their activities. An agent model assigns roles to agents, defines agent classes, and identifies the agent instances of each class to be instantiated in the actual system. Based on convenience and efficiency considerations, the roles-agent assignment would have either one-to-one or many-to-one relationships. The services model identifies coherent blocks of activity in which agents will engage to realize their roles. These services are derived from the list of protocols, activities, responsibilities and liveness properties of the roles that the agents implement. The analyst might define one service for each parallel activity, or multiple services to represent even sequential activity phases of an agent execution. For each service of an agent, it is necessary to document its properties – the inputs, outputs, pre-conditions and post-conditions. Inputs and outputs to services are derived from both the protocol and environmental models. Pre- and post- conditions represent the constraints on the execution and completion of services and are derived from the safety properties, organizational rules, environmental resources, and the data from the other agents. The completion of the Gaia process results in a well-defined specification of MAS that can be implemented using traditional programming or an appropriate toolkit-based agent-development framework.

2.3.3 Process-Centered Organization Metaphor for Multi-Agent Systems

The MAS paradigm, unlike the traditional approaches to software development, requires the adoption of new software engineering abstractions and metaphors (Zambonelli et al., 2005). Stressing the importance of understanding which abstractions are influenced by which metaphors in agent-oriented development, Zambonelli et al. (2003) claim that none of the metaphors in existence can reasonably claim to be general purpose. They identify four different types of metaphors that researchers have been applying – ant (in general, insect) colony (e.g., Hadeli et al., 2004; Dorigo et al., 2000), physical (e.g., Abelson et al., 2000; Mamei et al., 2003), societal (e.g., Veloso et al., 1999; Candea et. al., 2001), and organization (Zambonelli et. al., 2001a & 2001b). They assert that the organization metaphor is perhaps the most appropriate one for a wide-range of systems that involve workflow, process-oriented flow, and decentralized control. Since the purpose of any modeling effort is to mimic the real-world setting in the best possible manner, an organization-based design makes the model more realistic. In most of the agent-oriented methodologies, the analysis process starts with the identification and definition of roles and their interactions directly from the requirements statement. Which means that the actual organization is implicitly determined even before identifying how the organization is expected to work and what kind of organization best fits the requirements. Gaia methodology incorporates consideration of organizational abstractions before the role and the interaction models are finalized. These abstractions relate to the environment in which the MAS is situated, the roles to

be played by the different agents in the organization, the interactions between these roles, organizational rules, and organizational structures. It also suggests that the identification of preliminary roles and interactions be based on identification of “basic skills” required, goal-oriented early requirements analysis, or from organizational structure. This approach is clearly based on function-oriented view of the enterprises. Literature presents ample evidence of function-oriented approaches being replaced by more powerful process-centered approaches (Hammer, 1990, 1997, 2001, 2003; Davenport, 1992, 1995; Hammer and Champy, 1993; Davenport and Short 1990; Hammer and Stanton, 1999) in real-world organizations. We propose that in order to retain realism and gain substantial benefits, the development of MASs should also be based on process-centered organization metaphor instead of the usual function-oriented organization metaphors. Such a metaphor would incorporate not only the usual organization abstractions, but also focus on identification of roles, and interactions based on standardized process descriptions. In doing so, there exists a key role for process reference models such as SCOR. As SCOR already became a cross-industry, world-wide standard for process reference and description, we propose that the identification of the roles and interactions of a MASCS be based on SCOR. Such an approach would help create roles and interactions that are generic, industry independent, and reusable. The crucial role for SCOR in this endeavor is presented in detail in the subsequent sections.

2.4 The MASCF Framework

As illustrated in figure 2.3, the process of MAS development (like any other software development) involves the execution of four phases: the requirements collection, system analysis, design (architectural and detailed), and implementation. Development of a software agent-component based framework that focuses on the design and predominantly on implementation was detailed in Govindu and Chinnam (2006). In contrast, we develop MASCF that focuses predominantly on the system analysis and design phases of development. This section presents the details our framework MASCF, its scope and limitations, and how SCOR improves the precision and efficiency of Gaia methodology in the development of MASCS. By design, MASCF creatively combines the features of SCOR to improve the precision and effectiveness of the Gaia methodology in the analysis and design phases. Requirements collection phase is beyond its scope, and so is the implementation phase.

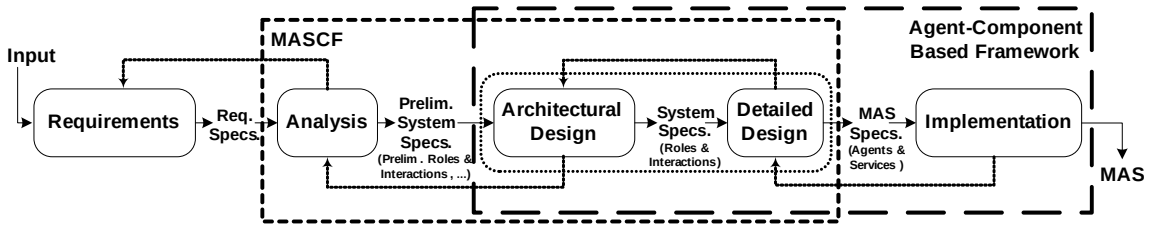


Figure 2.3: Phases in multi-agent system development – focus of MASCF

As Fig. 2.4 indicates, the application of the framework begins with requirements of the system to-be forming an input for the analysis phase involving both SCOR as well as Gaia. It ends with the completion of the detailed

design phase having generated an output in the form of agent, and services models. MASCF also does not commit to any specific development platform, as in Gaia, and allows for implementation using any generic multi-agent toolkit deemed appropriate.

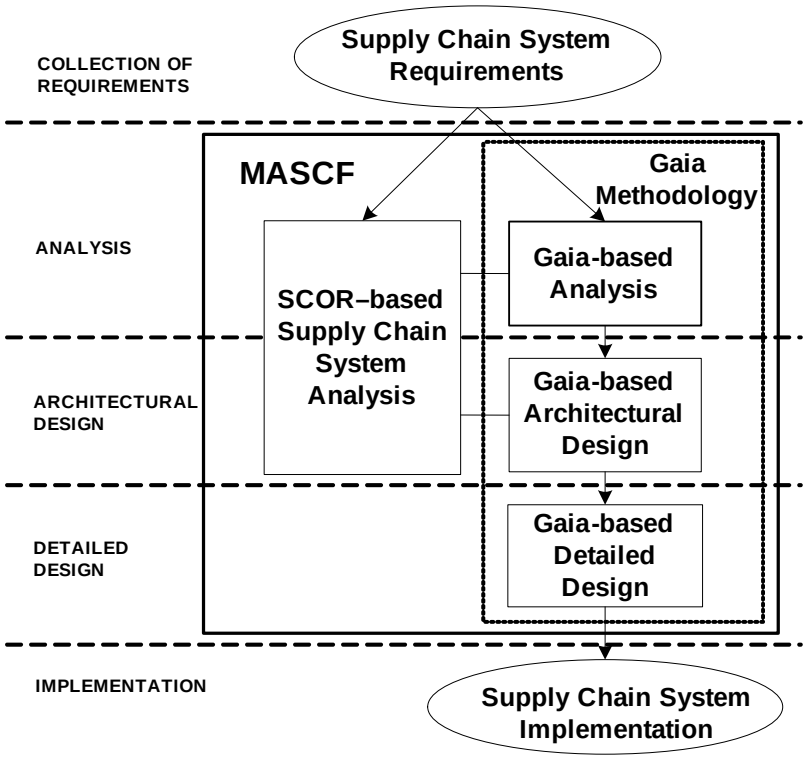


Figure 2.4: Multi-Agent Supply Chain Framework (MASCF)

MASCF conceives SCOR to play a predominantly major role in the analysis phase and to a minimal extent in the architectural design phase but not in detailed design. The part of the analysis phase involving SCOR turns the input system requirements into an information output for defining SCOR-based roles, interactions, and their attributes. This information becomes an input to the various models to be developed in the analysis phase using Gaia methodology.

The procedure of analysis using Gaia would remain the same as in the original methodology. The only exception being the Gaia-based analysis has more precise SCOR-based inputs in addition to the usual system requirements. Therefore, the output of the analysis phase would also be more precise and conforms to the standard process definitions of SCOR. Once the analysis phase using Gaia is complete, it generates an output in the form of SCOR-based attributes relating to environment, roles, interactions, and organizational rules for all the sub-organizations of the SC system. This output becomes an input to the architectural design phase. In addition, there could be some further SCOR-based inputs relating to those roles and interactions that are organizational structure-specific not considered in the analysis phase. With the exception of these additional inputs bringing in more focus, the architectural design phase in the framework would remain the same as in Gaia methodology. The output of the architectural design phase will be an input to the detailed design phase. With SCOR has no role to play, the detailed design phase would remain the same as in Gaia methodology and generate an output in the form of agent and services models for the MASCS under consideration. Having presented the details of our framework, we now focus our discussion on how SCOR improves the precision and efficiency of Gaia.

The stand-alone Gaia methodology offers only generic guidelines for the development of any generic MAS. These guidelines are highly conceptual, meant to provide some directions for carrying out analysis and design leaving a lot of decisions to be made by the individual analysts/developers. Although information

from the actual real-world systems does help bringing in more clarity, the developers would still have to deal with several possible alternatives for each and every element of the MAS, be it in defining the roles, interactions, sub-organizations, environment or the organizational rules. They also have to deal with modeling at the right resolution (aggregate level or detailed level). This makes MAS design a highly complex task. In addition, modeling in real-world situations is never a one time affair warranting modification, expansion, and reduction in scope from time to time depending on the changing requirements. Given these possibilities, we argue that unless more precise descriptions and information is brought into these activities, modeling MASCSs would remain a complex and tedious task. Although individual developers will still have a role to play even in our framework, bringing in precision based on standard process descriptions would help mitigate the complexity to a considerable extent. It is here that SCOR plays a major role in our framework. It compliments the Gaia methodology in defining each element of the system. SCOR-based analysis converts the system requirements into more precise, specific, structured, and well defined input for the analysis using Gaia methodology. This input helps Gaia to carryout a better structured analysis and the output it delivers would be much more precise. It helps in carrying out the design phase much better. A discussion on the operating mechanics of the framework in the next section would elucidate further on how SCOR brings in precision into the Gaia methodology. In order to see more real-world large scale implementations, it is crucial to speed up and simplify the development process. We argue that integration of a process

standard, SCOR, with Gaia is a prerequisite first step in the likely automation of at least some parts of the MAS analysis and design. However, aspects relating to such automation are beyond the scope of this research work. In this work, the focus is strictly on exploiting the elements of SCOR in improving the process of analysis and design of MASCSs.

2.4.1 Scope & Limitations

Before describing the operating mechanics of MASCF, we would like to briefly touch upon its characteristics, scope, and limitations. The framework is designed to adopt creatively, the elements of a powerful and generic process reference model, SCOR, to improve the precision of a generic conceptual methodology, Gaia, in the development of MASCSs. Hence, the framework is:

- Generic – applicable to model a wide range of MASCSs
- Process-Centered – introduces the notion of process-centered organization metaphor and implements it using SCOR
- Methodological – it follows well defined process steps
- Comprehensive – covers all aspects of SC operations, and deals with both macro (societal) level and micro (agent) level aspects of MAS development
- Neutral – with respect to both the target SC domain and the agent architecture within the MAS, and
- Applicable for the development of large-scale real-world MASCSs (both open, and closed)

The scope and limitations of the framework are defined likewise, by the Gaia methodology and SCOR.

- MASCF does not commit to specific techniques for modeling (e.g., roles, environment, and interactions) as the multi-agent standards are still evolving
- It is technology neutral and does not directly deal with implementation
- Activities related to requirements capturing and modeling are beyond its scope
- MASCF covers all operations as scoped by SCOR. Since SCOR is an operations reference-model, not every business process or activity (e.g., demand generation, post-delivery customer support, and administration) is covered in its scope. However, since MASCF is based on a generic methodology Gaia, it would still be possible to model the other aspects (e.g., tactical and strategic) of the systems not covered by SCOR.

2.5 Analysis and Design of Multi-Agent Supply Chain Systems using MASCF

Gaia methodology was briefly introduced in an earlier section along with references for further detailed understanding. Instead of presenting the details on how the framework operates in its entirety, we confine the discussion to only those additional aspects that SC modeling, and SCOR-based integration brings into MASCF. This helps focus only on the contribution of our research and

MASCF, and avoids repetition of how Gaia methodology is applied in practice. The discussion is illustrated, wherever possible and required, with SC specific examples. This section is structured as follows: Using the flow of Gaia methodology within MASCF, the role and contribution of the inputs from SCOR-based analysis are presented at each step. A discussion on SCOR-based Supply Chain System Analysis is presented with the purpose of identifying the specific elements of SCOR that are relevant for specific aspects of the Gaia models in MASCF towards the end of the section.

2.5.1 Gaia-based Analysis in MASCF

SCOR plays a major role in the analysis phase concerned with the development of the organizations, the environmental model, the preliminary role and interaction models, and the organizational rules.

2.5.1.1 Organizations model

The first step in the analysis phase is to define/describe the number (single/multiple), and the scope of sub-organizations that a SC system should comprise of. Ideally, one would like to have at least as many sub-organizations as the number of independent enterprises types (e.g., Manufacturer, Retailer, Supplier). It is also possible that multiple enterprises could exist within each type (e.g., two manufacturers, three retailers, four suppliers). Unless a real-time system is being developed to study lower level transaction-based operations,

considering too many organizations may not be necessary. Having too many organizations in the model might lead to such a complex network that making sense of its collective behavior could become virtually impossible. One would resort to certain level of aggregation if the focus is more on planning and studying behavior at an aggregated level. SCOR considers an organization in the form of a chain of source, make, and deliver processes with the planning process supporting the link between two processes. So, one might consider an organization comprising of a single chain of source-make-deliver processes, and the SC comprising of several such organizations. On the other hand, one could be building only an organizational level system that would be part of a distributed SC. In such cases where the entire SC model would be run in a distributed mode, any organization may have to model only itself. So, there would be just a single organization as far as that particular enterprise is concerned with relevant linkages to immediate organizations upstream and downstream. Even in such cases, however, the complexity and other considerations could force the enterprise to split its own system into multiple sub-systems. Therefore, the analyst must look at the requirements, SCOR, and other considerations before making a decision on the exact number, and the scope of the sub-organizations. It is obvious that, in this process SCOR helps narrowing down the potential space comprising all possible ways of describing/defining organizations comprising a system using precise process descriptions. This potential space would be much larger if it were not used otherwise. Using SCOR defining organization amounts to selecting one from the now narrowed down space. A

good starting point for defining the organizations is to use the requirements statement to configure SC thread and process map using SCOR level 2 process categories.

2.5.1.2 Environmental Model

Gaia considers environment in terms of abstract computation resources, such as variables or tuples, to be made available to agents. Let us consider an example from the SC domain. Assume that level 2 SCOR process category P2: “Plan Source” is a potential role. Pay attention to the word “potential”, since we have neither identified the actual roles comprising preliminary role model yet nor the agents. An agent with the responsibility for that role may have to read a “delivery plan” generated by some software system, and change the “sourcing plan” accordingly. Note that these plans are the input into, and output out of some level 3 process elements of process category P2. In this case, the environmental model includes the variables – the delivery plan, and the sourcing plan – to be read and changed, respectively. The environment model for the above description is expressed as:

reads delivery plan

changes sourcing plan

In the case of real-world industrial settings, the MAS under development would either be independent and stand-alone or reside on top of some other enterprise information system (e.g., an Enterprise Resource Planning (ERP),

SCM system). It is important to note that environment model plays a crucial role and an analyst will have to provide a detailed and structured view of the environment. Note the important role that SCOR plays again. Without the inputs from the SCOR model it would be difficult to identify all such standard variables comprising environment for any generic system to be developed. For each process element at level 3, SCOR lists a set of standard inputs, and outputs. For example, in the case of the process category P2 “Plan Source” (or, role) that has a process element P2.1 “identify, prioritize, and aggregate product requirements” requires the inputs “item master, bill of material, product routings”. In all probability, this information would be part of an MRP (Materials Requirements Planning) or an ERP system that is outside of the MAS under development. In which case, this particular input has to be identified and become part of the environmental model. When the MAS is being implemented, the agent responsible for this particular role has to have an interface/sensor by which this information gets extracted. The environmental model comprises of inputs and outputs along with the type of actions (read, change, or consume) that a potential role/agent might perform on it. The environmental model for the system is a collection of all such variables and tuples for the entire system.

2.5.1.3 Preliminary Role Model

Gaia’s preliminary role model includes the identification of all the roles and their attributes for each of the sub-organizations. The Role model is a collection of all the schemas of the roles comprising the system. SCOR plays a major role

in the definition of the role model, offering standard process descriptions that would become roles in the MAS being developed. However, as SCOR is organized into different levels, the analyst will have to grapple with an important issue of the right level of process detail providing the best representation for the preliminary roles. It is obvious from the Gaia methodology that the roles are at a lower level description, dealing with tasks and specific skills. The process definitions of SCOR at level 1 – plan, source, make, deliver, and return – are at a macro level, ruling them out from being roles. Process categories at level 2 and the process elements at level 3 could potentially become the roles depending on the resolution of the system being developed. If the system under development is at an aggregated level, then level 2 process categories would provide sufficient details to be the roles. In the cases where the system has to focus on a much more detailed level modeling, level 3 process elements would be the best candidates to become the roles. The same logic holds true even with the enabling processes. It is important to emphasize, however, that since SCOR is a generic model covering a wide range of SC operations for different types of industries, it is comprehensive in nature. Not all process categories, process elements would be relevant or applicable for any given SC system. It is important to pick only the relevant and applicable generic elements/aspects based on the system requirements while developing a specific MASCS. To elaborate more on this aspect, let us assume that an aggregated level (level 2 process categories of SCOR are relevant) SC model is being developed, and a retailer is a sub-organization in the overall system. Since the retailer will not have any

manufacturing operations, all the process categories and their associated elements specific to “Make” process are irrelevant. Therefore, while modeling the retailer sub-organization, all of them should be ignored. Likewise, if the system requirements necessitate that a SC downstream of a manufacturer is of focus and not the upstream, while modeling the manufacturer sub-organization, one should ignore all the categories and elements relating to the “Source” process.

To illustrate the process of role definition in MASCF, the development of an example role schema from the SC domain using SCOR is presented here. The description of part of the system could be as follows: Let us assume that a SC system is being developed, comprising a manufacturer and a retailer, in order to understand its behavior at an aggregated level. The retailer places a weekly order on the manufacturer. The manufacturer receives the order and fills it from stocked inventory. The shipment quantity is derived as the minimum of the customer order and the available inventory. Let us try to define a role at the manufacturer’s end that fills retailer orders. Based on the description provided, process categories at level 2 would be adequate for the role description. As per SCOR, the role to be developed would be based on SCOR process category at level 2, D1: “Deliver Stocked Product”. From SCOR and the requirement description made available, the process elements: D1.2 “Receive Order”, D1.3 “Reserve Inventory”, and D1.10 “Ship Product” would describe the process sufficiently. So, these elements become the activities and interactions the role is involved with. “Reserve Inventory” is the activity that the role can perform on its own based on organizational policies and the inventory information available. It

may not involve any other role in performing the action but to book certain quantity of the inventory available in some system (either MRP or ERP). On the other hand, “Receive Order” involves the interaction with a corresponding role at the retailer’s end. Likewise, “Ship Product” would involve interaction with a corresponding role at the customer’s end. Therefore, they form the protocols. Now, with this available information the role schema can be defined.

Role Schema: **DELIVERSTOCKEDPRODUCT**

Description:
This primary role fills customer order for every time period .

Protocols and Activities:
ReceiveOrder , ReserveInventory , ShipProduct

Permissions:

	reads	*CustomerOrder*
		InventoryAvailability
	changes	*Shipment*

Responsibilities
Liveness:
DELIVERSTOCKEDPRODUCT = (ReceiveOrder.ReserveInventory .ShipProduct) maximum_number

Safety:
- *shipment = min(customerorder , inventoryavailability)*

Figure 2.5: Schema for SCOR-based role DELIVERSTOCKEDPRODUCT

As shown in Fig. 2.5, the role schema includes the following: the name of the role, a brief description of what the role does, and a list of activities and protocols. It also provides the permissions that the role has for fulfilling its responsibilities (executing roles and protocols). In this specific case, the role would need to have the permission to read customer order, inventory available, and change the variable shipment - the quantity of the order needs to be shipped to the customer. Notice that these permissions are based on the inputs and outputs of the level 3 process elements of SCOR. The role has certain

responsibilities. For example, to perform the activities and protocols: “Receive Order”, “Reserve Inventory”, and “Ship Product” in that specific order repeatedly over some specified maximum number of weeks. This forms the liveness expression of the role. While fulfilling the responsibility, it also has to take care of certain invariants in the form of safety. The safety property in this particular case says that the Shipment has to be the minimum of the two variables “Customer Order” and “Inventory Availability”. It is obvious from the above descriptions that SCOR plays a crucial role in precisely defining standard roles for SC modeling.

2.5.1.4 Preliminary Interaction Model

Interactions capture the dependencies between the various roles in the MAS organization. In identifying interactions between roles, the inputs and outputs of level 3 process elements of SCOR help a great deal. Let us assume that level 2 process category S1: “Source Stocked Product”, has been identified as a role. One of the outputs of the process is a “Procurement Signal” to the supplier. Let us assume that the corresponding supplier’s role (D1: “Deliver Stocked Product”) receives the signal and sends the shipment from its stock. This interaction can be named as, say “Procure Products”, with the first and the second roles being the initiator and partner respectively. It is also possible to define the input to the interaction to be the “Procurement Signal” and the output as “Shipment”, while providing an appropriate description to this interaction at the same time. This completes the definition of the interaction protocol “Procure Products” as indicated in Fig. 2.6. It is possible that during the process of this

interaction, the roles might exchange several messages to negotiate on the fill quantity. All such communication is covered under one single protocol definition. While Gaia views protocol as an institutionalized pattern of interaction, SCOR on the other hand defines the process in a standard way without explaining how it is implemented in practice. This example illustrates how best both Gaia and SCOR complement each other in the analysis of MASCSs.

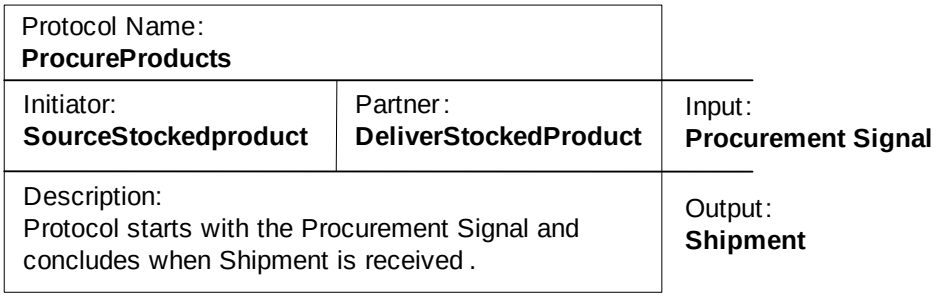


Figure 2.6: An example SCOR-based Interaction Protocol

2.5.1.5 Organizational Rules

Organizational rules and their correctness are fundamental to the design phase. Gaia considers organizational rules as responsibilities of the organization as a whole. They capture the general relationships between roles, between protocols, and between roles and protocols through liveness and safety rules. Liveness rules define the sequence of execution of the roles or protocols. Based on SCOR model (assuming level 3 process elements best represent the roles for the system under consideration) an agent dealing with planning manufacturing issues, for example, may have a couple of roles to play – “Balancing Production Resources With Production Requirements”, and “Establishing Production Plans.”

From the obvious considerations and also based on SCOR model, the “Balancing” role needs to be executed prior to the execution of the “Establishing Plans” role. Safety rules specify if a role has to be played by an entity, two roles can never be played by an entity, or any time-independent global invariants. They might also relate to the safety rules and environment variables of different roles.

2.5.2 Gaia-based Architectural Design in MASCF

SCOR plays a minor role in the architectural design phase concerned with the organizational structure, and completion of role and interaction model.

2.5.2.1 Organizational Structure

The choice of organizational structure is a very critical phase in MAS development; however, it is not possible to identify precise and formal methodology for obtaining the “best” design. Gaia provides guidelines for making choices regarding the topology and an appropriate control regime. The forces affecting this choice may include: the need to achieve organizational efficiency; the need to respect organizational rules; and the need to minimize the distance from the real-world organization. Gaia suggests that the decision on the size of an organization be based on the concept of “Bounded Rationality” - (the amount of information that an agent is able to store and process in a given amount of

time is limited). SCOR-based inputs may not play a role in affecting the organization structure decision.

2.5.2.2 Completion of Role and Interaction Models

Once the organizational structure is defined, the preliminary role and interaction models can be transformed into complete roles and interaction models. With regards to roles and protocols originating out of the organizational structure definitions, all the attributes will have to be specified. While completing the role and interaction models, it is important to differentiate the “intrinsic” (i.e., independent of the use of the role/protocol in a specific organizational structure) and the “extrinsic” (i.e., derive from the adoption of a specific organizational structure) characteristics. This distinction helps in the re-use and design for change. It also means that the changes in the organizational structure can be made without re-design and re-code agents from scratch. SCOR plays a minor role in these activities. Here is an example from the SC domain. Let us assume the designed organizational structure includes a role “Coordinator” that modifies procurement policy based on supplier performance. A role involved with procurement P2: “Plan Source”, has to be notified of this policy change in order for it to modify its “sourcing plan” specific to that supplier accordingly. Such situations introduce additional roles, activities, and interaction protocols. In the above example, one may have to define an activity based on EP.1 “manage business rules for plan processes.” Its output “Planning Decision Policies” forms an input to the role “Plan Source” as a part generating an output “Sourcing

Plans". As indicated, the information for defining the roles, interactions, and their attributes arising out of the organization structure could still come from SCOR.

2.5.3 Gaia-based Detailed Design in MASCF

The detailed design phase generates an agent model and the services model for the implementation of agents and their activities. Since the inputs to define agent and services model have already been identified in the analysis and architectural design phases, SCOR does not play any role in the detailed design phase. The successful completion of the entire process leads to the specification of MASCS that can be implemented using any multi-agent toolkit.

2.5.4 SCOR-based Supply Chain System Analysis in MASCF

MASCF also includes SCOR-based supply chain system analysis. This analysis involves taking the input in the form of system requirements and generating an output in the form of information related to a set of SCOR-based roles, interactions, environmental variables, rules and their associated attributes. If the inputs are in the form of generic or specific SC requirements, the associated outputs also will be either generic or specific, respectively. Having described in earlier sections how the models in various Gaia phases are supported by the inputs based on SCOR, we shift the focus here to summarize which specific elements of SCOR are relevant for which specific aspects of the Gaia models in the framework. Table 2.1 identifies some of the informational

requirements of the various Gaia models and how SCOR fills the needs. The table is self explanatory.

Table 2.1: Information-based Relationship between Gaia Models and SCOR Elements

Gaia Model	Aspects of Gaia / MASCF	Relevant SCOR elements
Organizations	Process Map	SCOR Process Map: Level 2 process categories
Environmental model	Variables & Tuples	Inputs and outputs of level 3 process elements – in particular information exchanged across inter-organizational boundaries (e.g., “Source” and “Deliver” process related)
Role model	Roles	Either level 2 process categories or level 3 process elements
	Activities	Level 3 process elements or their details
	Protocols	Level 3 process elements or their details
	Permissions	Inputs and outputs of level 3 process elements
	Liveness Responsibilities	Sequence of level 3 process elements
	Safety Responsibilities	Captured by the relationship between certain inputs and outputs of level 3 process elements
Interaction model	Initiator & Responder	Either level 2 process categories or level 3 process elements
	Inputs & Outputs	Inputs and outputs of level 3 process elements
Organizational Rules	Liveness rules	Sequence of either level 2 process categories or level 3 process elements
	Safety rules	Inputs and outputs of level 3 process elements (of other roles)

Whenever a MAS is to be developed, a SCOR process map can be generated from the requirements. It can be used as a useful starting point to identify relevant SCOR-based information and pass it onto the Gaia methodology within MASCF. Since SCOR is a comprehensive SC operations reference model,

for any given SC system requirements, it is highly unlikely that the entire model and its information gets used. We suggest the best way is to consider only those aspects that are relevant based on a given set of the requirements, unless of course the purpose is to build a comprehensive infrastructure for multi-agent SC modeling. In which case, the SCOR-based analysis can be made a one-time, offline activity. A library of roles, interactions, and other attributes can be created and stored in advance. Whenever a model has to be created, the user can put together a system specification using the pre-defined components. That forms an interesting and useful extension of this research. In the following section, we discuss the validation details of our framework using the “Tamagotchi” case.

2.6 Case Study

This section illustrates how MASCF can be applied in practice. We validate the framework using the Tamagotchi case influenced by real-world issues as outlined in Higuchi and Troutt (2004). The efficacy of the framework is established by actually implementing, using a multi-agent toolkit, the output generated by MASCF. A few results of the multi-agent simulation run are presented. Before doing so however, a brief introduction to the case is provided.

2.6.1 Introduction to Tamagotchi Case

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, Bandai 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 loses 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 life cycle product setting. We utilized the information provided in Higuchi and Troutt (2004) to setup our MAS. 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 using our SC process knowledge and industrial experience. In the absence of the actual requirements, the derived information was assumed to

be the system requirements and was used in the development of a 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.

2.6.2 Analysis & Design of Tamagotchi Supply Chain using MASCF

The objective of the first model of the analysis phase is to divide the Tamagotchi SC system into sub-organizations. It would be possible to consider the system-to-be as one single organization as a whole or consisting of multiple sub-systems depending on the purpose of model building. From the details of the model provided, it is appropriate to consider three sub-organizations corresponding to the three stages of the SC – the market, the retailer, and the manufacturer. This gives 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. Higuchi and Troutt (2004) considered the market level demand generation to be an aggregated level activity since; the purpose of the SC 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. The next step in MASCF is to carryout

SCOR-based supply chain system analysis based on the identified system requirements. With the decision made on the sub-organizations consisting Tamagotchi SC system, a SCOR process map (shown in Fig. 2.7) is generated for the entire system using the applicable SCOR level 2 process categories. The process map is self explanatory.

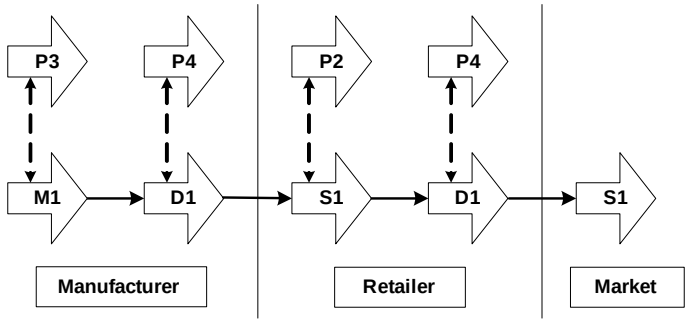


Figure 2.7: SCOR Process Map for Tamagotchi supply chain

Since the system being developed is at an aggregated level, process categories at level 2 provide the right level of resolution to be the roles in the system. Further analysis involved the identification of appropriate information required from the SCOR that would become input for the roles, interactions, and their attributes. A partial output of this step showing the list of potential roles is provided in Table 2.2.

Table 2.2: SCOR-based Roles for Tamagotchi supply chain system

Supply Chain Level	SCOR-based Roles
Market	▪ Source Stocked Product (S1)
Retailer	▪ Deliver Stocked Product (D1) ▪ Plan Deliver (P4) ▪ Manage Product Inventory (ES.4) ▪ Manage Finished Product Inventories (ED.4) ▪ Source Stocked Product (S1) ▪ Plan Source (P2)
Manufacturer	▪ Deliver Stocked Product (D1) ▪ Plan Deliver (P4) ▪ Manage In-Process Products (EM.4) ▪ Manage Finished Product Inventories (ED.4) ▪ Make Stocked Product (M1) ▪ Plan Make (P3)

Based on the nature of the system requirements, and also since the MAS under development being a stand-alone one, the environment model does not have any role to play. Accordingly, the development proceeded to the next step where the preliminary role and interaction models were defined. It is here the major portion of the information generated in the SCOR-based SC analysis is utilized. The entire role schema and the protocol definitions associated with the identified roles were completed based on the procedures elaborated in the previous sections. The next step is to identify the organizational rules. Organizational rules capture the general relationships between roles, between protocols, and between roles and protocols through liveness rules and safety rules. A few organizational (both liveness, and safety) rules identified for the Tamagotchi system are provided below as an example.

Liveness rules:

- “Source Stocked Product” precedes “Deliver Stocked Product”
- “Manage Finished Product Inventories” follows “Deliver Stocked Product”
- “Make Stocked Product” precedes “Manage In-Process Products”

Safety rules:

- “Manage Product Inventory” and “Manage Finished Product Inventory” should be handled by the same entity
- “Deliver Stocked Product” and “Source Stocked Product” should not be handled by the same entity

Having executed these steps, the analysis phase concludes and the architectural design phase starts with the preliminary role and interaction models, organizational rules and environment being the inputs. The first step here involves finalizing the organizational structure. Since the real-world details of the Bandai Corporation are not available, we only rely on the information that is available to us. We chose an organization structure based on collection-of-peers since the MAS is too small with very few roles and are diverse. With no additional roles or interactions identified, the role and interaction models are finalized and the architectural design phase is concluded. Due to space constraints, we present only an example role and interaction protocol of the Tamagotchi SC system in Fig. 2.8.

Role Schema: **PLANSOURCE (P2)**

Description:
This primary role prepare sourcing plan every week containing sourcing quantity (to be used by Retailer).

Protocols and Activities:
ReceiveDeliveryPlan , ReceiveInventoryAvailability , PrepareSourcingPlan

Permissions:

	reads	*Forecast*
		Inventory
	changes	*Sourcing Qty.*

Responsibilities
Liveness:
PLANSOURCE =
(ReceiveDeliveryPlan .ReceiveInventoryAvailability .PrepareSourcingplan)$^{maximum\_number}$

Safety:
- *sourcing qty. = max [(forecast – inventory), 0)]*

(a) An example Role Schema PLANSOURCE

Protocol Name: **Produce**		
Initiator: **PlanMake**	Partner: **Make-to-Stock**	Input: **Production Plan**
Description: Instruct the concerned partner to produce based on production plan .		Output: **Information Feed Back**

(b) An example Interaction Protocol

Figure 2.8: An example Role Schema and Interaction Protocol from the Tamagotchi case

Fig. 2.8(a) presents the role schema for a Tamagotchi role PlanSource at the retailer level. It performs an activity “Prepare Sourcing Plan” and interacts with other roles in “Receive Delivery Plan” and “Receive Inventory Availability” protocol. Fig. 2.8(b) presents an interaction protocol in the Tamagotchi case. It is initiated by the role “PlanMake” with the partner being the role “Make-to-Stock”. The whole purpose of this interaction is to share the production plan and receive an information feedback about the actual production. The detailed design phase receives the input in the form of the role and interaction models. We adopt the procedure as outlined in the Gaia methodology. The conclusion of the detailed

design phase generated an output of agent and services models. The identified agents and services for the Tamagotchi system are listed in Table 2.3. This concludes the MASCS analysis and design using MASCF.

Table 2.3: Agents and Services for Tamagotchi supply chain system

Agents	Services
Market Agent	▪ Place Market Demand
Retailer Sales Agent	▪ Fill Market Demand ▪ Establish Delivery Plans
Retailer Inventory Agent	▪ Update Inventory
Retailer Procurement Agent	▪ Source Products ▪ Establish Sourcing Plans
Manufacturer Sales Agent	▪ Fill Retailer Order ▪ Establish Delivery Plans
Manufacturer Inventory Agent	▪ Update Inventory
Manufacturer Production Agent	▪ Production ▪ Plan Production
Manufacturer Facility Agent	▪ Capacity Expansion

2.6.3 Implementation of Tamagotchi Supply Chain System & Results

The output of Tamagotchi SC system generated at the end of the analysis and design phases of MASCF can be implemented using any generic multi-agent framework. Java Agent DEvelopment Framework (JADE) is a Java-based FIPA-compliant (Foundation of Intelligent Physical Agents) middleware developed by TILAB (Telecom Italia Laboratories). It helps in the development of distributed multi-agent applications based on the peer-to-peer communication architecture (Bellifemine et al., 2001). Several industrial applications of JADE were recorded in diverse application areas including SCM, holonic manufacturing, rescue management, fleet management, auctions, tourism (Bellifemine et al., 2003). JADE is the most popular among all the open source toolkits available for implementing MASs. Because of its popularity and the functionality it offers,

JADE was selected as the implementation framework for the Tamagotchi case. The output agent and services models generated through the application of MASCF were implemented and several simulation experiments carried out. Information provided in Higuchi and Troutt (2004) was utilized to be the logic for the simulation model. A screen shot of the JADE model of the Tamagotchi SC system under execution is illustrated in Fig. 2.9. It shows various messages relating to interaction protocols getting exchanged between the agents comprising the system.

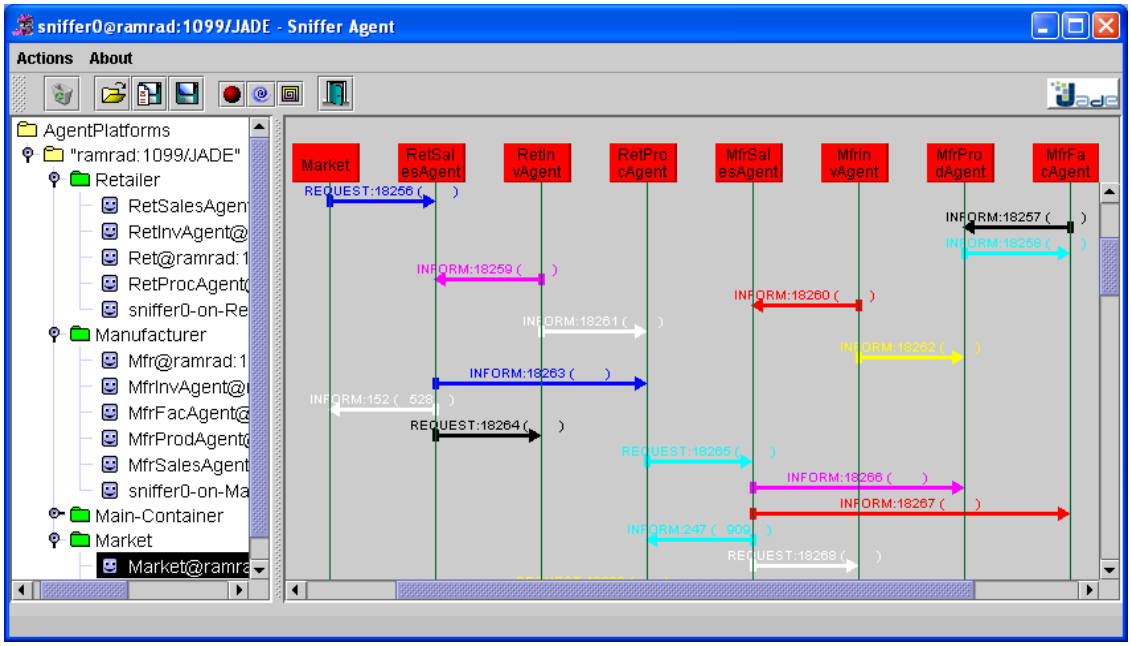
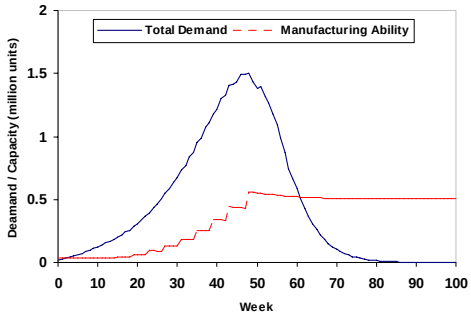


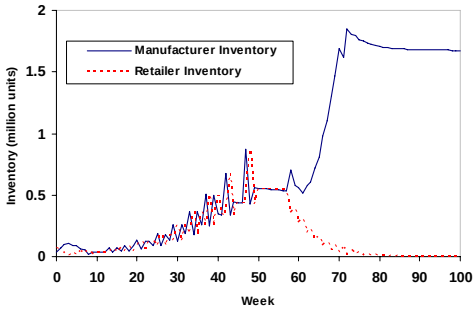
Figure 2.9: JADE screenshot showing message exchanges for a particular multi-agent simulation Run

Since not being the focus of this chapter, and due to space constraints, we avoid a detailed discussion on the multi-agent implementation of the Tamagotchi system using JADE. Instead we present partial results of a few multi-agent

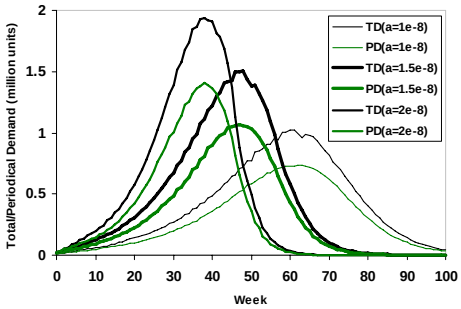
simulation runs in Fig. 2.10. Fig. 2.10(a) indicates boom and bust phenomenon that relates total demand and manufacturing capacity. The capacity had its peak enhanced by the over estimate of the demand. As the capacity increased to large levels, the demand dropped steeply leading to losses due to over investment. Fig. 2.10(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. 2.10(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 planning of very short life cycle products has to consider a lot of alternatives and options unlike the traditional stable demand products. Fig. 2.10(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).



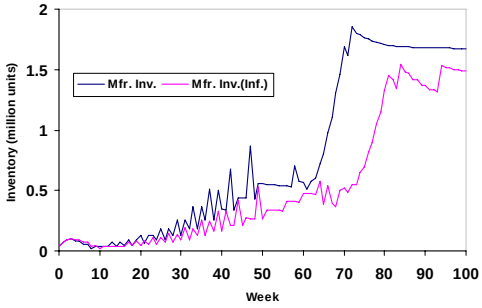
(a) Total Demand vs. Manufacturing Ability



(b) Inventory – Manufacturer vs. Retailer



(c) Impact of Diffusion Speed on Total & Periodical Demands



(d) Impact of Information Sharing on Manufacturer Inventory

Figure 2.10: Partial simulation results for Tamagotchi case

2.7 Conclusions and Research Extensions

MASs introduce a new paradigm for modeling complex systems such as SCs. 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 chapter 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 MASs. Although numerous methodologies exist for developing MASs, the support these methodologies offer is either at a too high level and abstract in nature, or they are too specific with the methodological process closely integrated and driven to a large extent by the capability of the implementation tool support. There exists no generic methodological support especially when it comes to developing MASCSs. Most of the MASs are based on some kind of metaphor with human-based organization being the most popular. However, organization metaphors currently being used are based more on function-oriented organizations. The chapter showed the evidence of process-centered organizations replacing function-based organizations. We introduced the notion of process-centered organization metaphor and argued that MASs, especially for SCs, be based on it. We proposed that a SC standard process-reference model, SCOR, can play a major role in the implementation of such metaphor. In order to improve the precision and efficiency of MASCS development, we proposed a generic process-centered methodological framework. Instead of developing it afresh, a generic yet comprehensive MAS development methodology, Gaia, and

the widely popular industry standard for process reference, SCOR, are integrated. This chapter provided the details of the framework, and showed how the framework operates with the help of SC domain examples. We validated the framework through the analysis and design of Tamagotchi MASCS. In order to demonstrate the effectiveness of the framework in practice, we implemented the output generated using JADE and presented the results. Having proved its utility, we are confident that the framework would serve its intended purpose well. Major contributions of the research work presented in this chapter can be summarized as:

- Development and validation of a generic methodological support for analysis and design of MASCSs, possibly the first methodology of its kind for MASCSs
- Creative adoption of SCOR to generic MAS development methodology, Gaia
- Introduction of the notion of process-centered organization metaphor in MASs

Finally, we conclude this chapter with an outline on the potential extensions of the research work. Integration of a generic process-reference framework is a crucial first step, towards automating some of the steps involved in the development of a MAS. The development of necessary infrastructure support in terms of building libraries of SCOR-based standardized roles and interactions has been identified as a logical extension to the research work carried out. Availability of such a library of reusable roles and interactions could

potentially support automation and speed up the process of MASCS analysis and design, In addition, one could consider integrating graphical model description support (e.g., UML-based) in the development of various models of MASCF. We argue that such a support is extremely crucial and could potentially facilitate automation of specification generation in order to speed up the development process of MASCSs even further. Finally since the focus of SCOR is only on SC operations, integration of similar standards that address strategic/tactical aspects of SCs would make the framework much more comprehensive in terms of standardization.

2.8 References

Abelson, H., D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Napal, E.R., G. Sussmann & R. Weiss. (2000). Amorphous computing. Communications of the ACM, 43 (5), 43–50.

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.

Angerhofer, B. J. & M.C. Angelides. (2000). System Dynamics Modeling in Supply Chain Management. In: J.A. Joines, R.R. Burton, K. Kang & P.A. Fishwick, Proceedings of the 2000 Winter Simulation Conference (pp. 342-351).

Anupindi, R. & Y. Bassok. (1999). Chapter 7: Supply contracts with quantity commitments and stochastic demand. In: S. Tayur, R. Ganeshan & M. Magazine (Eds.), Quantitative Models for Supply Chain Management, (pp. 197-232). Springer.

Arntzen, B.C., G.G. Brown, T.P. Harrison & L. Trafton. (1995). Global supply chain management at Digital Equipment Corporation. Interfaces, 25(1), 69-93.

Baatz, E. B. (1995). The Chain Gang. CIO Magazine, http://www.cio.com/archive/080195/supply_content.html

Bagchi, S., S.J. Buckley, M. Ettl & G.Y. Lin. (1998). Experience Using IBM Supply Chain Simulator. In: D.J. Medeiros, E.F. Watson, J.S. Carson & M.S. Manivannan, Proceedings of the 1998 Winter Simulation Conference, (pp. 1387-1394).

Beamon, B.M. (1998). Supply chain design and analysis: Models and methods. International Journal of Production Economics, 55, 281-294.

Bellifemine, F., G. Caire, A. Poggi & G. Rimassa. (2003). JADE - A white paper. TILAB "EXP in search of innovation" a special issue on JADE, 3(3), 6-19.

Bellifemine, F., A. Poggi & G. Rimassa. (2001). Developing multi agent systems with a FIPA-compliant agent framework. Software - Practice & Experience, 31(2), 103-128.

Bhaskaran, S. (1998). Simulation analysis of a manufacturing supply chain. Decision Sciences, 29(3), 633-657.

Biswas, S. & Y. Narahari. (2004). Object oriented modeling and decision support for supply chains. European Journal of operational Research, 153, 704-726.

Bresciani, P., A. Perini, P. Giorgini, F. Guinchiglia & J. Mylopoulos. (2004). Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems, 8, 203-236.

Cachon, G. P. (2003). Chapter 6: Supply chain coordination with contracts. In: S. Graves & T. de Kok (Eds.), Supply Chain Management – Design, Coordination and Operation, Handbooks in Operations Research and Management Science Volume 11, (pp. 229-340). Elsevier.

Cachon, G. P. & M.L. Fisher. (2000). Supply chain inventory management and the value of shared information. Management Science, 46(8), 1032-1048.

Candea, C., H. Hu, L. Locchi, D. Nardi & M. Piaggio. (2001). Coordination in multi-agent RoboCup teams. Robotics and Autonomous Systems, 36(2-3), 67-86.

Castro, J., M. Kolp & J. Mylapoulos. (2002). Towards requirements-driven information systems engineering: the Tropos project. Information Systems, 27, 365-389.

Cernuzzi, L., M. Cossentino & F. Zambonelli. (2005). Process models for agent-based development. In: Engineering Applications of Artificial Intelligence Conference.

Cernuzzi, L., T. Juan, L. Sterling & F. Zambonelli. (2004). Chapter 4: The Gaia Methodology. In: F. Bergenti, M. Gleizes & F. Zambonelli (Eds.), Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook, (pp. 69-88). Springer.

Chan, F.T.S., N.K.H. Tang, H.C.W. Lau & R.W.L. Ip. (2002). A simulation approach in supply chain management. Integrated Manufacturing Systems, 13(2), 117-122.

Chella, A., M. Cossentino, L. Sabatucci & V. Seidita. (2006). Agile PASSI: An agile process for designing agents. International Journal of Computer Systems Science & Engineering, Special issue on "Software Engineering for Multi-Agent Systems", In printing.

Chopra, S. & P. Meindl. (2006). Supply Chain Management: Strategy, Planning, and Operation. Prentice Hall, 3rd Edition.

Christopher, M. (1998). Logistics and supply chain management – strategies for reducing cost and improving service. 2nd ed., London et al.

Coleman, D., P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes & P. Jeremaes. (1994). Object-oriented development: The FUSION method. Prentice Hall Object-Oriented Series.

Cooper, M.C., D.M. Lambert & J.D. Pagh. (1997). Supply chain management: More than a new name for logistics. The International Journal of Logistics Management, 8(1), 1-14.

Cossentino, M. (2005). Chapter IV: From Requirements to Code with the PASSI Methodology. In: B. Henderson-Sellers & P. Giorgini, Agent-oriented methodologies, (pp. 79-106). Idea Group Publishing.

Cossentino, M. & L. Sabatucci. (2004). Agent system implementation. In: M. Paolucci & R. Sacile, (Eds.), Agent-based manufacturing and control systems: New agile manufacturing solutions for achieving peak performance, CRC Press.

Davenport, T.H. (1992). Process innovation: Reengineering work through information technology. Harvard Business School Press.

Davenport, T.H. (1995). Business process reengineering: Its past, present, and the possible future. Harvard Business School Note, 9-196-082.

Davenport, T.H. & J.E. Short. (1990). The new industrial engineering: Information technology and business process redesign. Sloan Management Review, Summer 1990, 11-27.

Deloach, S.A. (2005). Engineering organization-based multiagent systems. 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'05), St. Louis, MO.

Deloach, S.A., M. Kumar. (2005). Chapter IX: Multi-agent systems engineering: An overview and case study. In: B. Henderson-Sellers & P. Giorgini, Agent-oriented methodologies, (pp.236-276). Idea Group Publishing.

Deloach, S.A., M.F. Wood & C.H. Sparkman. (2001). Multiagent systems engineering. International Journal of Software Engineering and Knowledge Engineering, 11(3), 231-258.

Dorigo, M., E. Bonabeau & G. Theraulaz. (2000). Ant algorithms and stigmergy. Future Generation Computer Systems, 16 (8), 851-871.

Drucker, P.F. (1998). Management's new paradigms. Forbes, 162(7), http://www.forbes.com/forbes/1998/1005/6207152a.html

Fox, M.S., M. Barbuceanu & R. Teigen. (2000). Agent-Oriented supply chain management. The International Journal of Flexible Manufacturing Systems, 12, 165-188.

Gartner-Dataquest Report. (2005). User study: supply chain management initiatives, United States, 2004, L.M. Clark & C. Eschinger, ID Number: G00125217.

Giorgini, P., J. Mylapoulos & R. Sebastiani. (2005). Goal-oriented requirements analysis and reasoning in the Tropos methodology. Engineering Applications of Artificial Intelligence, 18, 159-171.

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.

Goetschalckx, M., C.J. Vidal & K. Dogan. (2002). Modeling and design of global logistics systems: A review of integrated strategic and tactical models and design algorithms. European Journal of Operational Research, 143, 1-18.

Govindu, R. & R.B. Chinnam. (2006). A software agent-component based framework for multi-agent supply chain modelling and simulation. (accepted for publication by: International Journal of Modelling and Simulation).

Hadeli, V., M. Paul, M. Kollingbaum & H. Van Brussel. (2004). Multi-agent coordination and control using stigmergy. Computers in Industry, 53 (1), 75-96.

Hammer, M. (1990). Reengineering work: don’t automate, obliterate. Harvard Business Review, July-August 1990, 2-8.

Hammer, M. (1997). Beyond reengineering: How the process-centered organization is changing our work and our lives. Collins.

Hammer, M. (2001). The superefficient company, Harvard Business Review, September 2001, 1-10.

Hammer, M. (2003). The agenda: What every business must do to dominate the decade. Three Rivers Press.

Hammer, M. & J. Champy. (1993). Reengineering the corporation: A manifesto for business revolution. HarperCollins.

Hammer, M. & S. Stanton (1999). How process enterprises really work. Harvard Business Review, November-December 1999, 1-9.

Hayes, C.C. (1999). Agents in a nutshell – A very brief introduction. IEEE Transactions on knowledge and Data Engineering, 11(1), 127-132.

Henderson-Sellers, B. & P. Giorgini. (2005). Agent-oriented methodologies. Idea Group Publishing.

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.

Holweg, M. & J. Bicheno. (2002). Supply chain simulation - a tool for education, enhancement and endeavour. International Journal of Production Economics, 78, 163-175.

Huang, Z., S.X. Li & V. Mahajan. (2002). An analysis of manufacturer-retailer supply chain coordination in cooperative advertising. Decision Sciences, 33(3), 469-494.

Jennings, N.R., K. Sycara & M. Wooldridge. (1998). A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems, 1(1), 7-38.

Jiao, J. (R.), X. You & A. Kumar. (2006). An agent-based framework for collaborative negotiation in the global manufacturing supply chain network. Robotics and Computer-Integrated Manufacturing, 22(3), 239-255.

Julka, N., R. Srinivasan & I. Karimi. (2002a). Agent-based supply chain management – 1: framework, Computers and Chemical Engineering, 26, 1755-1769.

Julka, N., I. Karimi & R. Srinivasan. (2002b). Agent-based supply chain management – 2: a refinery application. Computers and Chemical Engineering, 26, 1771-1781.

Karim, S. (2004). Experiences with Prometheus and the Prometheus Design Tool (PDT). http://web.dis.unimelb.edu.au/pgrad/mkarim/work/SimplePathPlanningAgent/Design_Reports/Experiences_with_Prometheus.pdf

Lambert, D.M., (Ed.). (2004). Supply chain management: processes, partnerships, performance. Sarasota, FL: Supply Chain Management Institute.

Lambert, D.M. & M.C. Cooper. (2000). Issues in supply chain management. Industrial Marketing Management, 29, 65-83.

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.

Lambert, D.M., S.J. García-Dastugue & K.L. Croxton. (2005). An evaluation of process-oriented supply chain management frameworks. Journal of Business Logistics, 26(1), 25-51.

Lee, H.L. & C. Billington. (1995). The Evolution of Supply-Chain-Management Models and Practice at Hewlett-Packard. Interfaces, 25(5), 42-63.

Lee, H.L. & S. Whang. (1999). Decentralized multi-echelon supply chains: Incentives and information. Management Science, 45(5), 633-640.

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.

Mamei, M., F. Zambonelli & L. Leonardi. (2003). Distributed motion coordination in co-fields. In: Proceedings of the 6th Symposium on Autonomous Decentralized Systems (Pisa, Italy, Apr.), IEEE Computer Society Press, Los Alamitos, Calif., (pp. 63–70).

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.

Odell, J.J. (2005). Foreword. In: B. Henderson-Sellers & P. Giorgini. (2005). Agent-oriented methodologies, (pp. vi-ix). Idea Group Publishing.

O'Malley, S.A. & S.A. DeLoach. (2001). Determining when to use an agent-software engineering paradigm. In: Proceedings of the Second International Workshop On Agent-Oriented Software Engineering (AOSE-2001), Montreal, Canada.

Padmos, J., B. Hubbard, T. Duczmal & S. Saidi. (1999). How i2 integrates simulation in supply chain optimization. In: P. A. Farrington, H. B. Nembhard, D.

T. Sturrock & G. W. Evans, (Eds.), Proceedings of the 1999 Winter Simulation Conference, (pp. 1350-1355).

Petrovic, D. (2001). Simulation of supply chain behaviour and performance in an uncertain environment. International Journal of Production Economics, 71, 429-438.

SCC. (2001). Supply-chain operations reference-model (SCOR) - Version 5.0. Supply Chain Council.

SCC. (2005). Supply-chain operations reference-model (SCOR) - Version 7.0 overview. Supply Chain Council.

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.

Sterman, J.D. (1989). Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment. Management Science, 35(3), 321-339.

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.

Taylor, D. A. (2003). Supply chain vs supply chain. Computer World, http://www.computerworld.com/printthis/2003/0,4814,86908,00.html

Terzi, S. & S. Cavalieri. (2004). Simulation in the supply chain context. Computers in Industry, 53, 3-16.

Veloso, M., P. Stone & K. Han. (1999). The CMUnited-97 robotic soccer team: Perception and multi-agent control. Robotics and Autonomous Systems, 29(2-3), 133-143.

Wagner, T., V. Guralnik & J. Phelps. (2003). TAEMS agents: enabling dynamic distributed supply chain management. Electronic Commerce Research and Applications, 2, 114-132.

Weiss, G., (Ed.). (1999). Multiagent systems: A modern approach to distributed artificial intelligence. The MIT Press.

Weiss, G. (2001). Agent orientation in software engineering. The knowledge engineering review, 16(4), 349-373.

Wooldridge, M. & N.R. Jennings. (1998). Pitfalls of agent-oriented development. Proceedings of the Second International Conference on Autonomous Agents (Agents 98), ACM Press, New York.

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.

Zambonelli, F. & H. Van Dyke Parunak. (2003). Signs of a revolution in computer science and software engineering. In: Societies in the Agents World III: Third International Workshop, ESAW 2002, P. Petta, R. Tolksdorf, F. Zambonelli (Eds.): LNCS vol. 2577, Springer-Verlag GmBH ISSN: 0302-9743 (pp. 13-28).

Zambonelli, F., N.R. Jennings & M. Wooldridge. (2001a). Organisational rules as an abstraction for the analysis and design of multi-agent systems. International Journal of Software Engineering and Knowledge Engineering, 11(3), 303-328.

Zambonelli, F., N. R. Jennings & M. Wooldridge. (2001b). Organizational abstractions for the analysis and design of multi-agent systems. In: Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering, LNCS vol. 1957, Springer-Verlag (pp. 253-252).

Zambonelli, F., N.R. Jennings & M. Wooldridge. (2003). Developing multiagent systems: The Gaia methodology, ACM Transactions on Software Engineering and Methodology, 12(3), 317-370.

Zambonelli, F., N.R. Jennings & M. Wooldridge. (2005). Chapter VI: Multi-agent systems as computational organizations: The Gaia methodology. In: B. Henderson-Sellers & P. Giorgini, Agent-oriented methodologies, (pp. 136-171). Idea Group Publishing.

CHAPTER 3

A SOFTWARE AGENT-COMPONENT BASED FRAMEWORK FOR MULTI-AGENT SUPPLY CHAIN MODELING AND SIMULATION

Multi-agent systems (MASs) have emerged as the most natural intrinsic paradigm for modeling complex systems, such as supply chains (SCs). However, their development process remains quite involved and extremely time consuming, hindering wide-spread adoption in industrial-strength applications. A software agent-component based framework is proposed to simplify and speedup multi-agent (MA) modeling and simulation for SC applications. With the help of pre-developed libraries of reusable components – organizational agents (OAs), SC agents (SCAs), behaviour and policy objects; the framework allows model developers to quickly configure a MAS in order to simulate SC dynamics and study control and coordination issues. Being generic, flexible, and scalable; it supports development of either pseudo-centralized models by a single model developer, or distributed models by either a single or group of enterprises constituting a SC. The framework is unique; based on requirements it allows for representing different segments of SC network at either aggregated or detailed levels resulting in models of hybrid resolution. It facilitates studies involving intra- and inter-organizational dynamics (either independently or collectively), considering information asymmetry explicitly. The framework is validated through MA-simulation of Tamagotchi SC. Its results are presented, and the research extensions outlined.

3.1 Introduction

Efficient management of SCs has become an essential enabler providing competitive advantage to enterprises. With decades of relentless pursuit making intra-organizational processes efficient, enterprises are realizing further substantial benefits through the integration of inter-organizational SC interfaces. Individual firms in the market place today no longer compete as independent enterprises but rather as integral parts of SCs ([1], [2]). As more and more firms embrace efficient supply chain management (SCM) practices, such initiatives are becoming a fundamental pre-requisite for survival in lieu of offering any significant competitive leverage. This in turn is compelling enterprises to continuously innovate, and implement creative practices (both without and with the help of latest technologies) in the business processes of their SCs for achieving substantial benefits. MASs (evolved as one of the two primary areas of distributed artificial intelligence (DAI) [3]), are becoming increasingly popular due to their inherent ability to model the distributed and autonomous features (including information asymmetry) of various entities constituting a SC in the most natural and realistic way. They have the ability to integrate within the well known decision modeling techniques (e.g., simulation, optimization, …) and diverse fields of knowledge (e.g., game theory, computational intelligence, behavioral sciences, cognition …) leading to potentially the most powerful modeling paradigm. But what exactly is MAS or an “agent” for that matter? Literature offers numerous definitions and we present a few here.

An agent is a computational entity that perceives, acts upon its environment, and is autonomous in its behaviour [4]. It is a computer system capable of flexible autonomous action in dynamic, unpredictable, typically multi-agent domains [5]. In a generic sense, an agent is capable of carrying out goals and has two key properties [6]: partial autonomy; and part of a community in which mutual influence occurs. MAS, as its name indicates, is a software system of multiple interacting agents [7] that coordinate, cooperate and may even compete. There are at least a couple of reasons for why MASs are becoming increasingly popular [4]: modern computing and information environments are distributed, large, open, and heterogeneous; and they have the capability to model theories of interconnectivity in human and artificial societies. The characteristic features [8] of such MASs are: each agent has incomplete information, capabilities, thus a limited viewpoint; there is no global system control; data is decentralized; and computation is asynchronous. In terms of their application potential, MASs (as is obvious from the above features) are best suited and hold a great promise for modeling complex and adaptive real-world systems, in particular, SCs. For more than a decade, MASs are projected by researchers to be the next generation modeling paradigm. However, the process of MAS development still remains quite involved and extremely time consuming. Despite a few a real-world commercial applications in the recent past (refer [9], [10]), MAS implementations (multi-agent supply chain systems (MASCSs) included) remain confined with researchers, technology enthusiasts and visionaries; needing a major thrust for their mass market adoption to become a

reality [11]. A few important reasons for this state of affairs include: still “evolving” standards, “work-in-process” development toolkits, and lack of “mature” development methodologies.

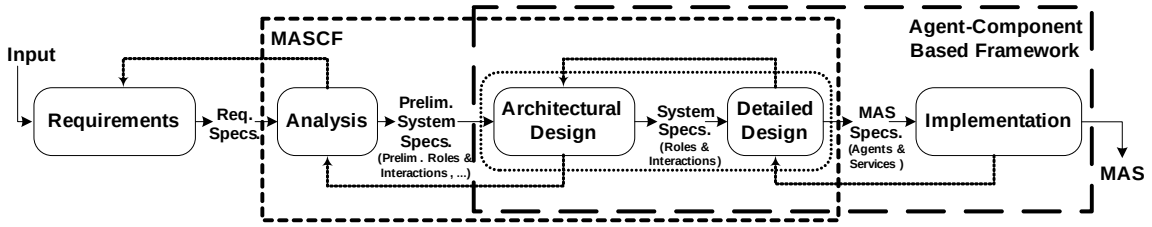


Figure 3.1: Phases in multi-agent system development – focus of software agent-component based framework

Our research focuses on simplifying MASCS development as an effective and efficient strategy, in order to realize its wide-spread adoption for the both research and industrial applications. As illustrated in Fig. 3.1, the process of MAS development (like any other software development) involves the phases of requirements analysis, system analysis, design (architectural & detailed), and implementation; applied sequentially and iteratively as needed. The development of a generic methodological framework, Multi-Agent Supply Chain Framework (MASCF) was discussed in detail in [12]. MASCF creatively adopted Supply-Chain Operations Reference-model (*SCOR*) [13] with *Gaia* methodology [14] focusing on the system analysis and design phases of development. In contrast, we developed a software agent-component based framework (complimenting MASCF) that focuses on the design and implementation phases. This chapter presents the details of the framework, its implementation, and how to model MASCSs using its components. In order to validate the framework, Tamagotchi

SC (as described in [15]) is modeled, its dynamics simulated, and the results presented. Finally, conclusions and summary are provided along with some of the research extensions being pursued. Before presenting those details, however, a brief review of the pertinent literature is provided in the following section.

3.2 Multi-Agent Supply Chain Modeling

Researchers have been exploring MASs in order to better model various SC problems. A brief review of the literature pertaining to MA-modeling as applied to SCM is presented here along with a discussion on the research gaps. In perhaps one of the earliest research papers that discussed reusable components for SC modeling, [16] investigated the construction of intelligent agent-based software architecture for managing SCs at the tactical and operational levels. The development of an “agent building shell (ABS)” that provides generic, reusable components and services to support cooperative work perturbed by stochastic events was discussed. Its focus to a large extent was confined to the agent architecture and communication framework. An overview of a reconfigurable, multi-level, agent-based planning and scheduling architecture, MASCOT (multi-agent supply chain coordination tool) was detailed in [17]. Key architectural elements of the system, real-time support for finite capacity scheduling, and the development of new coordination protocols were discussed. How TAEMS (task analysis, environment modeling and simulation) framework-agents automate and manage a distributed dynamic SC with the help of

coordination mechanisms was the focus for [18]. The objective was to enable SCs to be more flexible and responsive through producer/consumer negotiation and reasoning. Reengineering the order fulfillment process (OFP) in supply chain networks (SCN) through multi-agent information system (MAIS) approach was presented in [19]. An enhanced version of SWARM (a MA-simulation platform) was utilized to identify and evaluate potential improvement strategies through simulated experiments. Quick development of SC simulation models of appropriate resolution through MAS framework was the fundamental idea for [20]; one of the most often cited papers. Its approach involved composing models from a library of reusable, domain specific software components representing SC agents, their interaction protocols, and control element objects. A cross-docking prototype and a full-scale application at IBM were mentioned but, neither the results nor the complete system details were presented. In the context of SC integration in a major enterprise, [21] discussed how intelligent SC agents conduct business on behalf of product users, buyers and vendors. The SC process design, its structure, and agent federation behaviour were implemented using Agent Development Environment (ADE) on an expert system shell, G2. MA-modeling techniques to simulate, control, and optimize the manufacturing component of a simple demand-driven SC network system was focused in [22]. A framework to represent various elements of a SC in a unified, intelligent, and an object-oriented fashion to model, monitor, manage, and help analyze business policies was presented in [23]. Implemented using ADE on G2 its applicability was demonstrated on a prototype decision support system to study the effects of

internal policies, exogenous events and plant modifications in a petroleum refinery. JADE implementation of an agent-based multi-contract negotiation to achieve global SC manufacturing coordination in a mobile phone company was the focus for [24]. A flexible prototype agent system that adapts to transaction changes introduced by new products and trading partners for a computer manufacturing SC application was detailed in [25].

Although MASs have been in existence for over a decade, there exist hardly any references of industrial-strength SC applications. Except for a few papers (e.g., [16], [20]) that discussed the importance of developing generic components to promote reusability, most of the literature seemed to offer specific solutions to particular modeling problems. Although architectural issues were discussed, focus for articles like [17] and [18] had been on specific aspects like real-time coordination and scheduling. It is evident from the literature in general, that MASs are implemented directly from pre-stated requirements with very little focus explicitly on the system analysis, design, and implementation (the most important aspects for industrial-strength applications) in an integrated manner. Developing specific techniques tailored to MASs is necessary in order to realize their potential as a software engineering paradigm [26]. In order for MAS applications to flourish, it is essential to develop generic methodologies and toolkit-based implementation frameworks. Clearly defining, simplifying, and speeding up the development process is crucial for widespread industrial adoption of the MAS paradigm, be it for SC modeling or otherwise. The focus of our research has, accordingly, been on how to simplify MAS development for SC

modeling and decision support. One of its outcomes is the development of a *software agent-component* based framework, the central theme for this chapter. Before its details are presented, a brief discussion on MA-development toolkits is provided here.

An interesting aspect is that while agent-based systems are becoming increasingly well understood, the development of MASs is not [27]. Researchers are emphasizing on devoting more effort to understand the pragmatics and the reality of development. Literature indicates the usage of programming languages (e.g., Java, C++ …), commercial software (e.g., JACK, AgentBuilder …), and open-source toolkits (e.g., Zeus, JADE …) for MAS development. With the existence of a review [28] of thirty-six software products, and more than hundred registered agent-related software [5]; the lack of modeling/development tools certainly cannot be the cited reason for very few MAS applications (industrial or otherwise). We emphasize that, the toolkits issues are more attributable to their limited functionality, lack of maturity, unavailability of detailed documentation and methodology support, and lack of graphical user interface (GUI) support for model specification and development. Better functionality and more maturity are beginning to appear in at least some of the toolkits leading to their wide-spread usage and in some cases even in commercial applications. A few popular toolkits of that category include: JADE among the open-source; Agent Builder and JACK among the commercial software. Developed by Telecom Italia Lab (TILab), JADE is a software development framework that simplifies MAS implementation through a platform-independent, FIPA (Foundation for Intelligent Physical

Agents) standards-compliant middleware [29, 30]. It includes an agent platform that allows MAS to be distributed on multiple machines, a package to develop Java agents, and a set of graphical tools that offer debugging and runtime visualization support. JADE, however, has the following limitations: it is a low level programming environment with limited documentation, not easy (for a “typical user”) to quickly learn to do programming, and offers no GUI support for model specification and development. Despite these limitations, based on its superior functionality and ease of extensibility, JADE has become the most popular among open-source toolkits and is being used in commercial applications as well. Numerous plug-ins/extensions being developed to improve upon the “basic” JADE functionality indicates to its popularity and effectiveness. It is precisely for such important reasons JADE is adopted in our framework and research.

3.3 Software Agent-Component Based Framework

This section presents the details of the framework, its implementation, how to configure MASCSs using its component libraries, and summarizes its characteristic features. The framework by design helps explicit study of SC dynamics (both intra- and inter-organizational) involving Bullwhip-effect [31, 32] and Boom-and-Bust phenomenon [33]; and coordination issues in any SC, considering information asymmetry (due to its being public and private). As Fig. 3.2 indicates, it consists of two layers – an *infrastructure layer* and a *modeling layer*. The infrastructure layer integrates multiple software tools; playing the

crucial role of a development platform, and offering debugging and runtime support. Modeling layer, on the other hand, consists of component libraries pre-developed with the help of the infrastructure layer; and simplifies the process of MASCS development reducing it to a matter of quick configuration.

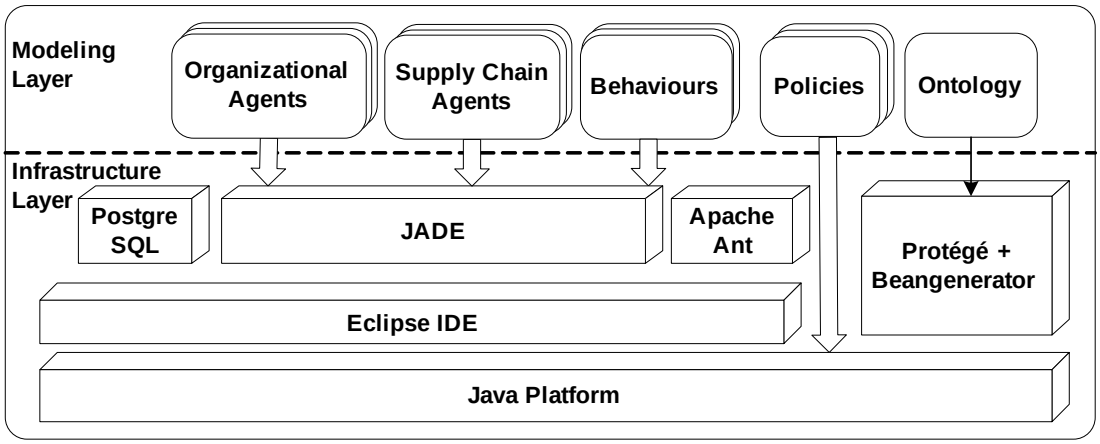


Figure 3.2: Architecture of software agent-component based framework

3.3.1 Infrastructure Layer

As illustrated in Fig. 3.2, the infrastructure layer by design includes the *Java* platform of Sun Microsystems providing the basic foundation on which the proposed framework operates. As multiple toolkits/software had to be integrated to make the framework powerful, *Eclipse* was chosen to be the integrated development environment (IDE). Eclipse is an extensible open source Java-based IDE using which multiple toolkits and software can be seamlessly integrated in order to simplify application development. *Apache Ant*, a Java-based build tool and programming utility, is utilized for compiling Java code and building new version of the MAS software whenever the source files are updated. *JADE* was adopted to be the MA-development toolkit of the framework. It

provides pre-developed artifacts (e.g., agent classes, behaviour classes, standard template objects and methods …) that can be easily extended, simplifying MAS development and promoting reusability. One of the fundamental characteristics of any MAS is that individual agents communicate and interact through message exchanges. JADE supports the usage of three alternate approaches to agent communication – String representation of the message content; Java serialized objects; and Ontology objects. Ontology objects can be transferred as extensions of predefined classes that JADE agents can automatically use to encode and decode messages in a standard FIPA format. Communicating with the help of domain specific ontology classes, that are based on FIPA standards of Agent Communication Language (ACL) and Semantic Language (SL), is the most powerful of the three modes. Such an approach simplifies, standardizes, and ensures that communication happens not only among JADE agents but also, between JADE agents and agents developed using any other FIPA-compliant code/development platform (e.g., Zeus, FIPA-OS …) promoting interoperability. Most importantly, it can ensure agent interactions at a semantic-level rather than just pure syntactic-level. Ontology-based agent communication is adopted in the framework due to all such beneficial reasons and for its development, *Protégé* (the most popular Java-based extensible ontology editor and knowledge-base framework) is utilized with an added plug-in tool, *Beangenerator*. SC domain ontology is defined using the GUI of Protégé with the help of JADE abstract ontology template of the Beangenerator. Once the domain ontology is defined, the Beangenerator tool helps to automatically

generate the java code representing FIPA-compliant ontology that JADE agents can utilize for communication and coordination at runtime. This approach eliminates the need for carefully programming the ontology code in JADE acceptable FIPA-format and reduces the development effort substantially. The remaining component of the infrastructure layer is the database software, *PostgreSQL*, integrated into the framework using JDBC technology. PostgreSQL (one of the most advanced open-source relational database systems), helps in persisting at runtime the values of the agents' "knowledge-base" variables. It relieves the developer of having to deal with output storage issues (like defining flat-file structures), and allows imports into desirable file-formats (e.g., excel) for further analysis by permitted decision makers. Most importantly, it can provide the agents with data "access" (based on permissions) to facilitate knowledge acquisition through agent intelligence and reasoning. The database segment would play even more important role in the future extensions of the framework that incorporates rule/data-mining based learning and Belief-Desire-Intention (BDI) reasoning based rational agent architecture.

3.3.2 Modeling Layer

The modeling layer helps to quickly configure any given SC simplifying the process of MASCS development. It includes, by design, reusable SC domain ontology and libraries of reusable and extensible components (OAs, SCAs, behaviour objects, and policy objects) pre-developed with the help of infrastructure layer. The framework utilizes for development – *JADE* for agent-

components and behaviour objects, *Java* for policy objects, and *Protégé* with *Beangenerator* for SC domain ontology. While this section presents the details of the component libraries, in order to have a better focus, SC domain ontology is discussed in a later section.

3.3.2.1 Agents

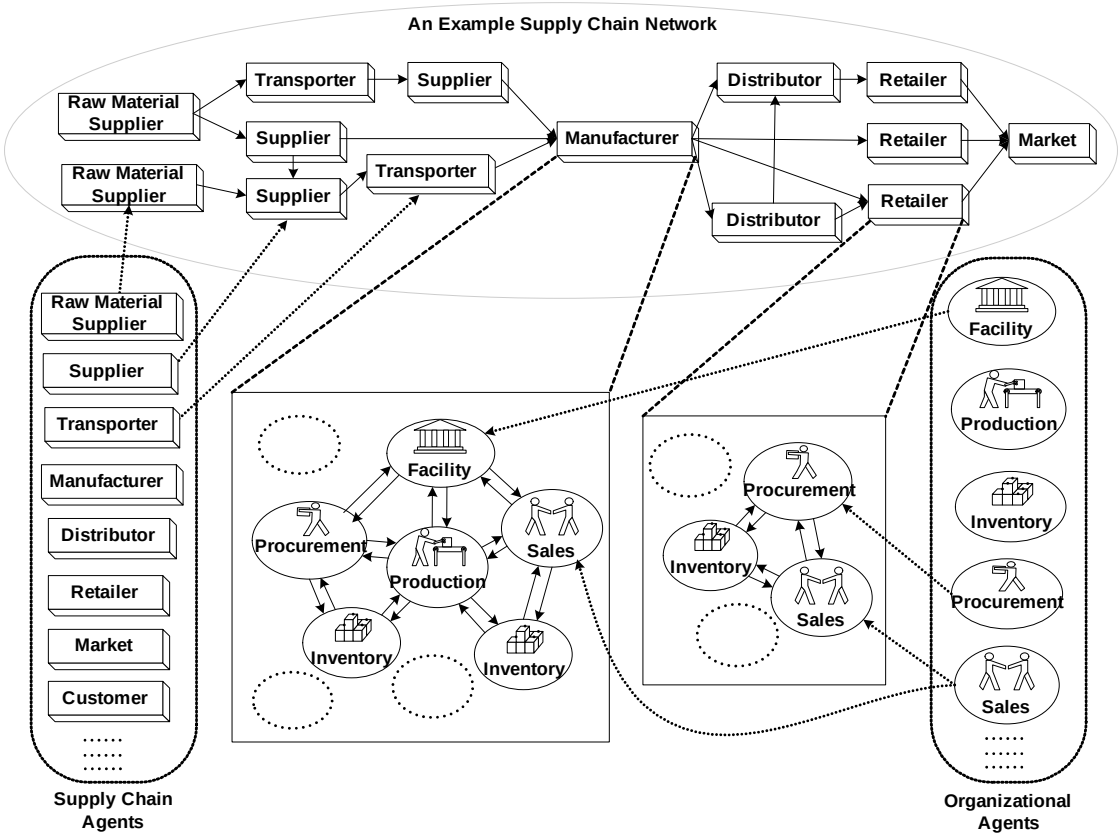


Figure 3.3: An example supply chain configuration through reusable agents of the framework

The framework defines two types of agents – *Supply Chain Agents* (SCAs) and *Organizational Agents* (OAs). It is obvious that any real-world SC network consists of multiple interacting entities (independent enterprises) at

different levels/tiers (e.g., retailer, supplier, manufacturer …). The set of agents representing all such SC entities constitute the library of SCAs (refer Fig. 3.3). While it is not necessary for any given SC model to include the entire set of SCAs; however, most of the SCs can be defined using the elements of this set. In general, SC modeling is taken up at an aggregated or a detailed level. The right level of model resolution for a given situation would depend on its nature and intended purpose (strategic, tactical, or operational). From a modeling standpoint, any SCA would exist in a SC model irrespective of its resolution – aggregated or detailed. One exception to the above logic relates to the ultimate customer or market SC entity. Consider a particular SC network having a market entity representing the aggregated behaviour of all the possible customers for a product/s. The market agent representation would be sufficient, if the model being developed is at an aggregated level rather than detailed. For models involving strategic and tactical level issues, such a representation would be most appropriate. Studies involving operational level and real-time issues would however necessitate a much more detailed representation. In which case, the ultimate customers may have to be represented explicitly for an aggregated representation in the form of market entity would be insufficient and may indeed be undesirable. Accordingly, both market and customer agents were included to be the SCAs to widen the framework scope. In the real-world, each of the SC entities in turn, would consist of multiple functions. For example, a retailer SC entity would have a sales function, an inventory function, a procurement function … etc. A manufacturer SC entity on the other hand, would in addition have a

production function, facility function … etc. The set of all such agents representing organizational functions (within the SC entities) constitute the library of OAs (refer Fig. 3.3). More SCAs and OAs could be added, as indicated in Fig. 3.3, to their respective sets as needed. Having introduced various agent-components of the framework, we elaborate on their reusability aspects now.

The framework by design conceives OAs to be reused in multiple SC entities represented by SCAs – e.g., an OA “sales agent” would exist inside the “retailer”, “manufacturer”, and “supplier” SC entities. At a conceptual level, the basic functionality of the “sales agent” remains the same irrespective of to which SC entity it belongs to. For example, a sales agent receives customer orders and fills them based on certain pre-defined order filling policy, forecasts the demand for a future time period and informs the other concerned OAs about it, and so on and so forth. This functionality of the “sales agent” remains the same conceptually whether the agent belongs to the “manufacturer”, “retailer”, or the “supplier” SC entity. The agents with whom all this particular “sales agent” corresponds with could be different depending on to which SC entity it belongs to. Such issues within this framework are taken care of by the list of agent “acquaintances” (instantiated during configuration), but it doesn’t make a “sales agent” under one SC entity drastically different from another. Likewise, any SC model would potentially include multiple SC entities (represented by SCAs) at each level. For example, when a “retailer” SCA is defined, it is possible to conceive multiple retailers to be the same conceptually, hence can be represented using the same SCA “retailer”. The internal logic could vary

somewhat (may use a different behaviour and a policy based on instantiation – however, all of them are already referenced inside the agent code) but, the basic agent structure would remain the same. It is such logic that forms the basis for defining reusable agents – SCAs or OAs.

A unique feature of the framework is its ability to generate SC models of hybrid (some network segments at detailed and others aggregated) resolutions. To attain this functionality, every SCA (except market and customer) is enabled (through instantiation) with the ability to configure its model segment at the desired resolution. If an aggregated option is chosen for a particular SCA, then only the SCA will be created and not the OAs within. If instead the detailed option is chosen, then not only the SCA is created but also either all or some of the OAs within that SC entity. This partial selection option would be extremely useful if for some reason a manufacturer SC entity wants to focus only on the downstream operations but not upstream, then, perhaps, procurement OA may not be necessary and hence need not be created. The dotted oval shapes of Fig. 3.3 symbolically represent all such OAs not meant to be created within the SC model. The figure illustrates that three OAs of the manufacturer SC entity and two of the retailer SC entity are not created and, therefore, do not get included in the model. Depending on the MAS requirements, all such options are instantiated and the “acquaintances” lists of the agents are re-adjusted through configuration. This particular feature is extremely powerful given the fact that the framework is generic, applicable for a wide-range of SC modeling applications; and can be

used by a single model developer, single enterprise, or group of enterprises in a SC.

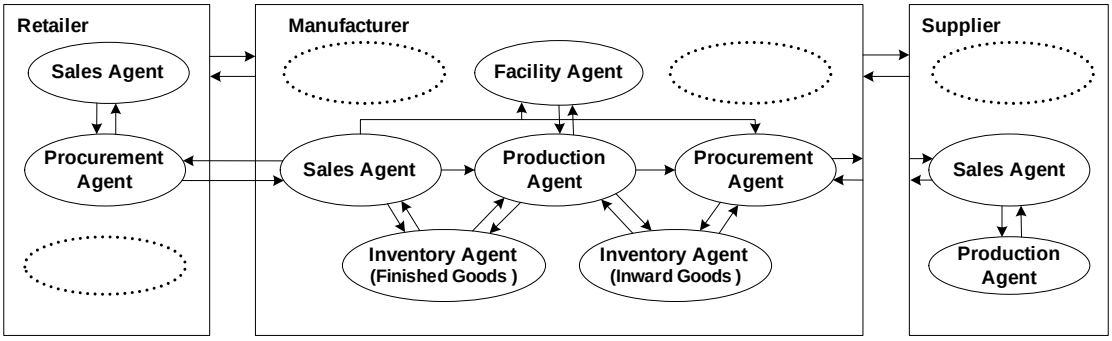


Figure 3.4: Framework supports both direct and indirect communication between agents

We emphasize here that by design, both SCAs and OAs are only generic agent shells with communication abilities. JADE supports a *direct peer-to-peer communication* architecture with any peer allowed to play the roles of both the initiator and responder. However, keeping in mind the real-world SC modeling requirements, the proposed framework provides both *direct* (peer-to-peer), and *indirect* (routed through the corresponding SCAs) modes of communication. Communication between two SCAs (being peers) shall remain peer-to-peer irrespective of the model being aggregated or detailed. The communication between two OAs within the same SC entity shall obviously remain peer-to-peer. However, the communication between two OAs that belong to two different SC entities will have both the direct and indirect options with one of them chosen at the time of configuration. Fig. 3.4 illustrates such aspects graphically. The communication between the OAs within the retailer, manufacturer, and supplier SC entities; between the SCAs; between the Procurement OA of the retailer SC

entity and the Sales OA of the manufacturer SC entity is all peer-to-peer. However, the communication between the Procurement OA of the manufacturer SC entity and the Sales OA of the supplier SC entity is routed through their respective SCAs. Such flexibility in the communication structure is realized through instantiation of the lists of “acquaintances” of the agents through model configuration.

```
public class Retailer extends Agent implements SupplyChainParameters
{ ……. ..
  private AgentContainer container  = null;
  private AgentController controller  = null;
  private AID[ ] acquaintances ;
protected void setup ( ) {……..
  // initialize parameters
  … .
  // if modelling aggregated
      // add aggregated behaviours … .
  // if modelling detailed
      // add  createOA behaviour … .
  // add Coordination behaviour
  // add  Write to Database behaviour   … . }

class createOA extends OneShotBehaviour   {...
  try {   // get a container controller for creating new agents
  container = (AgentContainer)getContainerController ( );
……
   // create organizational agents
    controller = container.createNewAgent (sales, …….);
    AgentController cr = container.createNewAgent (Procurement, ….);
…….}
```

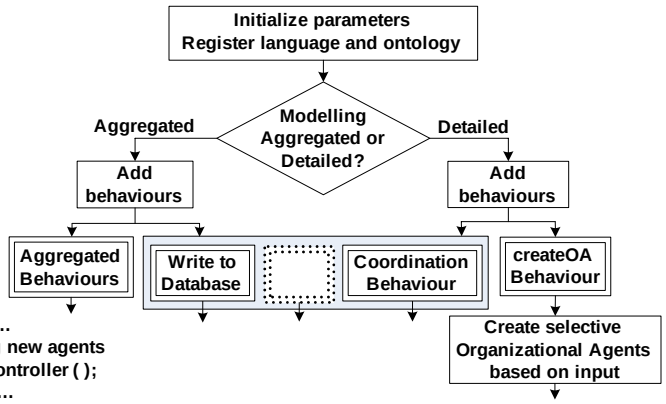


Figure 3.5: Partial pseudo-code and graphical representation for “Retailer” SCA

The agents in the framework are implemented by simply extending the “Agent” class that JADE provides. For a detailed conceptual model of the JADE “Agent” class, readers are suggested to refer [34]. In order to better explain the structure of agent-components, we provide here a couple of example partial pseudo-codes and their corresponding graphical representations. Fig. 3.5 illustrates an example SCA “Retailer”, which is an extension of the JADE “Agent” class and gets its parameter values from the file “SupplyChainParameters”. The first few lines of the agent code correspond to the initialization of parameters, and

registration of ontology and language. Based on the model resolution, appropriate behaviours (figure explicitly indicates only a few) are added to the agent queue and scheduled for execution at runtime based on non pre-emptive cooperative round-robin logic. If a detailed option is chosen then this SCA "Retailer" creates the required OAs (based on the corresponding input).

```
public class ProcurementAgent extends Agent implements SupplyChainParameters   {
…….. ..
   //  Set Codec & Ontology
   private Codec codec  = new SLCodec( );
   private Ontology ontology  = SupplyChainOntology.getInstance( );

protected void setup ( ) { ……..
   // Initialize agent parameters
   // register language and ontology to agent's content manager
   // Add Agent's initial Behaviours …… .
   addBehaviour(new ReceiveMessages (this));
   addBehaviour(new WriteToDatabase (…));
   ……..…….}
   // Invoke this method when there is change in the state of the behaviour  .
public void notifyChangeBehaviourState (Behaviour b, String from, String to)
 {       …….
          sendOrder( );
……..}
class ReceiveMessages extends CyclicBehaviour   {…….
    // Choose necessary action based on message type
          // If Inventory msg
                addBehaviour(new ReceiveInventory (…));
          // If Forecast msg
                addBehaviour(new ReceiveForecast (…));
          // if Shipment msg
                addBehaviour(new ReceiveShipment (…));
                    // Update inventory Agent
…… }
  void sendOrder ( ) {…….
       Order order = new Order ( );
       FillOrder fillOrder = new FillOrder( );
       fillOrder.setOrder (order); ……..
       //  Send Order message to Vendor Agent
          send(msg1);
…….}
```

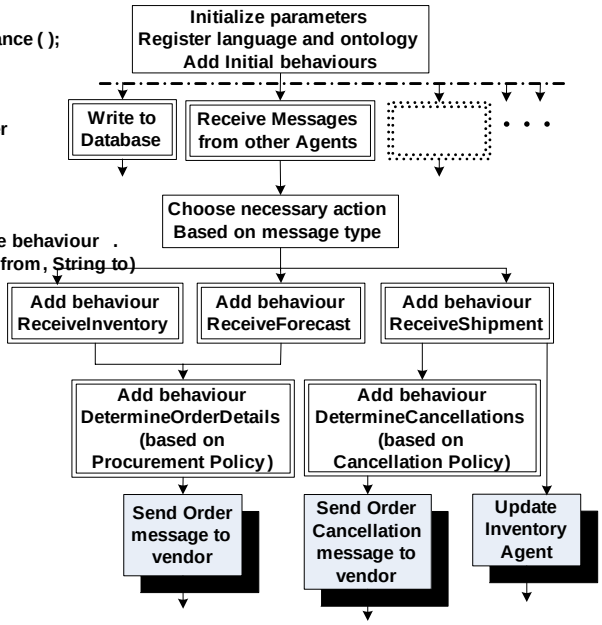


Figure 3.6: Partial pseudo-code and graphical representation for "Procurement" OA

Fig. 3.6 illustrates an example OA, "Procurement Agent" which is again an extension of the JADE "Agent" class and gets its parameter values from the file "SupplyChainParameters". The initial few lines of the agent code correspond to the initialization of parameters, and registration of ontology and language. All the appropriate behaviours (figure explicitly indicates only a few) are added to the

agent queue and scheduled for execution at runtime based on non pre-emptive cooperative round-robin logic. The “receive messages” behaviour receives messages addressed to “Procurement Agent” and based on its type, adds subsequent behaviours (that may in turn, reference appropriate policies – e.g., procurement policy). The “send order” method includes the code for composing and sending an “order” message to corresponding vendor agents. Based on the number of orders to be sent to vendors, this method may be iterated multiple times.

3.3.2.2 Behaviour Objects

Agents of any MAS are conceived to offer certain services. For all practical purposes, every agent performs multiple services simultaneously leading to the requirement of multi-threaded agents. Although, JADE agents can be programmed to be multi-threaded using Java, such an approach is not recommended due to efficiency considerations and development complexities involved [35]. JADE provides a set of “behaviour” templates using which model builders implement an agent service or a portion of it so that it consumes only a small execution time whenever called. Hence, every service is implemented by extending either one or multiple behaviours in practice. Multiple behaviours of an agent enter its queue at runtime and their execution is scheduled based on non pre-emptive cooperative round-robin logic. Although this most-widely adopted practice calls for extra programming effort; it helps circumvent the multi-threaded development complexities yet mimic its outcome, while retaining execution

efficiencies. The list of behaviours (refer [34]) that JADE provides range from simple to composite with several types within each category including Finite-State-Machine (FSM).

The framework defines generic behaviours that can be reused across multiple agents (SCAs and OAs) within a SC. For example, an inventory OA may offer the service of informing inventory levels at certain time periods to the other agents (e.g., sales OA, production OA, procurement OA …) needing that information. Let us assume that this service is implemented through “Inform Inventory” behaviour. The process logic of this particular behaviour remains the same irrespective of whether the inventory agent is in a manufacturer SC entity or a supplier SC entity. What could possibly change from one inventory OA to another are: how often inventory levels are informed, and who all are informed. Within the framework all such information is made available to the agent as parameterized values. Another example for a generic reusable behaviour is the “Inform Forecast” behaviour of sales OA that communicates to concerned agents at periodic intervals about demand forecasts. One more such example is the “Create OA” behaviour used by a SCA. With the details of the OAs to be created passed on as parameter values, this behaviour includes generic code for creating OAs that can be used by any SCA leading to its reusability. All such agent behaviours used by SCAs and OAs constitute the *library of behaviour objects* in the framework. We emphasize here that, the behaviours include only the common reusable code with any specific organizational policy-based logic separated out (as policy objects and is only referenced inside) of it. Fig. 3.7

illustrates an example partial pseudo-code and its corresponding graphical representation for “Inform Inventory” behaviour. The example shows that the behaviour has a list of message receivers available to it. The next few lines represent initialization of the parameters, ontology, and language. Whenever this behaviour is scheduled for execution, it composes an inventory level message, adds all the receivers, and sends it.

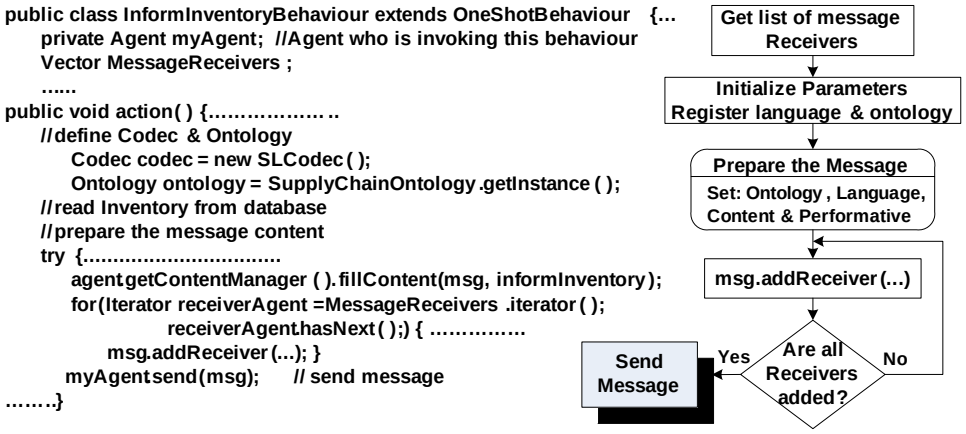


Figure 3.7: Partial pseudo-code and graphical representation for “Inform Inventory” Behaviour

3.3.2.3 Policy Objects

Real-world SC entities (and agents within) operate by employing certain policies for making decisions. In addition, from time-to-time they might resort to changing these policies based on various business conditions, or to carry out studies to understand the impact of certain policy changes. Within the framework, the agent behaviours and hence their computational logic is driven by specific policies employed. It is possible to define these policies as generic reusable objects that are referenced inside of behaviours. For example, the “Inform

Forecast” behaviour requires that the sales OA determines the demand forecast based on certain “forecasting” policy. The forecasting policy object could in-turn include sub-objects (can be policies too) one for each type of forecasting method that the sales OA could choose to adopt (e.g.: moving averages, exponential smoothing, ARIMA, …). The execution logic for any policy remains the same and the likely changes might relate to its specific parameter values. For example, exponential smoothing based forecasting policy logic remains same wherever it is used except for certain parameters like smoothing factor, single/double smoothing … etc. Such logic forms the basis for developing policy objects that can be reused across multiple agents in a SC. A “procurement policy” employed likewise, could be based on periodic or continuous reviews. A “facility expansion policy” on the other hand, could be conservative, nominal, or aggressive with relevant logic for each. All such policies constitute the *library of reusable policy objects* in the framework. The set of policies in the library includes, but not limited to: forecasting policy, order filling policy, procurement policy, production policy, capacity planning policy, facility expansion policy … etc. These policy objects are developed using Java, as the agents and behaviours are Java-based. A partial pseudo-code for an example “Forecasting Policy” is provided in Fig. 3.8 and is self-explanatory. Within the framework, appropriate policies to be employed by SCAs and OAs are initialized at the time of configuration. It is also possible to change the policies at runtime based on pre-determined logic, agent learning, or reasoning.

```
public class ForecastingPolicy  {  ……
        private int forecast ;
        private int Forecasting _Method;
        …..
int Forecast (... ) {    …..
   If (Forecasting _Method == 0) { ….   // calculate forecast using Moving Average method
        return forecast ; }
   If ( Forecasting _Method == 1) {  ….. // calculate forecast using Exponential smoothing
        return forecast ;  }
   If ( Forecasting _Method == 2) {  ….. // calculate forecast using ARIMA
        return forecast ;  }
    ……    }
return forecast ;  }
…. }
```

Figure 3.8: Partial pseudo-code for "Forecasting" Policy

3.3.3 Supply Chain Domain Ontology

It is prudent to introduce ontology and establish its purpose in the framework, before the details of SC domain ontology (the remaining aspect of the modeling layer) are presented. There exist numerous definitions for the term "*ontology*". For a long time it represented the study of the kind of things that exist, and branch of metaphysics that deals with the nature of being [36, 37]. Simply stated, ontology represents a common vocabulary to share information in a domain (e.g., organization, healthcare, SC, engineering, education …) and includes its terms, their properties and interrelationships. In the world of information technology, "ontology" takes the form of machine-interpretable vocabulary that includes the definitions of basic concepts, their property slots, and the relationships among the various concepts/slots in a domain of interest. Ontology 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 [38]. With its extensive usage in diverse application areas (e.g., information modeling, knowledge-base creation, automated reasoning, e-

Commerce, Semantic-Web …) is well recorded; researchers are predicting that the subject of ontology would bring the next breakthrough in software development. Moreover, agents cooperating in a multi-agent setting need to have a shared ontology [39]. In MASs ontology is widely-used to facilitate communication, interaction, coordination, and negotiation among agents (e.g., [40], [41]). Accordingly, our framework adopts an ontology-based approach to agent communication and coordination. It utilizes Protégé with Beangenerator to define and generate the code for reusable FIPA/JADE-compliant SC domain ontology. Utilizing such ontology promotes structured information exchanges in MASCS and helps interoperability. The process of ontology development is creative and quite involved. What is even more intriguing is the fact that ontology can be defined in several ways, yet serve the intended purpose with the same efficiency. A very good starting point for its development is provided by [38]. In addition, [40] and [41] also show how ontology helps in the context of business processes and provide a good intuitive feel for its development. There exist numerous, freely available pre-developed ontology for various domain applications. However, we did not come across any SC ontology that could have been used in the context of our research and hence developed one on our own. Based on our research experience, we provide guidelines for defining and developing FIPA-compliant domain ontology. Although the examples provided relate to SC applications, the guidelines can still be used for defining reusable ontology for MAS applications of any domain.

In technical terms, FIPA/JADE-compliant ontology consists of a set of schemas defining the structure of the predicates, agent actions and concepts that are relevant to a domain of interest [42]. Domain ontology, therefore, has to make a clear distinction between three abstract elements (classes) – *Concept*, *AgentAction*, and *Predicate*. Concepts are the terms (abstract/concrete) of a domain that agents use to communicate and reason about. In SC domain for example, the terms like: Retailer, Manufacturer, Order, Shipment, Product, Inventory, Capacity, Demand, Forecast, Cost, Profit …; can be represented as concepts. A cursory analysis of the terms above indicate that concepts are potentially the terms related (but certainly not limited) to: SC entities, products, services, resources, goals and performance metrics, documents and information exchanged … etc. In fact, these terms could qualify themselves to be the concepts. Ontology is often developed to be reusable in multiple applications and hence, certain level of “comprehensiveness” is always desirable while defining the concepts of a domain. These concepts would have specific properties (informational or otherwise) that are known as “slots”. In fact, some of the concepts themselves could be slots under other concepts (e.g.: a “manufacturer” makes “products” using certain “components”; an “order” has a “sender”, “receiver”, and an “order ID”; hence “manufacturer” and “order” concepts can include the indicated concepts as slots). Each of these “slots” in turn, could have certain other slots as their properties (e.g., a manufacturer’s “location” slot might have a “building number”, “street name”, “city”, “state”, “zip code”; and/or a “URL” … as slots). The inner (bottom) most slots are represented by atomic data types

(e.g., strings, integers, real numbers …). The slots may also have constraints in terms of cardinality (e.g., single, multiple, required single …), and value restrictions (e.g., non-negative, positive …). Certain slots could be common across multiple concepts. They could even be defined as independent slots (not under the concepts), and referenced (e.g., an “address” instead of being under a concept could be defined as an independent slot, and referenced as a property slot under manufacturer, retailer, supplier concepts). Hence, concepts and slots most certainly will have hierarchical relationships (e.g.: SC entity concept could include manufacturer, retailer concepts as slots; they in turn could have organizational function concepts like sales, inventory, procurement as slots). The hierarchy could be the other way too with the manufacturer concept having SC entity concept as a slot. Ontology derives its power essentially from its ability to cleanly represent, process, and reason about the data conceptually entangled in a complex web like (and not just pure hierarchical) structure. All the terms of a domain (e.g. SC) can be defined as concepts along with their property slots and interrelationships under the abstract “Concept” class in the “simple JADE abstract ontology template” of the Beangenerator.

Concepts typically make no sense to be used directly as a content of an ACLMessage (a JADE class representing ACL message) [42]. However, they are referenced inside special type of concepts (AgentActions), and Predicates. “AgentAction” is a concept that indicates an action that can be performed by some agent when requested by another agent. They are defined separately since unlike “normal” concepts, they are meaningful for certain types (e.g. REQUEST)

of ACLMessage “performatives” (types of communicative act). For example, a customer procurement agent may need certain products and hence can REQUEST a particular supplier sales agent to perform the “Fill Order” action. A manufacturer agent may need to know the inventory levels of components in order to deliberate how many items of a certain product to be produced, so in turn can REQUEST the inventory agent to perform an “Inform Inventory Level” action. All such agent actions of a domain are defined along with their slots (the other relevant concepts and slots) under the “AgentAction” class of the JADE ontology template. “Predicate”, the other element of ontology (that can also take the form true or false) is an expression that says something about the status of the “world” (expressed as variables) and/or concepts. For example, a supplier sales agent might INFORM a particular customer procurement agent of the acceptance status of an order represented by a predicate “Order Status” (that can be either accepted or rejected) having its own slots (other relevant concepts and slots associated with an order). An inventory agent could INFORM a sales agent about finished goods inventory level periodically, using a predicate “Inventory Level”. A customer procurement agent wanting to know if a batch of products is shipped, can send a QUERY-IF message to a particular supplier shipping agent using the predicate “Shipment Despatch Status”. All such predicates of a domain are defined along with their slots (the other relevant concepts and slots) under the “Predicate” class of the JADE ontology template. Predicates can be used meaningfully inside an INFORM or QUERY-IF message but not in a REQUEST message. That leads to an important point worth

emphasis that REQUEST an agent action “Inform Inventory Level”; and INFORM a predicate “Inventory Level” might ultimately result in the appropriate agent acquiring the “knowledge” of inventory level, but are initiated by different agents and involve different types of interactions.

Therefore, ontology may have to be defined appropriately to facilitate all possible agent interactions. SCOR [13], a SC process description standard, can potentially play an important role in the development of standardized reusable SC domain ontology. Since it is an operational process reference-model, its role however can only be limited. SCOR can help more in terms of defining concepts (process terms and metrics of SCOR provide crucial inputs) and their relationships. But its role will be quite limited in defining the lower level “slots”. The reason being, SCOR focuses on the “what” (e.g. Sourcing Plan, Product Routing) part of the information exchanges, leaving out the “content” that constitutes the “what” part to be defined by individual companies at Level 4 (“implementation” or “decompose process elements” level). The Protégé screenshot in Fig. 3.9 illustrates, partially, the details of SC domain ontology defined and utilized in the validation process of the framework. To provide a better idea for the ontology developed, the version of our SC domain ontology utilized in validating the developed framework includes about 30 concepts and independent slots, lot more property slots, 5 agent actions, and 5 predicates. The ontology can be reused and would facilitate communication and coordination in modeling SC dynamics more at tactical levels. We are currently extending the ontology further by combining information from various resources and standards

(like SCOR) to make it “reasonably” comprehensive and useful for a wide-range of SC decision modeling applications.

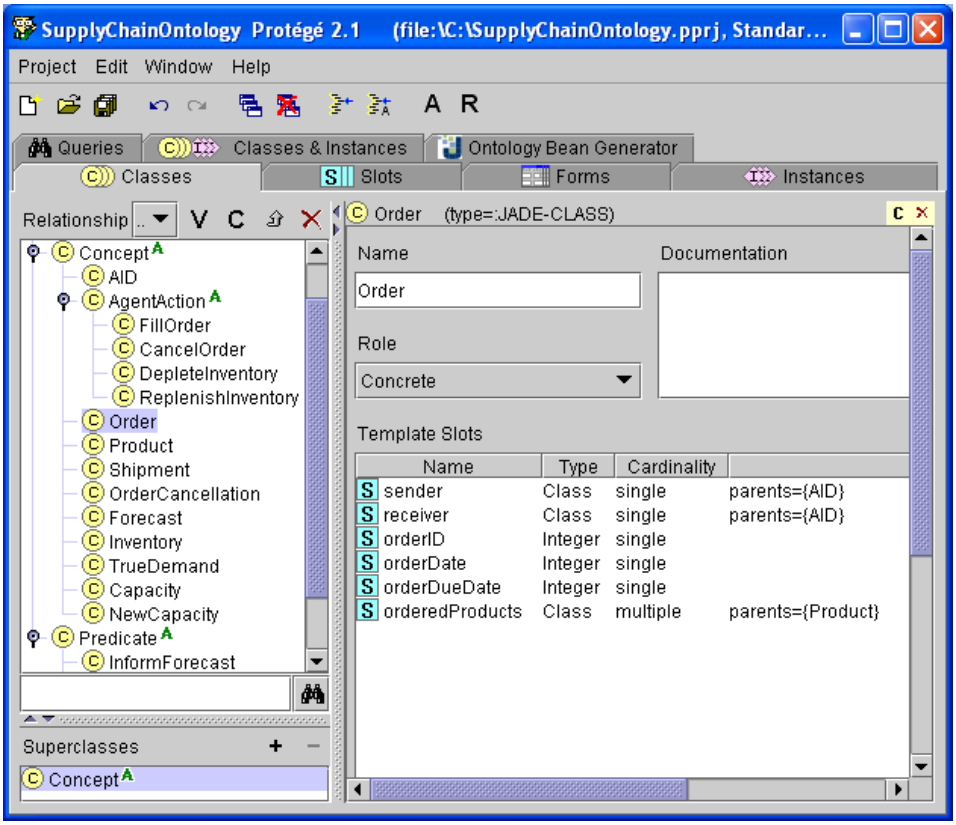


Figure 3.9: Supply chain domain ontology – partial details

3.3.4 Component Specifications & Interrelationships

Fig. 3.10 graphically illustrates the interrelationships between the various components of the framework. In addition to SC ontology, the modeling layer includes four component categories – SCAs at a higher level, then OAs, followed by behaviour and policy objects; with the higher level ones referencing the lower level ones. When a network segment is modeled at a detailed resolution, SCAs create OAs and all of them utilize their behaviours that are driven by certain policies. If instead it is modeled at an aggregated resolution, then SCAs

reference aggregated behaviours that are driven by their corresponding policies. In addition, agents and behaviours within a SC model register and utilize domain ontology for communication and coordination.

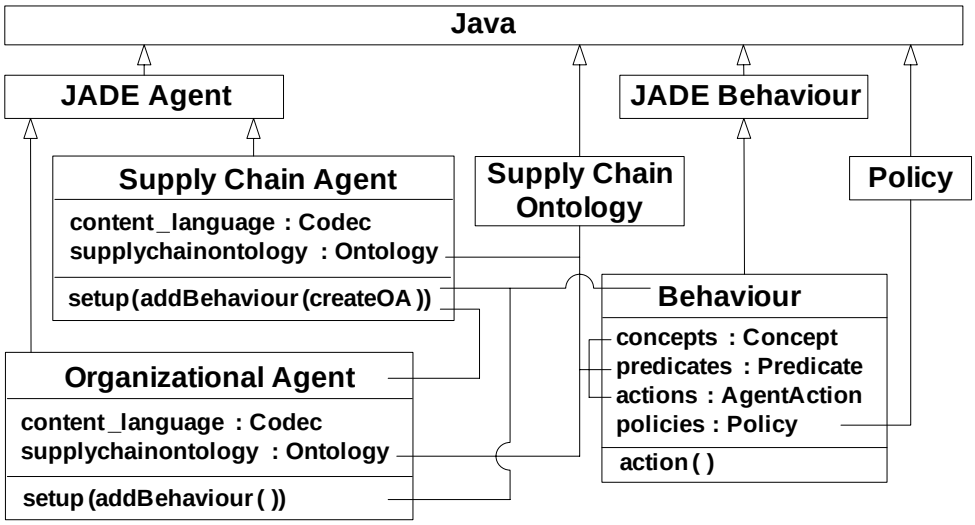


Figure 3.10: Interrelationships between the components of the framework

Having provided the details of various components and their interrelationships, we shift our focus to their software specifications. It was mentioned earlier that the details of MASCF, a generic methodological framework that creatively adopts SCOR with the Gaia methodology for analysis and design of MASCSs, was presented in [12]. Through a detailed conceptual-level analysis, MASCF progressively translates the requirement specifications of MASCS into its software specifications (agents and services models along with their schemas) that can be implemented using any MAS toolkit. If the inputs to MASCF are in the form of specific SC requirements then, its output would be specific to that SC. Therefore, given the requirements for a specific MASCS, MASCF analysis clearly identifies how many agents are needed in the system

along with the services they have to offer. If instead the inputs to MASCF are generic requirements applicable to any SC then, its output would also be applicable to any generic SC. Moreover since MASCF is based on SCOR, which is a generic standard for SC process description, the output would be standards-based and generic promoting reusability. An important point that needs emphasis here is that, in our research, the output of MASCF provided the necessary input in terms of specifications for implementing the components of various libraries of the framework (as detailed in this chapter). The framework was in fact designed to complement MASCF in order to very much achieve this intended purpose. The framework was validated (refer the later sections) by utilizing the components developed. Due to space constraints we exclude here a discussion on the development of software specifications for the various components and instead suggest the readers to refer [12] for further insights. In the following section we elaborate on how we conceive to utilize the components in modeling MASCS.

3.3.5 Configuring Supply Chain Models

This section presents a discussion on how various SC model configurations are realized. The framework supports development of either pseudo-centralized models (on single or multiple machines/platforms by a single model developer) or distributed models (on multiple geographically distributed machines/platforms by either a single or group of willing SC entities). Based on the nature of development, the network of any given SC model is configured through either one or multiple “SC Configurator” (SCC) agents. Partial pseudo-

code for an example SCC agent is illustrated in Fig. 3.11. This agent is an extension of the JADE “Agent” class and the first few lines correspond to its variable definitions and parameter initializations. The rest of the code indicates getting hold of an instance of JADE runtime, creating a container and an instance of a SCA (Retailer in this case) in it, starting the agent, and creating the rest of the SCAs as needed.

```
public class SupplyChainConfigurator extends Agent  {
  ……..
  private AgentContainer ac1 = null;
  private AgentController r1 = null;
  ……..
protected void setup ( ) {……..
  // initialize parameters
  … .
try {   //get a hold on JADE runtime
         Runtime rt= Runtime.instance();
  // Exit the JVM when there are no more containers around
          rtsetCloseVM(true);
          ……...
          Profile p= new ProfileImpl(false);
          psetParameter (Profile.CONTAINER_NAME, "Retailer");
  // Create a new Retailer container , connecting to the default
        AgentContainer ac1 = rt.createAgentContainer (p);
  // Create a new Supply Chain Agent , Retailer
        AgentController r1 = ac1.createNewAgent (“Retailer", …);
         r1.start();
  // Create all the other containers  & supply chain agents
          ……...
     } catch (Exception ex) {
         ex.printStackTrace (); }
 … }
```

Figure 3.11: Partial pseudo-code for “Supply Chain Configurator” (SCC)

Any SCC agent can be specified to create either one or all of the SCAs based on requirements. If a single model builder is developing the SC model then a single SCC agent can be specified to create all the SCAs needed. On the other hand, if multiple SC entities are involved in the development process then multiple agents are needed with every SC entity utilizing its own SCC agent to create only its corresponding SCA. Each of these SCAs have the ability to configure their segment of SC network at either aggregated (representing the

corresponding SC entity all by itself) or detailed (by creating all or some of its OAs and together representing the SC entity) leading to an overall SC model of aggregated, detailed, or hybrid resolution, as the case may be. We now elaborate on the importance of such features.

Let us assume a single entity (enterprise) wants to model the SC within its control. It may consider one level upstream and one level downstream to study its SC dynamics. The entity would for all practical purposes models the upstream and downstream levels at aggregated levels, while modeling itself at a detailed level. Moreover, the entity may be interested in modeling only either upstream or downstream operations but not both depending on requirements, and hence needs the ability to activate only some (not all) of the OAs within. In addition, multiple entities of a SC network might agree to conduct a joint study to understand SC dynamics requiring every entity to model only itself. However, SCs can quickly become extremely complex networks making interpretation of its dynamics virtually impossible. Practical considerations might force a principal entity (let us say an OEM) within the SC network to include only a select few (most important) upstream and downstream players (and not all) in the joint study. Moreover, the entity might wish to model (by itself) the rest of the upstream/downstream entities (that are not involved explicitly) together at an aggregated level. The framework is flexible to model all such configurations. In addition, it can also incorporate either information asymmetry at an inter-enterprise (or SC entity) level (all the SCAs having their own private “parameter” files and knowledge bases in the form of database tables that are also available

to the OAs that they might create) or total information asymmetry at even intra-enterprise (or individual agent) level (all the agents SCAs and OAs having their own private “parameter” files and knowledge bases in the form of database tables). These parameter files also include variables corresponding to the implementation of (direct or indirect) agent communication as shown in Fig. 3.4 and the list of agent acquaintances. Fig. 3.3 indicates one particular SC network configuration as an illustration.

3.3.6 Characteristic Features

The characteristic features of the framework (in addition to reusable and extensible components, easier installation, totally open-source freeware, and interoperability) can be summarized as:

- Generic – the framework can model wide-range of SCs (automotive, manufacturing, chemical industry … etc.) to simulate its dynamics and study control and coordination issues.
- Flexibility – the framework supports development of either pseudo-centralized or distributed models. It can incorporate information asymmetry explicitly, and facilitate study of intra- and inter-organizational dynamics, either independently or collectively.
- Uniqueness – depending on the focus and requirement, the same framework can model SCs at aggregated, detailed, or hybrid resolutions. To the best of our knowledge, we have not come across any other framework offering such modeling capability. In addition, the

framework can also help to seamlessly link strategic, tactical, and operational models in order to study their mutual influence and impact which is very difficult to implement (if not impossible) using the traditional modeling paradigms.

- Scalability – literature presents evidence that JADE is highly scalable (can model hundreds and thousands of agents simultaneously) yet remains efficient [43], and is constrained only by the standard limitations of Java programming language. Hence, the framework can easily model complex SC networks of real-world sizes with several entities at each layer/tier.
- Extensibility & Replaceability – additional tools and technologies can be easily integrated to extend the current reactive agent architecture to incorporate proactive & intelligence features (e.g.: deliberative goal-oriented rationality using Jadex, logical reasoning using Jess, semantic agent architecture …). It also allows for easier replacement of tools and technologies (e.g.: replace PostgreSQL with MySQL or Oracle).

3.4 “Tamagotchi” Case Study

In order to validate the framework and demonstrate its utility, a short-life-cycle product SC, “Tamagotchi” (detailed in [15]), is implemented as a case study. This section presents some of its modeling aspects and partial results of MA-simulation runs. Before doing so however, a brief introduction to the case is provided here. Japanese toy manufacturer Bandai Co. introduced “Tamagotchi”,

the first of the virtual pet games, in 1996. Bandai estimated that this toy had the potential to be a big hit; however, it could not accurately foresee the demand and its fluctuations. Although the product was not advertised in the mass media, the effect of word of mouth was much stronger than anticipated resulting in demand boom outpacing supply. Despite the high risks of overstocking and excess capacities involved, in order to achieve certain level of demand-supply balance, Bandai had to expand its manufacturing capacity to produce 2–3 million units per month. Subsequent to expansion, it met with a sharp decline of demand leading to huge unsold inventory resulting in an after-tax loses of US$123 million in fiscal 1998.

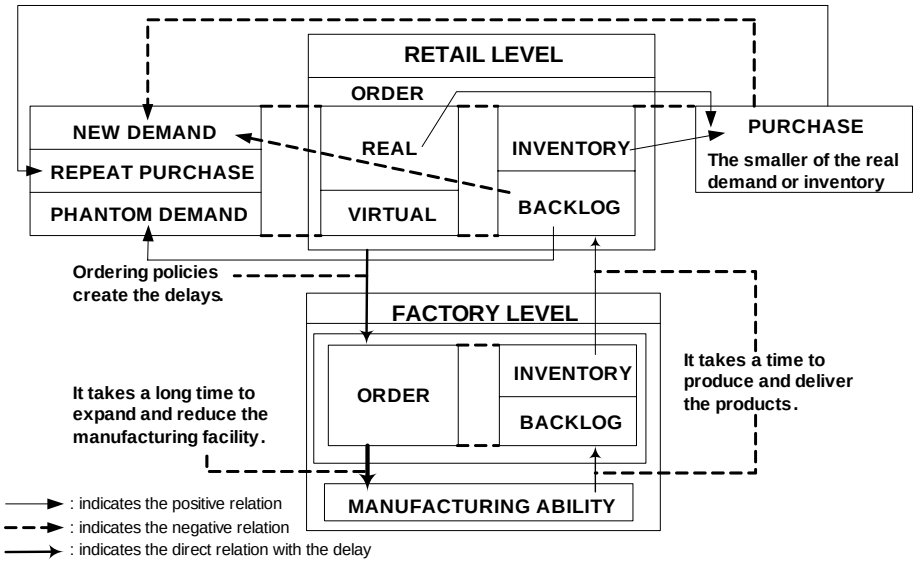


Figure 3.12: Conceptual model of Tamagotchi supply chain (source: [15])

In order to understand Tamagotchi SC dynamics and study how possibly Bandai could have avoided its adverse effects, a system dynamics simulation model was developed in [15]. The conceptual framework of the model is

illustrated in Fig. 3.12 and the readers are suggested to refer [15] for its details. Its purpose was to show how such an approach would help planners and decision makers dealing with similar new product introductions. Tamagotchi case study is a good example illustrating SC dynamics and problems associated with real-world interactions between boom and bust, capricious demand, and capacity decisions in a very short-life-cycle product setting.

3.4.1 Conceptual Analysis & Design of Tamagotchi MASCS

Given the scope and requirements of a MASCS, the most important questions that arise in its implementation are: how many agents are needed and what functionality each one of them should offer. It is extremely important that a systematic approach is adopted in finding answers to such questions and MASCF [12] was designed to perform exactly that crucial role. In our research, the entire system dynamics model information and logic (as provided in [15]) was taken-in as the input requirements for MASCS analysis and design using MASCF. Due to space constraints and not being the focus of this particular chapter, we omit the details of MASCF analysis and design, and instead suggest interested readers to refer [12] for further details and insights. We provide here only a high-level summarized output it generated in the context of the framework validation. As mentioned earlier, since MASCF received requirements of a specific SC (Tamagotchi, in this case), the output it generated is also specific to that SC. The initial steps of the MASCF analysis generated a SCOR process map that clearly identified (for the MAS) the need for three SC entities (market,

retailer, and manufacturer) associated with the three levels (market, retail, and factory) of the system dynamics conceptual model. The entire logic of the system dynamics model was split among these three SC entities, and was projected into process level logic while determining the role and interaction models. Subsequent analysis and design steps of MASCF resulted in the identification (as Table 3.1 indicates) and specification of a total of eight agents along with the services that they need to offer in order to implement Tamagotchi MASCS. It identified the requirement of one agent in the market entity, three in the retailer entity, and four in the manufacturer entity. It is obvious from the table that the sales and inventory agents are common to the both retailer and manufacturer entities and so are the services that they offer.

Table 3.1: Agents and Services for Tamagotchi Supply Chain

SC Entity	Agents	Services
Market	Market Agent	▪ Place Order
Retailer	Sales Agent	▪ Fill Order ▪ Establish Delivery Plans
	Inventory Agent	▪ Update Inventory
	Procurement Agent	▪ Source Products ▪ Establish Sourcing Plans
Manufacturer	Sales Agent	▪ Fill Order ▪ Establish Delivery Plans
	Inventory Agent	▪ Update Inventory
	Production Agent	▪ Production ▪ Plan Production
	Facility Agent	▪ Capacity Expansion

3.4.2 Implementation & Configuration of Tamagotchi Supply Chain Model

Having determined conceptually the agents and services needed for Tamagotchi MAS, the next step is to implement and configure its SC model using

the framework components. The implementation process involves mapping the output MAS software specifications generated at the end of MASCF analysis and design, onto the existing pre-developed components of the framework libraries. This step is required to determine the agent (SCA and OA) instances to be activated, and identify specific behaviours, policies, and agent interactions needed for configuring the runtime model of the system. At the outset this might appear a huge task. However, the framework components are based on SCOR, generic (applicable to any SC), and designed using MASCF; and likewise any specific model to be implemented (Tamagotchi, in this case) also goes through same MASCF analysis and design process. Hence, there in fact exists a perfect match between the components available in the library and the components needed for any specific system implementation. This reduces the mapping process to just selection of the right components from the available set, and assembling the model through configuration. In the runtime implementation of the Tamagotchi system the three SC entities are obviously represented by three SCAs: market, retailer, and manufacturer. Although the market SCA has a specific service required of it, the only purpose of retailer and manufacturer SCAs is to create OAs of their respective SC entities. As indicated in Table 3.1, the system needs to be configured to create three OAs for the retailer and four OAs for the manufacturer SC entities. Fig. 3.13 represents Tamagotchi SC model graphically showing SCAs, OAs, and all the agent interactions.

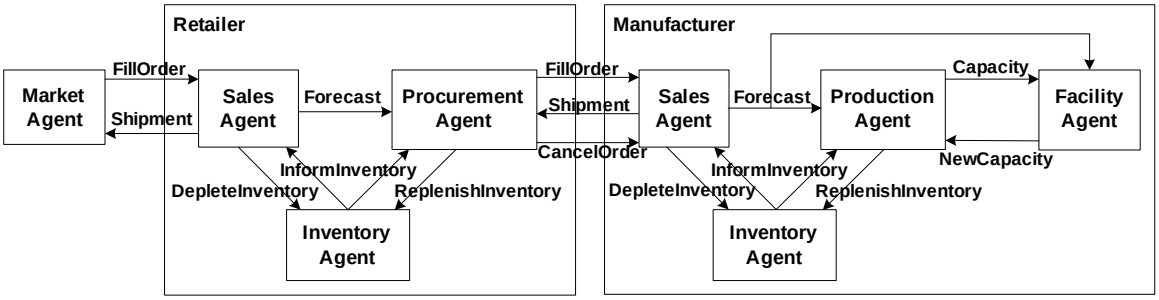


Figure 3.13: Graphical representation of Tamagotchi Supply Chain model

The next step in the validation is to configure the Tamagotchi SC runtime model. This involves preparing the SCC agent file/s and all the parameter files as necessary. The details of the SCC agents and parameter files are already presented earlier. The required number of SCC agents and parameter files essentially would depend on how the system is implemented. The framework offers flexibility to implement the system by a single model builder or, by either a single or multiple entities of a SC if they agree to do so. 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. Therefore, using the framework, Tamagotchi SC model can be implemented on a single machine, single platform on multiple machines over a network, or on multiple platforms and machines geographically spread and connected over some network. Moreover, since the framework is easily scalable, multiple entities at any level (e.g. multiple retailers for Tamagotchi SC model) can be configured with little effort. Since Tamagotchi model is meant for off-line analysis and decision support and not operational or real-time support it can also be assumed that one principal (in this case the manufacturer) SC entity implements it in its

entirety. For purpose of this research, we have configured the Tamagotchi system on a single machine. In place of the continuous time representation as in the system dynamics model, for the MA-simulation, we have utilized discrete weekly time intervals and executed it for a predetermined number of weeks. Fig. 3.14 presents a JADE screen shot of the model under execution showing various message exchanges during agent interactions. It also shows separate containers for each of the SC entities in which their corresponding SCA and OAs reside.

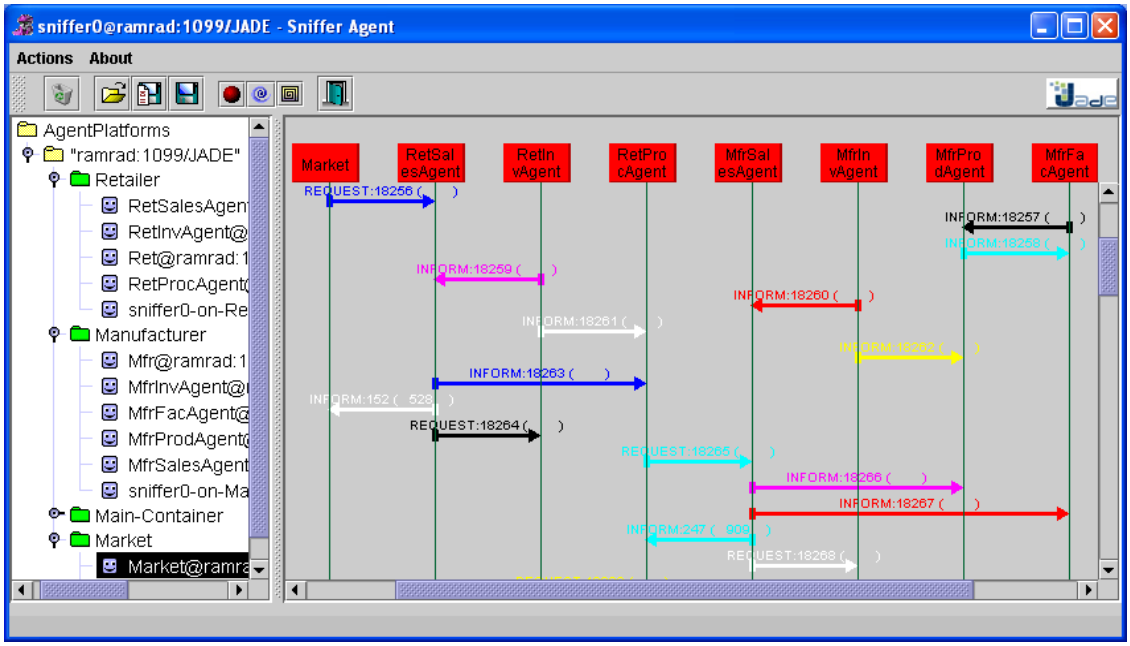
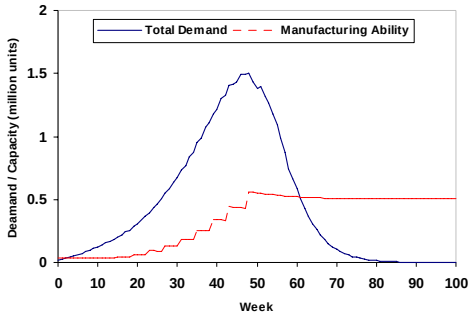


Figure 3.14: JADE screenshot for a particular simulation run of Tamagotchi supply chain model

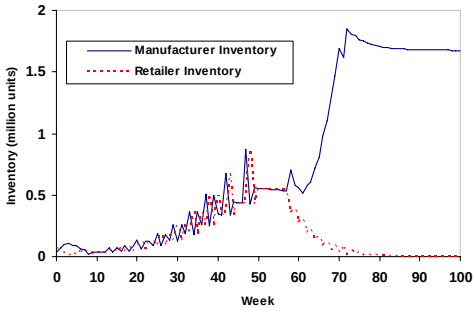
3.4.3 Multi-Agent Simulation Results

Fig. 3.15 presents some of the results obtained in the MA simulation runs. Fig. 3.15(a) indicates boom and bust phenomenon that relates total demand with manufacturing ability (capacity). The capacity is added sharply between the weeks 30 and 50 mainly driven by the phantom demand during boom phase. It

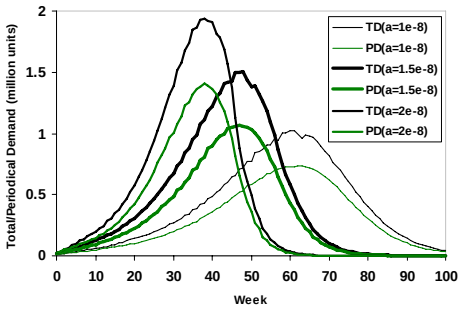
shows as the capacity reaches its peak, demand drops steeply leading to losses due to over investment and high unsold inventory. Fig. 3.15(b) shows the inventory levels at the manufacturer and the retailer. As the manufacturer builds more and more capacity reaching a certain peak, the demand vanishes and the manufacturer is left with too much of inventory largely due to excess capacity additions driven by boom and bust. Impact of multiple diffusion speeds (rate of product awareness in the market) was studied and the results are presented in Fig. 3.15(c). The plot indicates the total demand and the periodical demand for three different 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 are also much higher. This fact is evident from the “phantom demand” at the three diffusion speeds, which is largely the difference between the total and periodical demands in the figure. Therefore, it is extremely important that planning of short-life-cycle/innovative products has to consider a lot of alternatives and options unlike the functional products having stable demand profiles. Fig. 3.15(d) indicates the benefits of information sharing. The two plots represent the manufacturer inventory levels when the information is shared and not shared. It shows that when retailer shares the sales information with manufacturer, the manufacturer’s inventory drops considerably. Finally, an important aspect we would like to emphasize that the simulation results obtained for Tamagotchi MASCS model are comparable to the ones presented in [15].



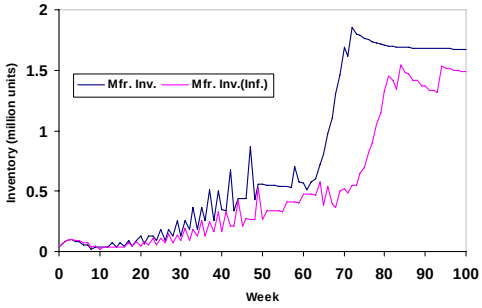
(a) Total Demand vs. Manufacturing Ability



(b) Inventory – Manufacturer vs. Retailer



(c) Impact of Diffusion Speed on Total and Periodical Demands



(d) Impact of Information Sharing on Manufacturer Inventory

Figure 3.15: Partial simulation results for Tamagotchi case

3.5 Conclusions and Research Extensions

MASs introduce a new paradigm for modeling complex systems such as SCs. Despite gaining popularity, their implementations (MASCSs in particular) however are confined to academic research to a large extent. It was asserted that in order for the real-world industrial strength MA applications to proliferate, simplification of their development process is extremely crucial. Our research focuses on simplifying the development of MASs for SC modeling and one of its outcomes was the development of a software agent-component based framework the focus of this chapter. It presented the details of the framework, the design of reusable component libraries, and how to configure MASCS models using reusable components. The strength of the framework is highlighted through its characteristic features like: uniqueness, flexibility, and scalability. Generic guidelines for developing FIPA/JADE-compliant SC domain ontology are presented. The components libraries were populated and the validity/utility of the framework was established through Tamagotchi SC case study implementation. With its effectiveness demonstrated, we are quite confident that the framework would certainly serve its intended purpose of simplifying MASCS development well. Major contributions of the research work presented in this chapter can be summarized as:

- Infrastructure development in terms of: selection of technologies, and design and implementation of a ‘Software Package’ for MASCSs implementation

- Design and development of component libraries that are based on SCOR
- Guidelines for defining and developing FIPA-compliant domain ontology in general and for SC applications in particular
- Development and validation of a framework that is efficient, easily scalable, interoperable, flexible, unique, generic, reusable, easily extensible/replaceable, 100% open-source implementation framework that reduces MASCS modeling to a matter of quick configuration
- A framework that can be applied for real-world industrial applications not just for research-oriented applications

The framework however has certain limitations that we would be addressing in its future extensions. They include incorporating graphical model specification and code-generation features for further simplification of the development process. In addition, we intend to further populate the libraries and improve upon the component functionality. Some of such extensions include: incorporating for agents the learning, intelligence, and logical reasoning features. We strongly believe that MASCSs require agents that support hybrid agent architectures – agents that are reactive, proactive (goal-directed, intelligent), logical, semantic-enabled, simultaneously. Accordingly, we are in the process of extending the current “reactive agent” architecture to support “proactive & intelligence” based architectures. Some of the extensions that we have identified for the basic framework presented in this chapter include: deliberative goal-directed BDI-reasoning based rationality, rule/data-mining/neural network based

learning, semantic agent and web services integration. Such extensions in our opinion would make the framework much more powerful and hold a great potential for real-world industrial strength MASCS applications.

3.6 References

[1] H. Min, & G. Zhou, Supply chain modeling: past, present and future, *Computers & Industrial Engineering, 43*(1-2), 2002, 231-249.

[2] D.M. Lambert, M.C. Cooper, & J.D. Pagh, Supply chain management: Implementation issues and research opportunities, *The International Journal of Logistics Management, 9*(2), 1998, 1-19.

[3] A.H. Bond, & L. Gasser (Eds.), *Readings in Distributed Artificial Intelligence,* (Morgan Kaufmann, 1988).

[4] G. Weiss (Ed.), *Multiagent systems: A modern approach to distributed artificial intelligence,* (The MIT Press, 1999).

[5] M. Luck, *50 Facts about Agent-Based Computing,* (AgentLink III, 2006).

[6] C.C. Hayes, Agents in a nutshell – A very brief introduction, *IEEE Transactions on knowledge and Data Engineering, 11*(1), 1999, 127-132.

[7] M. Wooldridge, *An Introduction to MultiAgent Systems,* (John Wiley & Sons, 2002).

[8] K.P. Sycara, Multiagent systems, *AI Magazine, 19*(2), 1998, 79-92.

[9] N.R. Jennings, P. Faratin, T.J. Norman, P. O'Brien, B. Odgers, & J.L. Alty, Implementing a business process management system using ADEPT: a real-world case study, *Applied Artificial Intelligence, 14*, 2000, 421-463.

[10] R.A. Belecheanu, S. Munroe, M. Luck, T. Payne, T. Miller, P. McBurney, & M. Pechoucek, Commercial Applications of Agents: Lessons, Experiences, and Challenges, *AAMAS'06*, 2006.

[11] T. Wagner, L. Gasser, M. Luck, J. Odell, & T. Carrico, Impact for agents, *AAMAS'05,* 2005, 93–99.

[12] R. Govindu, & R.B. Chinnam, MASCF: A Generic Process-Centered Methodological Framework for Analysis and Design of Multi-Agent Supply Chain Systems, *Working Paper*, 2006.

[13] SCC, *Supply-chain operations reference-model – version 5.0,* (Supply Chain Council, 2006).

[14] F. Zambonelli, N.R. Jennings, & M. Wooldridge, Developing Multiagent Systems: The Gaia Methodology, *ACM Transactions on Software Engineering & Methodology, 12*(3), 2003, 317-370.

[15] T. Higuchi, & M. D. Troutt, Dynamic simulation of supply chain for a short life cycle product - Lessons from the Tamagotchi case, *Computers & Operations Research, 31*, 2004, 1097-1114.

[16] M.S. Fox, M. Barbuceanu, & R. Teigen, Agent-Oriented supply chain management, *The International Journal of Flexible Manufacturing Systems, 12*, 2000, 165-188.

[17] N.M. Sadeh, D.W. Hildum, D. Kjenstad, & A. Tseng, MASCOT: an agent-based architecture for dynamic supply chain creation and coordination in the internet economy, *Production Planning & Control, 12*(3), 2001, 212-223.

[18] T. Wagner, V. Guralnik, & J. Phelps, TAEMS agents: enabling dynamic distributed supply chain management, *Electronic Commerce Research and Applications, 2*, 2003, 114-132.

[19] F. Lin, & M.J. Shaw, Reengineering the Order Fulfillment Process in Supply Chain Networks, *International Journal of Flexible Manufacturing Systems, 10*(3), 1998, 197-229.

[20] J.M. Swaminathan, S.F. Smith, & N.M. Sadeh, Modeling supply chain dynamics: A multiagent approach, *Decision Sciences, 29*(3), 1998, 607-632.

[21] M.E. Nissan, Agent-based supply chain integration, *Information Technology and Management, 2*, 2001, 289-312.

[22] J. Gjerdrum, N. Shah, & L.G. Papageorgiou, A combined optimization and agent-based approach to supply chain modeling and performance assessment, *Production Planning & Control, 12*(1), 2001, 81-88.

[23] N. Julka, R. Srinivasan, & I. Karimi, Agent-based supply chain management – 1: framework, *Computers and Chemical Engineering, 26*, 2002, 1755-1769.

[24] J. Jiao, X. You, & A. Kumar, An agent-based framework for collaborative negotiation in the global manufacturing supply chain network, *Robotics and Computer-Integrated Manufacturing, 22*(3), 2006, 239-255.

[25] H.J. Ahn, H. Lee, & S.J. Park, A flexible agent system for change adaptation in supply chains, *Expert Systems with Applications, 25*, 2003, 603-618.

[26] M. Wooldridge, N.R. Jennings, & D. Kinny, The Gaia methodology for agent-oriented analysis and design, *Autonomous Agents and Multi-Agent Systems, 3*, 2000, 285-312.

[27] M.J. Wooldridge, & N.R. Jennings, Software engineering with agents: Pitfalls and pratfalls, *IEEE Internet Computing, 3*(3), 1999, 20-27.

[28] AgentLink, *Review of Software Products for Multi-Agent Systems*, (AgentLink, 2002).

[29] F. Bellifemine, A. Poggi, & G. Rimassa, Developing multi-agent systems with a FIPA-compliant agent framework, *Software – Practice & Experience, 31*(2), 2001, 103-128.

[30] F. Bellifemine, G. Caire, A. Poggi, & G. Rimassa, JADE – A White Paper, *EXP – in search of innovation, 3*(3), 2003, 6-19.

[31] H.L. Lee, V. Padmanabhan, & S. Whang, Information Distortion in a Supply Chain: The Bullwhip Effect, *Management Science, 43*(4), 1997, 546-558.

[32] H.L. Lee, V. Padmanabhan, & S. Whang, The Bullwhip Effect in Supply Chains, *Sloan Management Review, 38*(3), 1997, 93-102.

[33] M. Paich, & J.D. Sterman, Boom, Bust, and Failures to Learn in Experimental Markets, *Management Science, 39*(12), 1993, 1439-1458.

[34] F. Bellifemine, G. Caire, T. Trucco, & G. Rimassa, *JADE Programmer's Guide,* (TILab S.p.A., 2005).

[35] G. Caire, *JADE Tutorial – JADE Programming for Beginners,* (TILab S.p.A., 2003).

[36] W. Swartout, & A. Tate, Ontologies, *IEEE Intelligent Systems, 14*(1), 1999, 18-19.

[37] B. Chandrasekaran, J.R. Josephson, & V.R. Benjamins, What are ontologies, and why do we need them?, *IEEE Intelligent Systems, 14*(1), 1999, 20-26.

[38] N.F. Noy, & D.L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05,* 2001.

[39] M. Wooldridge, & N.R. Jennings, Intelligent Agents: Theory & Practice, *Knowledge Engineering Review, 10*(2), 1995, 115-152.

[40] V. Tamma, S. Phelps, I. Dickinson, & M. Wooldridge, Ontologies for supporting negotiation in e-commerce, *Engineering Applications of Artificial Intelligence, 18*, 2005, 223-236.

[41] C. van Aart, R. Pels, G. Caire, & F. Bergenti, Creating and Using Ontologies in Agent Communication, *AAMAS'02,* 2002.

[42] G. Caire, & D. Cabanillas, *JADE Tutorial – Application-defined Content Languages and Ontologies,* (TILab S.p.A., 2004).

[43] K. Chmiel, M. Gawinecki, P. Kaczmarek, M. Szymczak, & M. Paprzycki, Efficiency of JADE agent platform, S*cientific* Programming*, 13*, 2005, 1-14.

CHAPTER 4

TOWARDS AGENT RATIONALITY IN MULTI-AGENT SUPPY CHAIN SYSTEMS THROUGH BDI-BASED REASONING

Multi-agent systems (MASs) have emerged as the most natural paradigm for modeling complex systems, such as supply chains (SCs). The focus of the previous chapters had been predominantly on agent externalities. On the contrary, this chapter focuses more on agent's internal architecture and capability. We extend the agent framework (as detailed in chapter 3) to introduce the notion of rational agency and implement it through the Belief-Desire-Intention (BDI) model using Jadex. This chapter presents a *rational agent framework* along with an illustrative case involving a Facility Agent making rational investment decisions at a very rudimentary level. Finally, it provides conclusions and identifies some of the research extensions in this area.

4.1 Introduction

MASs (systems with multiple interacting agents) have evolved out of distributed artificial intelligence (DAI) (Bond and Gasser 1988; Nwana, 1996) and are being increasingly utilized for modeling business processes and SCs (Jennings et al., 1996, 2000; Van Dyke Parunak, 2000; Luck, 2006; Belecheanu et al., 2006). A software agent is defined as a computational entity that perceives, acts upon its environment, and is autonomous in its behaviour (Weiss, 1999). While as an agent is defined to be intelligent (Wooldridge and Jennings, 1995; Jennings et al., 1998; Luck and d'Inverno, 2001; Ndumu and Nwana,

1997) if it demonstrates, in addition to the central notion of agency – “autonomy”, the following capabilities/features:

- Reactiveness: perception of environment and responding to changes in a timely manner
- Proactiveness: exhibition of goal-directed behaviour through self-initiation
- Social Ability: capability to interact with other agents (including even humans if required)

It helps to better understand, before getting into the details of these capabilities further, the differences between what are known as functional and reactive systems. A functional system is defined as a system that takes an input, performs some computation using it, and produces some output. A typical example for the same perhaps is a traditional optimization model that an organization adopts for decision making, for example in: production planning, distribution planning, inventory control … etc. A reactive system on the other hand is defined as a system that cannot adequately be described by the functional view (***f: I → O***); interacts with the environment on a continuous basis, and hence must be described in terms of their on-going behaviour (Pnueli, 1986). The other typical usages of the term “reactive” include: responding rapidly to changes in the environment, and responding directly to the world rather than explicitly reasoning about it. Most of the computer systems that are developed are usually reactive in nature (Wooldridge, 2002). Before adopting intelligent agents, it is essential to understand further the features mentioned above in

order to figure out the answer for: “why do we need intelligent agents with these capabilities in the first place”. We present the following discussion predominantly based on inputs from Wooldridge (2002) that synthesized the research work carried out by numerous researchers over the years.

In the real-world, very few of the agent goals can be achieved without the cooperation of other agents that usually do not share similar goals. In such situations, the agent must negotiate and cooperate with other agents in order to achieve its goals. This calls for the requirement of ‘social ability’, and is implemented through message exchanges and/or interaction protocols. Building a system that exhibits a purely goal-directed behaviour is not very hard. A few examples for the same include: a procedure in Pascal, a function in C, or a method in Java. Goal-directed behaviours are stated in terms of the actual execution procedure along with its pre- and post-conditions. When the precondition holds, it triggers the procedure of the goal that gets executed until the post-condition is fired (when the goal is achieved or unachievable). In simple terms, goal-directed behaviour utilizes a plan for achieving the goal. This works well for functional systems since the assumption is that the environment does not change while the plan is being executed. But for non-functional reactive systems, the environment does change, making the pre-conditions and goals invalid while-in execution, and in which case it does not make sense to continue the plan execution any further. In domains that are too complex for an agent to observe completely (due to information asymmetry or where uncertainty plays a major role) the environment is likely to change during goal executions. In such dynamic

environments, agent must react accordingly and affect required changes to the goals and/or underlying plans. Researchers state that building either purely goal-directed systems or purely reactive systems that continuously respond to their environment is not difficult; what turns out to be difficult is building a system with an effective balance between goal-directed and reactive behaviours (Wooldridge, 2002). The reason for the same being: we expect the agents to execute certain plans to achieve a certain goal but, want the plan to change if it doesn't work or the goal is no longer valid. In which case, the agents are expected to react to the situation and pick an appropriate goal and a corresponding plan for execution, if needed. But then, the agent has to focus on a goal for as long as needed and should not continuously react and keep changing the goals and plans. Hence, effectively integrating goal-directed and reactive behaviours is one of the key problems that a MAS designer faces.

Associated with the notion of 'goal-directedness' is the concept of 'rationality'. Research on rationality is being conducted for several decades in various domains, including economics, philosophy, cognitive science, decision theory, game theory …; even before the concept of MASs came into existence. What is new, however, is its entry most recently into the fields of computer science, artificial intelligence, and hence MASs. The following discussion on agent rationality is predominantly based on an extensive survey of research by van der Hoek and Wooldridge (2003). Based on their definition, an agent is an entity situated in some environment capable of action to modify, shape, and control it; while a rational agent is one that acts in its own best interests (by

deliberating over the possible outcomes and choosing an optimum). Formal theories on such agents view them to be practical reasoning systems; that decide on the next course of action, given their personal beliefs about the world and their personal desires, and influence to change the environment around them in their own way. van der Hoek and Wooldridge (2003) critically review and assess three of the best known theories of rational agency: Cohen and Levesque's *intention logic*, Rao and Georgeff's *BDI logic*, and the *KARO framework* of Meyer et al. The Belief-Desire-Intention (BDI) model (Bratman et al., 1988) is one of the best known approaches for reasoning about rational agents (van der Hoek and Wooldridge, 2003). The name BDI comes from the recognition of the primacy of beliefs, desires, and intentions in rational action that excludes emotion. The most interesting aspect of the BDI model, and perhaps the main reason for its popularity, is that it derives its power on three distinct accounts – a strong theoretical foundation, proven computability and implementability in different software architectures, and widely-accepted logical formalization. Since modern computing and modeling technology is increasingly leaning towards mimicking the way humans perceive, deliberate, reason, and rationalize; the BDI model with its strong foundations in the theory of human practical reasoning (developed by Bratman, 1987) has a substantial role to play in its evolution.

What is obvious from the above discussion is that complex systems such as SCs are in fact hybrids of reactive and proactive behaviours. Traditional approaches, however, usually model them to be either purely functional (i.e., goal-directed, which is proactive minus self-initiation) or purely reactive systems.

It is also obvious that, systems such as SCs are better modeled as hybrid combinations of reactive and proactive behaviours. Such an approach is much more realistic, powerful, and represents advancement over traditional modeling. Intelligent agents that exhibit both reactive and proactive functionalities are, therefore, expected to play a crucial role in this process. Researchers have begun to apply such modeling practices in various test-bed and real-world applications, for example: decentralized control of warehouse transportation system (Weyns et al., 2005), contract decommitment in a large-scale logistics setting (Jan't Hoen et al., 2005), mobile task execution in telecom service support (Lee et al., 2005), event-based policy monitoring (Braubach et al., 2005b), decision support (Hall et al., 2005), distributed and dynamic scheduling (Paulussen et al., 2003; Paulussen et al., 2004), and personal intelligent travel assistants (Beelen, 2004).

Our research focuses on simplifying multi-agent supply chain system (MASCS) development as an effective and efficient strategy, in order to realize its wide-spread adoption for both research and industrial applications. The focus of previous couple of chapters had been predominantly on the agent externalities, with agent internal architecture and reasoning being paid less attention. It is obvious from the discussion above that the strength, power, and superior functionalities of an agent are essentially derived out of its rationality and reasoning abilities. It is precisely for such reasons we adopt, for our agents, the BDI-reasoning based architecture. We extend the agent architecture of Govindu and Chinnam (2006) to incorporate proactive behaviour through agent

rationality by integrating the popular open-source BDI-reasoning engine, Jadex.

In the following section, we present a brief review on the BDI framework and pertinent literature. We present in section 4.3 the details of our framework based on rational agent architecture for modeling MASCSs. In section 4.4, we provide the details of an elementary case study involving a facility agent (refer Govindu and Chinnam, 2006; 2006 – Working Paper) making rational investment decisions as a part of SC case study involving Tamagotchi (Higuchi and Troutt, 2004). We provide the conclusions and further research directions in the final section. Before presenting the details, however, we want to emphasize the fact that the objective of this chapter is to present only an elementary first step demonstration of modeling rational agency and not on providing compelling results and benefits to be derived out of it. Also, the intention and focus is certainly not on presenting a methodology for rational agency based modeling either for SC modeling and control in general, or the entire Tamagotchi case in particular. They of course, most certainly, form part of our future research extensions in this area.

4.2 BDI-Agent Systems

In this section, we provide a short note on the foundation of BDI model, its brief introduction, discuss some of the software tools for implementing BDI, and provide a brief overview of the Jadex BDI-reasoning system. We base the following review on BDI, from among other things, predominantly on Braubach et

al. (2003; 2004a; 2005a; and 2006), Pokahr et al. (2005a), and van der Hoek and Wooldridge (2003).

There exist several theories to implement cognitive capabilities within agents with each one having its own strengths and weaknesses. Braubach et al. (2005a) identify the most influential of them as: the BDI model (Bratman, 1987), the theory of Agent Oriented Programming (AOP) (Shoham, 1993), the unified theories of cognition (UTC leading to SOAR) (Lehman et al., 1996; Newell, 1990), and the subsumption theory (Brooks, 1986). Based on the theory of human practical reasoning, BDI model is successful because of its simplicity in reducing the framework for complex human behaviour to the motivational stance (Dennet, 1987). This means that the causes for actions are always related to human desires, ignoring other facets of human cognition such as emotions. Another important strength of the BDI model is the consistent usage of folk psychological notions that correspond to the way people talk about human behaviour. As already mentioned before, the BDI model has strong theoretical foundations from Bratman (1987; and 1990), and Bratman et al. (1988). It also has the basis of strong logical formalization from Rao and Georgeff (1991; 1992; 1995; and 1998), Kinney et al. (1996), Rao (1996), and Wooldridge (2000).

We now provide a brief introduction to the BDI system. As per van der Hoek and Wooldridge (2003), intuitively, *beliefs* correspond to the information an agent has about the world (the environment in which it is situated) that in fact may be incomplete or incorrect. *Desires* represent states of affairs that the agent would, in an ideal world, wish to be brought about. Finally, *intentions* represent

the desires that an agent has committed to achieve. Braubach et al. (2003; 2004a; and 2005a) and Pokahr et al. (2005a), on the other hand, define beliefs as the informational attitudes of an agent, i.e., beliefs represent the information, an agent has about the world it inhabits, and about its own internal state. The motivational attitudes of agents are captured in desires. They represent the agent's wishes and drive the course of its actions. *Plans* are the deliberative attitudes, by which agents achieve their goals and react to events. A plan is not just a sequence of basic actions, but could also include sub-goals forming a hierarchical structure of plans.

Literature refers to several implementations of BDI-based software systems, for example: the Procedural Reasoning System (PRS) (Georgeff and Lansky, 1987), JAM (Huber, 1999), AgentSpeak(L) (Rao, 1996), and JACK (Busetta et al., 1999). Yoshimura (2003) brought FIPA-compliance to the sophisticated BDI platform of JACK by developing a FIPA JACK plug-in. For a review of tools and standards, readers are suggested to refer (Braubach et al., 2006). Researchers from University of Hamburg developed Jadex (JADE extension), an open-source BDI-reasoning engine (Braubach et al., 2003, 2004a, 2005a; Pokahr et al., 2005a, 2005b, 2005c), as a plug-in for the most popular open-source middleware Java Agent Development framework (JADE) and even extended it to make it a stand-alone BDI-agent development system. Developed by Telecom Italia Lab (TILab), JADE is a software development framework that simplifies MAS implementation through a platform-independent, FIPA (Foundation for Intelligent Physical Agents) standards-compliant middleware

(Bellifemine et al., 2001; 2003). Since, FIPA standards promotes interoperability, its compliance becomes important for developing real-world applications. JADE includes an agent platform that allows MAS to be distributed on multiple machines, a package to develop Java agents, and a set of graphical tools that offer debugging and runtime visualization support. Based on its superior functionality and ease of extensibility, JADE has become the most popular among open-source toolkits and is being used in commercial applications as well. Numerous plug-ins/extensions being developed to improve upon the “basic” JADE functionality indicates to its popularity and effectiveness. A JADE agent however doesn’t conform itself to any specific agent architecture, it is generic and can support any type of architecture the developers might wish to adopt. That is where Jadex plays a crucial role of introducing readily available and more sophisticated BDI-reasoning based agents. To the best of our knowledge, currently there are only two alternatives for developing FIPA-standards complaint BDI-agent systems: adopt either JACK with JACK-FIPA plug-in or JADE with Jadex plug-in. Since stand-alone Jadex does not offer certain capabilities that JADE provides and its developers also rule out any such future plans (refer Braubach et al. 2005a), we rule out stand-alone Jadex as on option. However, since JACK is a commercial system, our choice was obviously to utilize JADE-Jadex based approach in our research. Most importantly, unlike most other BDI systems, Jadex does not assume that all adopted goals need to be consistent to each other (Braubach et al. 2004a, 2005a), and allows agents to have conflicting goals making it more suitable for real-world modeling applications.

4.2.1 Jadex BDI-System

We now present a brief overview on how Jadex implements the BDI model. The primary source for this information is Braubach et al. (2003, 2004a, 2005a), and Pokahr et al. (2005a). The abstract architecture of the Jadex agent is presented in Fig. 4.1.

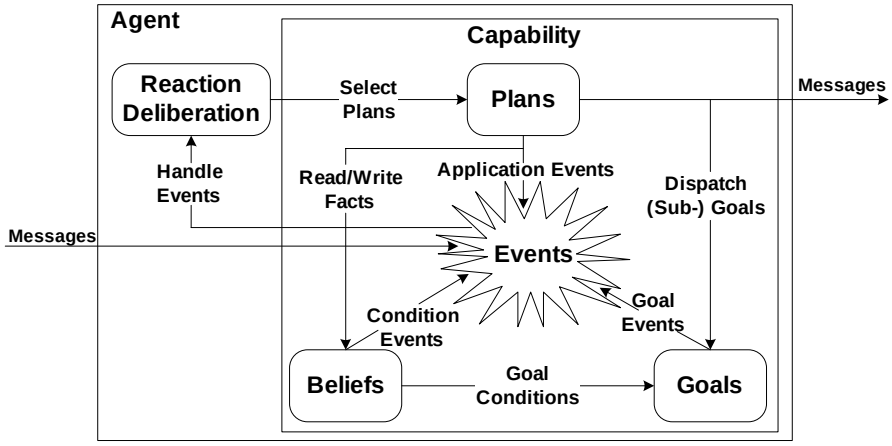


Figure 4.1: Abstract Architecture of Jadex Agent
(source: Braubach et al., 2005a)

Jadex agent is essentially an extension of JADE agent with BDI-reasoning capabilities. From outside the agent is a black-box with message reception and sending capabilities. Incoming messages, internal events, and new goals serve as inputs to agent's personal reaction and deliberation mechanism. Agent selects appropriate plans from the plan library based on the outcome of its reaction-deliberation process. Running plans might engage in some of the activities like: utilizing and updating beliefs, sending messages to communicate and coordinate with the other agents, creating new top-level goals or sub-goals, dropping some

goals, and triggering some further internal events. The reaction-deliberation mechanism is the only global component within a Jadex agent, with all other components contained in reusable modules called capabilities. Jadex defines flexible modules that essentially package functionally related entities (beliefs, goals, and plans) into a cluster for the purpose of reusability. Depending on requirements, the developer can decide whether or not to define/use capabilities. At a programming level, the developer specifies and defines all the properties of an agent through an XML document called the “Agent Definition File (ADF).” Each of the capabilities is also specified as an XML document that is very similar to ADF. Apart from these XML documents, the developer has to program each of the plans separately by extending a Jadex framework class “Plan” and make them readily available in the plan library for agent’s use at runtime.

For the sake of ease of use, Jadex doesn’t enforce a logic-based representation of beliefs. Instead it uses ordinary Java objects of any kind to be contained in the beliefbase. Objects are stored as named facts and fact sets. Through belief names, the beliefbase can be manipulated for setting, adding, or removing facts. Beliefs are used as an input for reasoning for automatically affecting changes to goals and plans. Jadex follows the general idea that goals are momentary, concrete instantiations of an agent’s desire. For any goal it has, Jadex agent directly engages in a suitable action (identified by Plan) based on a life-cycle, until it considers the goal as being reached, unreachable, or not desired any more. Jadex supports four types of goals. A perform goal is directly related to the execution of actions. An achieve goal defines a desired outcome

without specifying how to reach it. Agents may try several different alternative plans, to achieve a goal of this type. A query goal is similar to an achieve goal defined as some information that the agent wants to know about. For maintain goal, the agent keeps track of the desired state, and will continuously execute appropriate plans to re-establish the maintained state whenever needed. For further details on goal representation, readers are suggested to refer Braubach et al. (2004b). Jadex plans are the usual Java classes that extend the Jadex framework class “Plan”. Agent developer decomposes concrete agent functionality into separate plan bodies, which are predefined courses of action. Plans can be reused across agents promoting reusability.

4.3 A Rational Agent Framework for Implementing MASCS

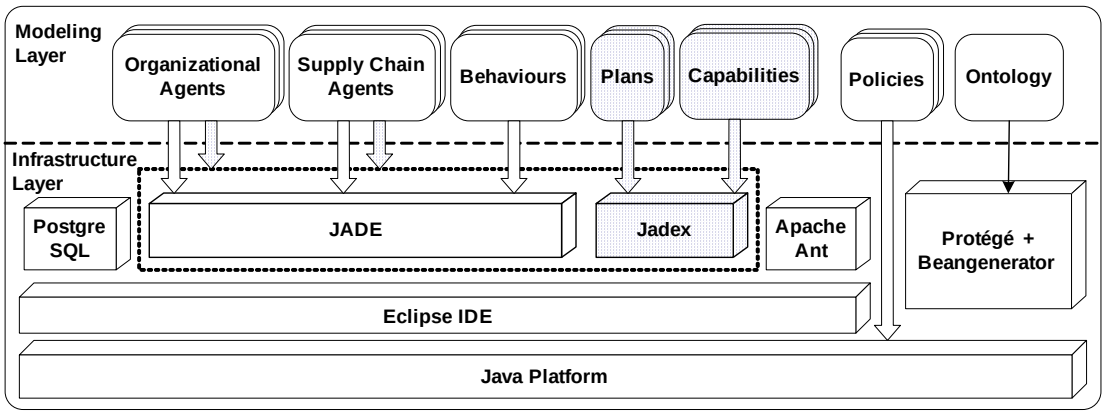


Figure 4.2: Rational Agent Framework – Architecture

In order to model MASCSs with BDI-reasoning capabilities, we have developed a Rational Agent Framework. Fig. 4.2 presents its architectural details and is quite self-explanatory. The Rational Agent Framework is developed by

extending the software agent-component based framework developed by Govindu and Chinnam (2006). In order to avoid repetition, we now describe only the changes and new additions that we have introduced; and direct the interested readers to refer Govindu and Chinnam (2006) for all further details of the base framework (the software agent-component based framework). The only change introduced in the Infrastructure layer is the addition of Jadex system as a plug-in to JADE for incorporating BDI functionality. Since Jadex was initially conceived, designed, and developed as a plug-in for JADE, they offer seamless support for developing Jadex agents, incorporating the necessary plans into Jadex agents, and runtime support for either Jadex model or a hybrid of JADE-Jadex model. The most important changes that we have incorporated correspond to the modeling layer. They include: introduction of two new generic libraries one each corresponding to “Plans” and “Capabilities”, that Jadex-based SC and organizational agents use. The Jadex Plans in fact utilize and integrate some of the JADE behaviours promoting reusability. The other changes correspond to the introduction of more flexibility into Supply Chain Agents (SCAs) and Organizational Agents (OAs). We modify their design so that the model developer could instantiate them as either pure-JADE agents or Jadex agents during SC model configuration. Such an approach allows the developers to define specific SC models that include, depending on requirement, pure-JADE agents corresponding to the ones that do not require any reasoning ability and Jadex agents corresponding to the ones that do require BDI-based rationality. This prevents the model builder from unnecessarily forcing BDI-reasoning into an

agent if it is not desired. In order to make better use of JADE communication features and SC domain ontology, we continue to utilize JADE behaviours to handle agent communication (both for receiving and sending messages) by referring them inside Jadex Plans wherever they are needed. We are currently in the process of designing and populating the components of libraries further to take care of Jadex related changes/introductions. Using an illustrative test case we have already validated the developed Rational Agent Framework, by converting the facility agent in the Tamagotchi model to be a Jadex agent. The details of the preliminary experiments and results are presented in the following section. In order to identify generic SC goals and beliefs, we propose to follow the methodological framework developed by Govindu and Chinnam (2006 – Working paper) and we adopt the Supply Chain Operations Reference-model (SCOR) (SCC, 2001). The metrics identified by the SCOR model document at various levels form the basis for defining generic goals. Likewise, the input and output information exchanges at various levels form the basis for defining generic beliefs. The process of defining such generic SC goals and beliefs are beyond the scope of this chapter but would most certainly form part of the research extensions.

4.4 An Illustrative Case

In order to validate and confirm the operational ability of the proposed Rational Agent Framework and to demonstrate the rational decision-making capabilities of the Jadex-based agents in a MASCS model (at an elementary

level), we utilize Tamagotchi case as described in (Higuchi and Troutt, 2004) and adopt the approaches proposed by Govindu and Chinnam (2006; 2006 – Working Paper) for model development. In order to avoid repetition, we refrain from presenting the details and direct the readers to refer those papers for the same. Based on the logic provided in Higuchi and Troutt (2004), the “Facility Agent” basically decides on when to invest and how much to invest in capacity, in order to achieve demand-supply balance. The agent has a leeway to do so by manipulating the variable: Investment Policy”. The system dynamics model proposed by Higuchi and Troutt (2004) is executed by fixing the “Investment Policy” variable upfront at ‘aggressive’, ‘nominal, or ‘conservative’ levels. The variable value remains the same for the entire simulation run over the entire product life cycle. If needed, the model would be run again by setting a different value, again, upfront. Having described the functionality of “Facility Agent”, we set the objectives for the illustrative case as follows:

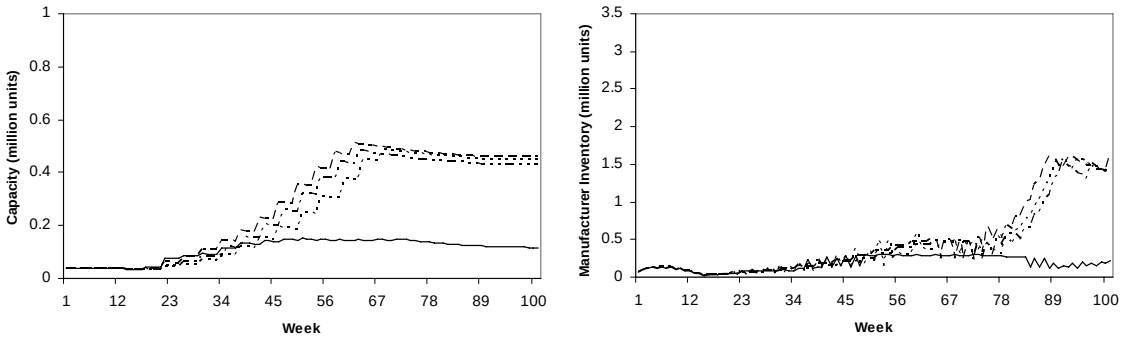
- Convert the Facility Agent into a BDI-agent by defining its required beliefs, goals, and plans
- Validate that the Rational Agent Framework indeed functions with both JADE and Jadex agents
- Run simulated experiments to determine if indeed the rational investment decisions by the Facility Agent lead to better results at an elementary level. Policy optimization is certainly not the objective; neither is finding the best learning plan/updation rules for “Investment Policy”.

First, we convert the JADE organizational agent, “Facility Agent”, as defined under the manufacturer SC entity into a Jadex-agent. The overall goal the agent has for every time period is defined as: “determine right investment”; we define it to be of type “Perform” based on Jadex. This goal is subdivided further into two goals – Goal 1: Perform “Determine Investment Strategy with a long term view” and Goal 2: Perform “Minimize short-term Demand-Capacity mismatch”. Goal 2 essentially executes a plan that determines the Demand-Capacity mismatch in the immediate future. Taking advantage of the BDI-logic, Goal 1 essentially follows the plan for updating the “Investment Policy” based on the agent’s beliefs about how its past decisions have impacted the outcome and how far into the future it has to look into.

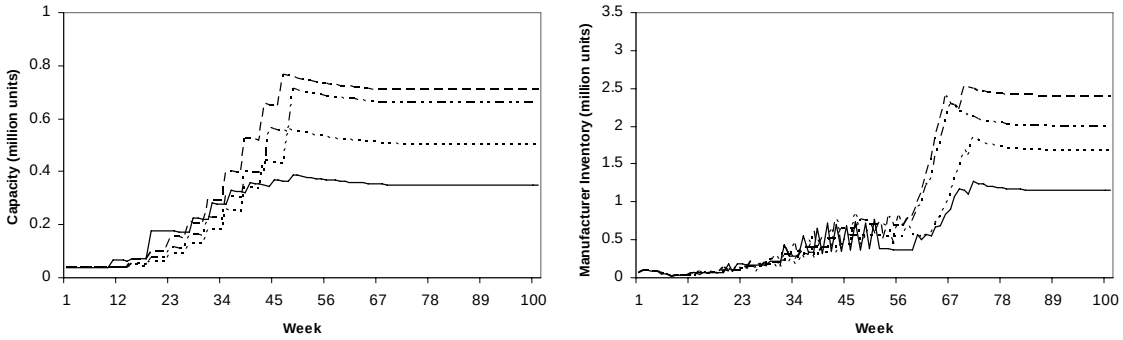
We allow the agent to update its beliefs and accordingly, allow it to generate new modified goals in way to improve the decision quality. We define a couple of beliefs for this Facility Agent: Demand-Capacity Mismatch and Demand (as realized from retailer). The facts of these beliefs are determined based on exponential smoothing in order to average out the fluctuations these variables undergo, and are updated every time period. To start with, we allow the Facility Agent to be “Aggressive” in its “Investment Policy” as in the System Dynamics model. Instead of the three integer levels for this “Investment Policy” variable, we make it to be a real variable of float type and allow it to operate within certain pre-determined bounds (lower and upper). The correction factor to be applied for the “Investment Policy” variable for each period is linked with the ratio of Average Demand-Capacity mismatch vs. Average Demand (the ratio of facts for a given

time period corresponding to the beliefs that the agent has). Having met the first objective as described above, we then developed the code for Jadex-based Facility Agent and ensured that the second objective for the case is fulfilled. Subsequently, we conducted multi-agent simulation runs as follows: For each of the three levels of diffusion speeds (Low, Medium, and High), we run four experiments – three experiments based on pure-JADE model with system dynamics logic of the value of “Investment Policy” variable fixed upfront as: Aggressive, Nominal, Conservative; the fourth one corresponds to the BDI-logic (MASCS with Jadex-based Facility Agent making rational investment decisions). We then ran all the 12 different simulation runs and generated results.

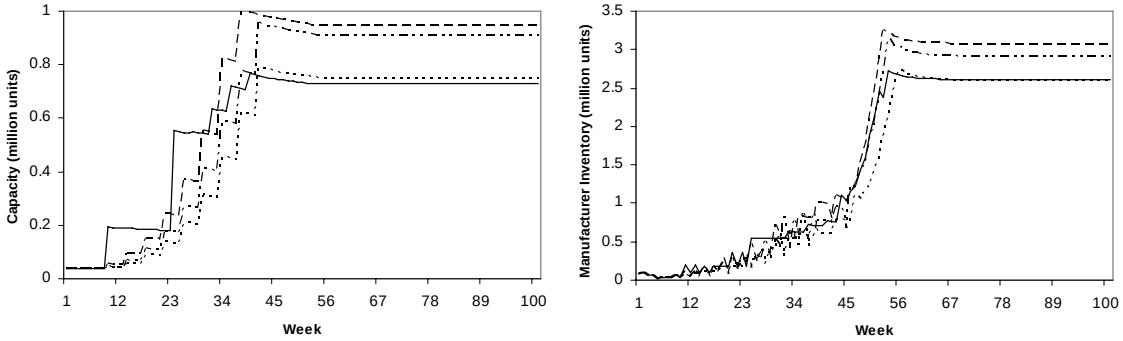
Fig. 4.3 presents the results graphically. With the Retails Sales, Retailer Inventory and the Demand Profile remaining more or less the same at each of the diffusion speeds for all the four experiments, we compare the results of the rational investment decisions with the fixed investment policies of aggressive, nominal, and conservative based on two metrics: Manufacturing Capacity, and Manufacturer Inventory. Fig. 4.3(a) and Fig. 4.3(b) show that BDI-agent is making very good rational investment decisions. It is ending up with less terminal capacity and less terminal inventory at the manufacturers’ end, the predominant reasons why Bandai Co. lost money ultimately. Fig. 4.3(c) shows that the BDI-model is making much better decisions than the aggressive and neutral strategies, but its performance is not too different from the conservative strategy.



(a) Mfg. Capacity & Mfr. Inventory at Low Diffusion



(b) Mfg. Capacity & Mfr. Inventory at Medium Diffusion



(c) Mfg. Capacity & Mfr. Inventory at High Diffusion

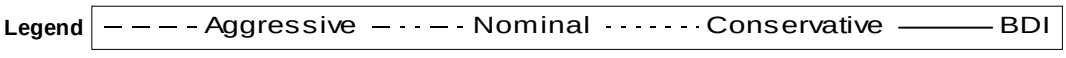


Figure 4.3: Partial simulation results – with & without BDI-reasoning

This indicates perhaps to the fact that the Plans that Tamagotchi MASCS considers cannot be the same for all the experimental settings. The plans have to be more “intelligent” and reflect the subtle differences in these experimental

settings; and that there is a need for perhaps optimization of the update function. Another aspect that needs further investigation is that the BDI model is making less investment in capacity but driving the overtime production to a higher level (not shown in figures) than any of the other three strategies. It also means that the Facility Agent should negotiate with the Production Agent to trade-off Capacity with Overtime. That perhaps calls for introduction of another goal and further plan/s. It also perhaps indicates to the fact the agent has to have some kind of costing model for making these rational decisions. These experiments have sufficiently demonstrated the capability of rational agency and fulfilled our third objective.

4.5 Conclusions and Research Extensions

This chapter introduced the notion of rational agency through an extensive review of literature. A rational agent framework for developing MASCSs based on reactive, BDI-reasoning based and hybrid agents is presented with its architecture, along with an elaboration on how we conceive the framework operates. An illustrative case of converting Tamagotchi Facility Agent into a Jadex-based BDI-rational agent is also presented along with an elaboration on how a BDI-agent is modeled. Multi-agent simulations were carried out to validate and demonstrate the power of rational agents at an elementary level. The results obtained certainly validate and prove the framework's utility. Further immediate research work being planned in this area includes: extending agent rationality to all the necessary agents of the Tamagotchi SC, and taking rationality to a higher

level by introducing more goals, costing models …etc. Major contributions of the research work presented in this chapter can be summarized as:

- Design and validation of rational agent framework that supports – reactive, BDI-reasoning based, and hybrid agent architectures
- Extending modeling layer to include generic plan and capability libraries

On the research extensions front, scope obviously exists to better design and develop the libraries of the framework to make their components more functionally capable and powerful. It is important to improve upon the quality of goals defined and the planning abilities provided by the framework. The most important aspect of future research includes the introduction of algorithmic aspects into the architecture. Since plans of the BDI-model can support any type of algorithms we would like to introduce a library of algorithms/methods based on mathematical programming, statistical, computational intelligence, game theory, auction theory, and discrete event/system dynamics simulation functionality into the framework in order to make it more powerful. Other extensions include enabling the framework infrastructure to seamlessly integrate with popular Enterprise Resource Planning (ERP) and SCM software products. Another important research extension corresponds to the introduction of goal-oriented requirements engineering into the framework to better model MASCSs.

4.6 References

Beelen, M. (2004). Personal Intelligent Travel Assistant – A distributed approach. Masters' Thesis, Delft University of Technology.

Belecheanu, R.A., S. Munroe, M. Luck, T. Payne, T. Miller, P. McBurney & M. Pechoucek. (2006). Commercial Applications of Agents: Lessons, Experiences, and Challenges, AAMAS'06.

Bellifemine, F., A. Poggi & G. Rimassa. (2001). Developing multi-agent systems with a FIPA-compliant agent framework, Software – Practice & Experience, 31(2), 103-128.

Bellifemine, F., G. Caire, A. Poggi & G. Rimassa. (2003). JADE – A White Paper, EXP – in search of innovation, 3(3), 6-19.

Bond, A.H. & L. Gasser. (Eds.), (1988). Readings in Distributed Artificial Intelligence, Morgan Kaufmann.

Bratman, M.E. (1987). Intentions, Plans, and Practical Reason. Harvard University Press: Cambridge, MA.

Bratman, M.E., D.J. Isreal & M.E. Pollack. (1988). Plans, and Resource-Bounded Practical Reasoning. Computational Intelligence, 4(4), 349-355.

Bratman, M.E. (1990). What is Intention? In P.R. Cohen, J.L. Morgan & M.E. Pollack (Eds.) Intentions in Communication, pp. 15-32, The MIT Press: Cambridge, MA.

Braubach, L., A. Pokahr & W. Lamersdorf. (2006). Tools and Standards. In: S. Kirn, O. Herzog, P. Lockemann & O. Spaniol. (Eds.). Multiagent Engineering – Theory and Applications in Enterprises, Springer Series:

International Handbooks on Information Systems.

Braubach, L., A. Pokahr & W. Lamersdorf. (2005a). Jadex: A BDI Agent System Combining Middleware and Reasoning. In: R. Unland, M. Calisti & M. Klusch. (Eds.). Software Agent-Based Applications, Platforms and Development Kits, Birkhäuser.

Braubach, L., W. Lamersdorf, Z. Milosevic & A. Pokahr. (2005b). Policy-rich Multi-Agent support for e-Health applications, In: Challenges of Expanding Internet: E-Commerce, E-Business, and E-Government: 5th IFIP conference on e-Commerce, e-Business, and e-Government (I3E 2005).

Braubach, L., A. Pokahr & W. Lamersdorf. (2004a). Jadex: A Short Overview. In: Main Conference Net.ObjectDays 2004, AgentExpo.

Braubach, L., A. Pokahr, D. Moldt & W. Lamersdorf. (2004b). Goal Representation for BDI Agent Systems. In: The Second International Workshop on Programming Multiagent Systems (PROMAS-2004).

Braubach, L., W. Lamersdorf & A. Pokahr. (2003). Jadex: Implementing a BDI-Infrastructure for JADE Agents. TILAB "EXP in search of innovation" a special issue on JADE, 3(3), 76-85.

Brooks, R. (1986). A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, 2(1), 24-30.

Busetta, P., R. Ronnquist, A. Hodgson & A. Lucas. (1999). JACK Intelligent Agents – Components for Intelligent Agents in Java. AgentLink News Letter, January 1999.

Dennet, D. (1987). The Intentional Stance. Bradford Books.

Georgeff, M. & A. Lansky. (1987). Reactive Reasoning and Planning: An Experiment with a Mobile Robot. In: Proceedings of the 1987 National Conference on Artificial Intelligence (AAAI 87), pp. 677-682, Seattle, WA.

Govindu, R. & R.B. Chinnam. (2006). A software agent-component based framework for multi-agent supply chain modelling and simulation. (accepted for publication by: International Journal of Modelling and Simulation).

Govindu, R. & R.B. Chinnam. (2006 – Working Paper). MASCF: A generic process-centered methodological framework for analysis and design of multi-agent supply chain systems.

Hall, D., Y. Guo, R.A. Davis & C. Cegielski. (2005). Extending unbounded systems thinking with agent-oriented modeling: conceptualizing a multiple perspective decision-making support system. Decision Support Systems, 41(1), 279-295.

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.

Huber, M. (1999). JAM : A BDI-Theoretic Mobile Agent Architecture. In: O. Etzioni, J.P. Muller & J. Bradshaw. (Eds.). Proceedings of the Third Annual Conference on Autonomous Agents. (pp. 236-243). ACM Press.

Jan't Hoen, P., V. Robu & H. La Poutre. (2005). Decommitment in a Competitive Multi-Agent Transportation Setting. In R. Unland, M. Klusch & M. Calisti (Eds.) Software Agent-Based Applications, Platforms and Development Kits. Whitestein Series in Software Agent Technologies. Birkhauser Verlag.

Jennings, N.R., K. Sycara & M. Wooldridge. (1998). A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems, 1, 7-38.

Jennings, N.R., P. Faratin, M.J. Johnson, T.J. Norman, P. O'Brien, B. Odgers & J.L. Alty. (2000). Implementing a business process management system using ADEPT: a real-world case study. Applied Artificial Intelligence, 14, 421-463.

Jennings, N.R., P. Faratin, M.J. Johnson, P. O'Brien & M.E. Wiegand. (1996). Using Intelligent Agents to Manage Business Processes. In: Proceedings of 1st International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology, (pp. 345-360), London, UK.

Kinney, D., M. Georgeff & A. Rao. (1996). A Methodology and Modelling Technique for Systems of BDI Agents. Technical Note 58, Australian Artificial Intelligence Institute.

Lee, H., P. Mihailescu & J. Shepherdson. (2005). TeamWorker: An Agent-Based Support System for Mobile Task Execution. In R. Unland, M. Klusch, M. Calisti (Eds.) Software Agent-Based Applications, Platforms and Development Kits. Whitestein Series in Software Agent Technologies. Birkhauser Verlag.

Lehman, J.F., J.E. Laird & P.S. Rosenbloom. (1996). A gentle introduction to Soar, an architecture for human cognition. In: S. Sternberg & D. Scarborough (Eds.), Invitation to Cognitive Science, Volume 4. Cambridge, MA: MIT Press.

Luck, M. & M. d'Inverno. (2001). A Conceptual Framework for Agent Definition and Development. The Computer Journal, 44(1), 1-20.

Luck, M. (2006). 50 Facts about Agent-Based Computing. AgentLink III.

Ndumu, D.T. & H.S. Nwana. (1997). Research and development challenges for agent-based systems. IEE proceedings on Software Engineering, 144(1), 1-10.

Newell, A. (1990). Unified Theories of Cognition. Harvard University Press.

Nwana, H.S. (1996). Software Agents: An Overview. Knowledge Engineering Review, 11(3), 205-244.

Paulussen, T.O., N.R. Jennings, K.S. Decker & A. Heinzl. (2003). Distributed Patient Scheduling in Hospitals. In Proceedings of 18th International Joint Conference on Artificial Intelligence, pp. 1224-1229, Acapulco, Mexico.

Paulussen, T.O., A. Zoller, A. Heinzl, A. Pokahr, L. Braubach & W. Lamersdorf. (2004). Dynamic Patient Scheduling in Hospitals. In: M. Bichler, C. Holtmann, S. Kirn, J. Muller & C. Weinhardt. (Eds.). Coordination and Agent Technology in Value Networks, GITO Berlin.

Pnueli, A. (1986). Specification and development of reactive systems. Information Processing, 86, pp. 845-858.

Pokahr, A., L. Braubach & W. Lamersdorf. (2005a). Jadex: A BDI Reasoning Engine. In: R. Bordini, M. Dastani, J. Dix & A. Seghrouchni. (Eds.). Multi-Agent Programming, Chapter 1, (pp. 1-31), Kluwer.

Pokahr, A., L. Braubach & A. Walczak. (2005b). Jadex – User Guide (Release 0.941). University of Hamburg.

Pokahr, A., L. Braubach, L. Rudiger & A. Walczak. (2005c). Jadex – Tool Guide (Release 0.941). University of Hamburg.

Rao, A.S. & M.P. Georgeff. (1991). Modeling Rational Agents within a BDI-Architecture. Technical Note 14, Australian Artificial Intelligence Institute.

Rao, A.S. & M.P. Georgeff. (1992). An abstract architecture for rational agents. In C. Rich, W. Swartout & B. Nebel. (Eds.). Proceedings of Knowledge Representation and Reasoning (KR & R – 92), (pp. 439-449).

Rao, A.S. & M.P. Georgeff. (1995). BDI Agents: From Theory to Practice. Technical Note 56, Australian Artificial Intelligence Institute.

Rao, A. (1996). AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In: W. van der Velde & J. Perram. (Eds.). Agents Breaking Away. Springer-Verlag.

Rao, A.S. & M.P. Georgeff. (1998). Decision procedures for BDI logics. Journal of Logic and Computation, 8(3), 293-344.

SCC. (2001). Supply-chain operations reference-model (SCOR) - Version 5.0. Supply Chain Council.

Shoham, Y. (1993). Agent-oriented Programming, Artificial Intelligence, 60(1), 51-92.

van der Hoek, W. & M. Wooldridge. (2003). Towards a Logic of Rational Agency. L. J. of the IGPL. pp. 1-25.

Van Dyke Parunak, H. (2000). A Practitioner's Review of Industrial Agent Applications. Autonomous Agents and Multi-Agent Systems, 3, 389-407.

Weiss, G. (Ed.) (1999). Multiagent systems: A modern approach to distributed artificial intelligence. The MIT Press.

Weyns, D., A. Helleboogh & T. Holvoet. (2005). The Packet-world: A Test Bed for Investigating Situated Multi-Agent Systems. In R. Unland, M. Klusch & M. Calisti. (Eds.). Software Agent-Based Applications, Platforms and Development Kits. Whitestein Series in Software Agent Technologies. Birkhauser Verlag.

Wooldridge, M. (2000). Reasoning about Rational Agents. The MIT Press: Cambridge, MA.

Wooldridge, M. (2002). An Introduction to MultiAgent Systems. John Wiley & Sons.

Wooldridge, M. & N.R. Jennings. (1995). Intelligent agents: theory and practice. The knowledge Engineering Review, 10(2), 115-152.

Yoshimura, K. (2003). FIPA JACK: A Plugin for Jack Intelligent AgentsTM, Manual, School of Computer Science and. Information Technology, RMIT University.

CHAPTER 5

CONCLUSIONS & FUTURE RESEARCH

Multi-agent systems (MASs) have emerged as the most natural paradigm for modeling complex systems, such as supply chains (SCs). Although there do exist few research applications, there are hardly any real-world commercial multi-agent supply chain system (MASCS) applications. We have identified some of the major reasons for this state of affairs as: lack of detailed and “mature” development methodologies; and “work-in-process” development toolkits lacking maturity, functionality and user-friendliness. In this dissertation research, we have developed and validated three methodological frameworks partially addressing the above mentioned issues. The first framework provides a detailed methodological support for the analysis and design (excluding implementation) of MASCS. The purpose of the second framework is to simplify MASCS implementation and reduce it to a matter of quick configuration with the help of component libraries of its modeling layer and also provide infrastructure support for the development of components as well as runtime execution support. We validated both the frameworks by implementing Tamagotchi case as described by Higuchi and Troutt (2004). Next we set out to increase the internal power and functionality of the framework agents. The popular and powerful BDI-logic based reasoning is introduced into these agents by extending the implementation framework to make it a “Rational Agent Framework” for the implementation of MASCSs. The framework was then validated by simulating the impact of rational investment decisions by a facility agent at an elementary level. Having developed

and validated these frameworks, we are quite confident of their utility and performance in modeling real-world MASCS. We now present briefly the future research extensions corresponding to each of the three framework areas.

MASCF is driven predominantly by SCOR which is oriented heavily towards SC operations. Therefore, in order to simplify modeling of the aspects related to tactical and strategic SC issues, we would like to identify and integrate any standard-based models in those areas similar to SCOR. In order to speed up the process of analysis and design, it is crucial that we automate at least some of the steps in the process. This is certainly feasible given that the framework is based on SCOR, and there exists references in literature about e-SCOR systems. In addition, we would like to integrate graphical model description/specification tools based on for example UML-technology and other technologies to simplify and speed up the process further.

The immediate research step in the area of software agent-component based framework is to further populate the existing libraries and improve upon the basic functionality offered by its components, so that they can be utilized for modeling a wide-range of MASCSs. Some of the most crucial extensions include, incorporating for agents, learning and intelligence features through neural networks and datamining, and logical reasoning features through systems such as Jess. Such extensions in our opinion hold a great potential for developing real-world industrial-strength MASCS applications. In addition, we want to incorporate graphical model specification/description tools and code-generation features for further simplification of the development process.

We identify the most important research extensions of the Rational Agent Framework as, to expand the rational abilities of the agents by incorporating algorithmic based functionality into the plans based on: mathematical programming, statistics, computational intelligence, game theory, auction theory, market-based mechanisms, and discrete event/system dynamics simulation based logic. In order to do so, we will create a library of algorithms that the agents can directly refer to during execution. We want to improve upon the quality of goals defined, and in order to do so, we plan to utilize SCOR for defining generic goals and beliefs. This would make the framework standard-based and generic, making it applicable for wide-range of industries and application areas. Another important research extension is to introduce goal-oriented requirements engineering into the framework and to develop a systematic methodology for modeling BDI-reasoning based rational MASCSs.

REFERENCES

Angerhofer, B. J. & M.C. Angelides. (2000). System Dynamics Modeling in Supply Chain Management. In: J.A. Joines, R.R. Burton, K. Kang & P.A. Fishwick, Proceedings of the 2000 Winter Simulation Conference, (pp. 342-351).

Anupindi, R. & Y. Bassok. (1999). Chapter 7: Supply contracts with quantity commitments and stochastic demand. In: S. Tayur, R. Ganeshan & M. Magazine (Eds.), Quantitative Models for Supply Chain Management, (pp. 197-232). Springer.

Arntzen, B.C., G.G. Brown, T.P. Harrison & L. Trafton. (1995). Global supply chain management at Digital Equipment Corporation. Interfaces, 25(1), 69-93.

Bagchi, S., S.J. Buckley, M. Ettl & G.Y. Lin. (1998). Experience Using IBM Supply Chain Simulator. In: D.J. Medeiros, E.F. Watson, J.S. Carson & M.S. Manivannan, Proceedings of the 1998 Winter Simulation Conference, (pp. 1387-1394).

Beamon, B.M. (1998). Supply chain design and analysis: Models and methods. International Journal of Production Economics, 55, 281-294.

Belecheanu, R.A., S. Munroe, M. Luck, T. Payne, T. Miller, P. McBurney & M. Pechoucek. (2006). Commercial Applications of Agents: Lessons, Experiences, and Challenges, AAMAS'06.

Bhaskaran, S. (1998). Simulation analysis of a manufacturing supply chain. Decision Sciences, 29(3), 633-657.

Biswas, S. & Y. Narahari. (2004). Object oriented modeling and decision support for supply chains. European Journal of operational Research, 153, 704-726.

Bond, A.H., and L. Gasser (Eds.), *Readings in Distributed Artificial Intelligence,* (Morgan Kaufmann, 1988).

Cachon, G. P. (2003). Chapter 6: Supply chain coordination with contracts. In: S. Graves & T. de Kok (Eds.), Supply Chain Management – Design, Coordination and Operation, Handbooks in Operations Research and Management Science Volume 11, (pp. 229-340). Elsevier.

Cachon, G. P. & M.L. Fisher. (2000). Supply chain inventory management and the value of shared information. Management Science, 46(8), 1032-1048.

Chan, F.T.S., N.K.H. Tang, H.C.W. Lau & R.W.L. Ip. (2002). A simulation approach in supply chain management. Integrated Manufacturing Systems, 13(2), 117-122.

Chopra, S. & P. Meindl. (2006). Supply Chain Management: Strategy, Planning, and Operation. Prentice Hall, 3rd Edition.

Christopher, M. (1998). Logistics and supply chain management – strategies for reducing cost and improving service. 2nd ed., London et al.

Davidson, P. (2000). Multi Agent Based Simulation: Beyond Social Simulation. In: S. Moss & P. Davidson (Eds.) Multi Agent Based Simulation, Springer Verlag LNCS Series, Vol. 1979.

Davis, W.J. (2001). Distributed Simulation and Control: The Foundations. In: B.A. Peters, J.S. Smith, D.J. Medeiros & M.W. Rohrer. (Eds.). Proceedings of the 2001 Winter Simulation Conference, pp.187-198.

Drucker, P.F. (1998). Management's new paradigms. Forbes, 162(7), http://www.forbes.com/forbes/1998/1005/6207152a.html

Feigin, G., C. An, D. Connors & I. Crawford. (1996). Shape Up, Ship Out. OR/MS Today, 23(2).

Fox, M.S., M. Barbuceanu & R. Teigen. (2000). Agent-Oriented supply chain management. The International Journal of Flexible Manufacturing Systems, 12, 165-188.

Gan, B.P., L. Liu, S. Jain, S.J. Turner, W. Cai & W. Hsu. (2000). Distributed Supply Chain Simulation across Enterprise Boundaries. In: J.A. Joines, R.R. Barton, K.Kang & P.A. Fishwick. (Eds.). Proceedings of the 2000 Winter Simulation Conference, pp.1245-1251.

Gartner-Dataquest Report. (2005). User study: supply chain management initiatives, United States, 2004, L.M. Clark & C. Eschinger, ID Number: G00125217.

Goetschalckx, M., C.J. Vidal & K. Dogan. (2002). Modeling and design of global logistics systems: A review of integrated strategic and tactical models and design algorithms. European Journal of Operational Research, 143, 1-18.

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.

Holweg, M. & J. Bicheno. (2002). Supply chain simulation - a tool for education, enhancement and endeavour. International Journal of Production Economics, 78, 163-175.

Honaver, V. (1999). Intelligent Agents and Multi-Agent Systems. Tutorial, IEEE Conference on Evolutionary Computation (CEC), Washington, DC.

Ilgar, E. (1998). Agents vs. Objects: What are the differences/similarities? Informatica ’98.

Jennings, N.R., K. Sycara & M. Wooldridge. (1998). A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems, 1(1), 7-38.

Jennings, N.R., P. Faratin, T.J. Norman, P. O’Brien, B. Odgers & J.L. Alty. (2000). Implementing a business process management system using ADEPT: a real-world case study, Applied Artificial Intelligence, 14, 421-463.

Julka, N., R. Srinivasan & I. Karimi. (2002a). Agent-based supply chain management – 1: framework, Computers and Chemical Engineering, 26, 1755-1769.

Julka, N., I. Karimi & R. Srinivasan. (2002b). Agent-based supply chain management – 2: a refinery application. Computers and Chemical Engineering, 26, 1771-1781.

Klugl, F., C. Oechslein, F. Puppe & A. Dornhaus. (2002). Multi-Agent Modelling in Comparison to Standard Modelling. In: F.J. Barros & N. Giambiasi. (Eds.). AIS’2002 Artificial Intelligence, Simulation and Planning in High Autonomy Systems, pp. 105-110.

Lambert, D.M. & M.C. Cooper. (2000). Issues in supply chain management. Industrial Marketing Management, 29, 65-83.

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.

Lau, J.S.K., G.Q. Huang & K.L. Mak. (2002). Web-based simulation portal for investigating impacts of sharing production information on supply chain dynamics from the perspective of inventory allocation. Integrated Manufacturing Systems, 13(5), 345-358.

Lee, H.L. & C. Billington. (1995). The Evolution of Supply-Chain-Management Models and Practice at Hewlett-Packard. Interfaces, 25(5), 42-63.

Lee, H.L. & S. Whang. (1999). Decentralized multi-echelon supply chains: Incentives and information. Management Science, 45(5), 633-640.

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.

Lin, G., S. Buckley, H. Cao, N. Caswell, M. Ettl, S. Kapoor, L. Koenig, K. Katircioglu, A. Nigam, B. Ramachandran & K. Wang. (2002). The Sense-and-Respond Enterprise. OR/MS Today, 29(2).

Macal, C.M. & M.J. North. (2005). Tutorial on Agent-Based Modeling and Simulation. In: M.E. Kuhl, N.M. Steiger, F.B. Armstrong, and J.A. Joines (Eds.) Proceedings of the 2005 Winter Simulation Conference.

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.

Odell, J. (2002). Objects and Agents Compared. Journal of Object Technology, 1(1), 41-53.

Padmos, J., B. Hubbard, T. Duczmal & S. Saidi. (1999). How i2 integrates simulation in supply chain optimization. In: P. A. Farrington, H. B. Nembhard, D. T. Sturrock & G. W. Evans, (Eds.), Proceedings of the 1999 Winter Simulation Conference, (pp. 1350-1355).

Petrovic, D. (2001). Simulation of supply chain behaviour and performance in an uncertain environment. International Journal of Production Economics, 71, 429-438.

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.

Schieritz, N. & A. Grobler. (2003). Emergent Structures in Supply Chains – A Study Integrating Agent-Based and System Dynamics Modeling. HICSS 2003: 94.

Shen, W. & D.H. Norrie. (1999). Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey, An International Journal of Knowledge and Information Systems, 1(2), 129-156.

Sterman, J.D. (1989). Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment. Management Science, 35(3), 321-339.

Swaminathan, J.M., S.F. Smith & N.M. Sadeh. (1998). Modeling supply chain dynamics: A multiagent approach. Decision Sciences, 29(3), 607-632.

Terzi, S. & S. Cavalieri. (2004). Simulation in the supply chain context. Computers in Industry, 53, 3-16.

Umeda, S. & A. Jones. (1998). A simulation-based BPR support system for supply chain management. Bprbook.doc submitted to World Scientific, pp. 1-17.

Van Dyke Parunak, H., R. Savit & R.L. Riolo. (1998). Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and User's Guide. Proceedings of Multi-Agent Systems and Agent-Based Simulation (MABS'98), Springer, LNAI 1534, 10-25.

Wagner, T., V. Guralnik & J. Phelps. (2003). TAEMS agents: enabling dynamic distributed supply chain management. Electronic Commerce Research and Applications, 2, 114-132.

Weiss, G., (Ed.). (1999). Multiagent systems: A modern approach to distributed artificial intelligence. The MIT Press.

Yuan, Y., T.P. Liang & J.J. Zhang. (2001). Using Agent Technology to Support Supply Chain Management: Potentials and Challenges. DeGroote School of Business Working Paper MGD#453.

Zambonelli, F. & H. Van Dyke Parunak. (2003). Signs of a revolution in computer science and software engineering. In: Societies in the Agents World III: Third International Workshop, ESAW 2002, P. Petta, R. Tolksdorf & F. Zambonelli. (Eds.) LNCS vol. 2577, Springer-Verlag GmBH ISSN: 0302-9743, (pp. 13-28).

ABSTRACT

MULTI-AGENT SYSTEMS FOR SUPPLY CHAIN MODELING:
METHODOLOGICAL FRAMEWORKS

by

RAMAKRISHNA GOVINDU

December 2006

Advisor: Dr. Ratna Babu Chinnam

Major: Industrial Engineering

Degree: Doctor of Philosophy

Multi-agent systems (MASs) have emerged as the most natural intrinsic paradigm for modeling complex systems, such as supply chains (SCs). However, their development process remains quite involved and extremely time consuming, hindering wide-spread adoption in industrial-strength applications. This dissertation research focused on developing methodological frameworks addressing three important aspects in particular.

In the first part of the research, we develop a generic process-centered methodological framework, “Multi-Agent Supply Chain Framework (MASCF)”, to fill the crucial gap of “lack of generic methodologies” for modeling SCs using MASs. The framework introduces the notion of process-centered organization metaphor, and creatively adopts Supply Chain Operations Reference (SCOR) model to a well-structured generic MAS development methodology, Gaia, for analysis and design of multi-agent supply chain systems (MASCSs).

Next, we develop an implementation framework, “software agent-component based framework”, to simplify and speedup MAS development. Based on Java Agent DEvelopment Framework (JADE), it integrates multiple tools and technologies providing the necessary infrastructure support. With the help of pre-developed libraries of reusable components – organizational agents (OAs), SC agents (SCAs), behaviour and policy objects; the framework allows model developers to quickly configure a MASCS to simulate dynamics and study control and coordination issues. Being generic, flexible, and scalable; it supports development of either pseudo-centralized models by a single model developer, or distributed models by either a single or group of enterprises constituting a SC. The framework is unique; based on requirements it allows for representing different segments of SC network at either aggregated or detailed levels resulting in models of hybrid resolution. It facilitates studies involving intra- and inter-organizational dynamics, considering information asymmetry explicitly.

The frameworks were validated using “Tamagotchi” case study. Its SC was designed and analyzed, the generated output specifications were implemented, and multi-agent simulations were carried out. Finally, we extend the capability of our framework agents by incorporating “goal-directed” rationality through the well-known Belief-Desire-Intention (BDI) model of human practical reasoning and implement it using Jadex. As an elementary step, we convert Tamagotchi “Facility Agent” into a BDI-agent and conduct preliminary experiments to study the impact of rational investment decisions.

AUTOBIOGRAPHICAL STATEMENT

Name: Ramakrishna Govindu

Education

- **Ph.D.**, Industrial Engineering, Wayne State University, Detroit, MI, 2006
- **M.Tech.** (Master of Technology) ***with Honors*** in Quality, Reliability & Operations Research, Indian Statistical Institute, Calcutta, India, 1992
- **B.E.** (Bachelor of Engineering). ***with Distinction*** in Mechanical with specialization in Production Engineering, Osmania University, Hyderabad, India, 1990

Experience

- 1999-Present: Graduate Teaching / Research Assistant in Ford and Visteon EMMP
- 1997-1999: Consultant & Senior Consultant, Management Consultancy Division, Satyam Computer Services Limited, Hyderabad, India
- 1995-1996: Consultant & Senior Consultant, Management Consultancy Division, ABC Consulants India (Private) Limited, Hyderabad, India
- 1994-1995: Freelance Consultant, Indal Electronics Limited, Nanjangud, India
- 1992-1994: Fellow of the Specialist Development Program (SDP), Indian Statistical Institute, Bangalore, India, 1994

Journal Publications (Past & Upcoming)

- A Software Agent-Component Based Framework for Multi-Agent Supply Chain Modeling and Simulation – **accepted for publication** in: ***International Journal of Modelling and Simulation***
- MASCF: A Generic Process-Centered Methodological framework for Analysis and Design of Multi-Agent Supply Chain Systems – **under review** with ***Computers & Industrial Engineering***
- A Rational Agent Framework for Multi-Agent Supply Chain Modeling and Simulation – (**Working Paper** – To be communicated)

Conferences/Presentations (Past & Upcoming)

- INFORMS Annual Meetings – 2004, 2005, 2006
- IERC'07 (Industrial Engineering Research Conference) – Invited session on Multi-Agent Systems
- General Motors R&D, IMEGRS 2006

Professional Membership

- Member, Institute for Management Science and Operations Research (INFORMS)

Contact Information

- Hotmail or Yahoo or Gmail – ID: **RamGovindu**