MASCF: A generic process-centered methodological framework for analysis and design of multi-agent supply chain systems

Ramakrishna Govindu 1, Ratna Babu Chinnam *

Department of Industrial and Manufacturing Engineering, Wayne State University, 4815 Fourth Street, Detroit, MI 48202, USA

Received 17 March 2006; accepted 9 June 2007
Available online 14 June 2007

Abstract

Multi-agent systems (MAS) are becoming popular for modeling complex systems such as supply chains. However, development of multi-agent systems remain quite involved and extremely time consuming. Currently, there exist no generic methodologies for modeling supply chains using multi-agent systems. In this research, we propose a generic process-centered methodological framework, Multi-Agent Supply Chain Framework (MASCF), to simplify MAS development for supply chain (SC) applications. MASCF 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 the Java Agent DEvelopment Framework (JADE).

© 2007 Elsevier Ltd. All rights reserved.

Keywords: Multi-agent systems; Supply Chain Operations Reference model (SCOR); Agent-oriented methodologies; Gaia methodology; Java Agent DEvelopment Framework (JADE)

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 supply chain 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 supply chain 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 supply chain 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 supply chains (Lambert, Cooper, & Pagh, 1998; Min & Zhou, 2002). The constant battle for supremacy is no longer between an enterprise and its competitors, but between the supply chain of the enterprise and those of its competitors (Baatz, 1995; Taylor, 2003). The success of any enterprise, accordingly, depends on its ability to integrate and coordinate the intricate network of business processes among its supply chain partners efficiently (Drucker, 1998; Lambert & Cooper, 2000). It follows that the ultimate success of an enterprise is not derived independent of, but coupled with the destiny of its supply chain.

Traditional supply chain modeling and management involves the application of: optimization (e.g., Arntzen, Brown, Harrison, & Trafton, 1995; Beamon, 1998; Goetschalckx, Vidal, & Dogan, 2002; Lee & Billington, 1995), mathematical models (e.g., Anupindi & Bassok, 1999; Cachon, 2003; Cachon & Fisher, 2000; Lee & Whang, 1999), simulation (e.g., Bhaskaran, 1998; Chan, Tang, Lau, & Ip, 2002; Holweg & Bicheno, 2002; Petrovic, 2001; Terzi & Cavaleri, 2004), system dynamics (e.g., Angerhofer & Angelides, 2000; Higuchi & Troutt, 2004; Sterman, 1989), 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 of a combination of tools in decision modeling, for example: simulation-optimization (Padmos, Hubbard, Duzcma, & Saidi, 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, Buckley, Ettl, & Lin, 1998; Biswas & 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 supply chain 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 supply chains 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 multi-agent systems 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 multi-agent systems represent a new way of analyzing, designing, and implementing complex software systems (Jennings, Sycara, & Wooldridge, 1998). They are expected to pioneer a revolutionary paradigm shift in software systems modeling and engineering (Zambonelli & van Dyke Parunak, 2003). Multi-agent systems can be used to model any phenomenon, scientific or behavioral, in order to study the underlying dynamics of complex systems such as supply chains 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 multi-agent systems 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 supply chain 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 multi-agent systems 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 multi-agent system development, in particular, for supply chain 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 models/methodologies that become its key elements. The framework introduces the notion of process-centered organization metaphor, and creatively adopts a generic process-standard for supply chain description (Supply Chain Operations Reference model, SCOR) to a well-structured generic methodology for multi-agent system development (Gaia). Since SCOR and Gaia are the key elements, MASCF is a generic tool widely applicable, and practical for modeling supply chains through multi-agent systems.

This paper is structured as follows. Pertinent literature is reviewed in Section 2. Section 3 describes the key elements of the framework. MASCF is introduced in Section 4 along with its scope and limitations. The operating mechanics of the framework are discussed in detail in Section 5. The framework is validated with the help of a case study, Tamagotchi, and its details are provided in Section 6. Finally, Section 7 summarizes research contributions and identifies some of the extensions.

2. Supply chain modeling, management and 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 multi-agent systems 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 multi-agent systems (systems with multiple interacting agents): (i) modern computing and information environments are distributed, large, open, and heterogeneous, and (ii) multi-agent systems have the capacity to play an important role in developing and analyzing models and theories of interconnectivity in human societies. In terms of the application potential, they are best suited and hold a great promise for a large spectrum of complex real-world systems, in particular, supply chains. Sycara (1998) identifies their characteristic features as: Each agent has incomplete information, capabilities, thus a limited viewpoint; there is no global system control; data is decentralized; and computation is asynchronous.

2.1. Multi-agent supply chain modeling and management

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

In one of the earliest and most widely cited papers, Swaminathan, Smith, and Sadeh (1998) present a multi-agent framework for developing supply chain simulation models of appropriate fidelity with minimal time and effort. It involves composing models from a library of reusable, domain specific, primitive software components representing supply chain agents, control element objects, and their interaction protocols. They discuss a cross-docking prototype, and compare multi-agent and conventional modeling approaches. Although the authors mention a full-scale application at IBM, they neither present the complete details/results nor discuss the system details. A framework that integrates various elements of a supply chain represented in a unified, intelligent, and an object-oriented fashion is proposed by Julka, Srinivasan, and Karimi (2002). It is designed to model, monitor, manage, and help analyze business policies within a supply chain. They demonstrate its application through a prototype decision support system, PRISMS (developed using ADE on G2), to study the effects of internal policies, exogenous events and plant modifications of a refinery. Jiao, You and Kumar (2006) propose an agent-based multi-contract negotiation system for global supply chain manufacturing coordination. The system is implemented using Java Agent DEvelopment Framework (JADE) based on blackboard architecture at a leading mobile phone manufacturing company. A flexible agent system for supply chains that can adapt to transaction changes brought about by new products or trading partners was presented in Ahn, Lee, and Park (2003). Their approach was demonstrated with the help of a PC supply chain application prototype. Gjerdrum, Shah, and Papageorgiou (2001) apply multi-agent modeling techniques to simulate and control a simple demand-driven supply chain network system. They utilize Java Agent Template Lite (JATLite) for modeling the multi-agent system and optimize its manufacturing component using general batch scheduling system (gBSS).

Although multi-agent technology has been in existence for sometime and gaining popularity, we hardly came across any literature that refers to industrial strength applications in general, and supply chain applications in particular. With the exception of a few papers (e.g., Fox et al., 2000; Swaminathan et al., 1998) that discussed the importance of development of generic components and reusability aspects, most of the applications seemed to offer a specific modeling solution to a particular problem. Articles like Sadeh et al. (2001) and Wagner et al. (2003) provide a discussion on the architectural issues, but the focus has been on specific aspects like coordination or real-time scheduling. From the literature, it appears that most of the applications are research oriented in nature. There 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 multi-agent supply chain system development as a generic process. As a step in that direction, researchers have begun developing generic methodologies, and toolkits for multi-agent systems.

### 2.2. Agent-oriented methodologies

An interesting aspect is that while agent-based systems are becoming increasingly well understood, the development of multi-agent systems is not (Wooldridge & Jennings, 1999). As agents are autonomous, the development of multi-agent systems differs 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. These methodologies range from extensions of existing object-oriented methodologies to new agent-oriented techniques utilizing new modeling abstractions (O’Malley & DeLoach, 2001). They differ in software development phases, aspects of inter- and intra-agent support,
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, Jennings, and Kinny (2000). It views multi-agent system as a computational organization consisting of various interacting roles, dealing with both macro-level (societal) as well as micro-level (agent) aspects. Zambonelli, Jennings, and Wooldridge (2003) extend 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 & 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, Cossentino, & Zambonelli, 2005; Karim, 2004). Inspired by the organizational concepts, unlike the existing development (structured or object-oriented) methodologies, Castro, Kolp, and Mylopoulos (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, Perini, Giorgini, Guinchiglia, & Mylopoulos, 2004). Giorgini, Mylopoulos, and Sebastiani (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, Wood, and Sparkman (2001) describe a general-purpose methodology, MaSE – Multi-Agent Systems Engineering, for developing heterogeneous multi-agent systems. 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 & 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, 2005; Cossentino & Sabatucci, 2004). PASSI characterizes the system development using five process components (models) that are divided into phases and described using UML diagrams. Chella, Cossentino, Sabatucci, and Seidita (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 multi-agent systems as software engineering projects. Considering them as such, would lead to the development of systematic and structured processes that facilitate development of industrial strength applications. Wooldridge et al. (2000) argue that for agents to realize their potential as a software engineering paradigm, it is necessary that specific techniques tailored to them be developed. 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 current agent-oriented methodology state of affairs in a nutshell as “each methodology has its own unique perspective; however, no one methodology is useful in every situation.”

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 (Hammer & Champy, 1993; Huang, Li, & Mahajan, 2002) is the most often cited example. Such industrial implementations resulted in research that later came into existence as: Business Process Reengineering (Davenport, 1992, 1995; Davenport & Short, 1990; Hammer, 1990; Hammer & Champy, 1993), and Process-Centric Enterprises (Hammer, 1997; Hammer & 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 supply chains (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, García-Dastugue, & Croxton, 2005). The GSCF framework (Cooper, Lambert, & Pagh, 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 supply chains. At the highest level in SCOR, supply chains 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.

2.4. Gaps in current research

The available literature does not present any generic methodology that is applicable for modeling supply chains using multi-agent systems. 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 supply chain 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 the multi-agent system 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 supply chains, SCOR provides the standards that define and describe supply chain 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 multi-agent supply chain system development.

3. Key elements of MASCF

A brief description of the key elements of MASCF is presented in this section. The focus is on: SCOR-based supply chain modeling, Gaia-based analysis and design of multi-agent systems, and the notion of process-centered organization metaphor. How these elements complement each other and work inside the framework are discussed in later sections.

3.1. Supply chain modeling using SCOR

Much of the material presented here to briefly describe SCOR and its role in supply chain modeling is adopted from SCC (2005) and SCC (2001). 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: standard descriptions of SCM 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 supply chains through effective communication among supply chain partners. It is used to describe, measure, and evaluate supply chain configurations in order to achieve competitive advantage.

As described in Fig. 1, SCOR contains three levels of process detail that help integrate a supply chain from supplier’s supplier to customer’s customer:
The process definitions level (level 1) provides a macro-level description of the entire supply chain operations. It defines the scope and content for the supply chain model with the help of five distinct management processes – plan, source, make, deliver, and return.

At the process categories level (level 2), a supply chain 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 supply chain. At this level SCOR is applied for configuring supply chain 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 supply chain 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 supply chain. Configuring a supply chain “thread” illustrates how SCOR configurations can be used to describe, measure, and evaluate supply chain configurations. Since SCOR is developed

Fig. 1. The Supply-Chain Operations Reference-model (SCOR) (source: SCC, 2005).
to be a process reference model, it can model any supply chain 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.

3.2. Analysis and 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), Cernuzzi, Juan, Sterling, and Zambonelli (2004), Zambonelli, Jennings, and Wooldridge (2005). Gaia encourages the developer to think of building multi-agent systems as a process of organizational design. Using a waterfall 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, 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.

3.2.1. Analysis phase

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

---

**Fig. 2. The Gaia methodology (source: Zambonelli et al., 2003).**
• The first step is to determine whether the system has to be decomposed into multiple multi-agent systems based on system specification, real-world structures, and modularity 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.

  o 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.

  o 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.

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. This involves filling the remaining attribute specifications for the roles and interactions identified in the preliminary models, as well as specifying the attributes in their entirety for the new roles and interactions originating out of the organizational structure definition.

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 a traditional programming approach or an appropriate toolkit-based agent-development framework.

3.3. Process-centered organization metaphor for multi-agent systems

The multi-agent system 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., Dorigo, Bonabeau, & Theraulaz, 2000; Hadeli, Paul, Kollingbaum, & Van Brussel, 2004), physical (e.g., Abelson et al., 2000; Mamei, Zambonelli, & Leonardi, 2003), societal (e.g., Candea, Hu, Locchi, Nardi, & Piaggio, 2001; Veloso, Stone, & Han, 1999), and organization (Zambonelli, Jennings, & Wooldridge, 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 methodologies, the analysis process starts with the identification and definition of roles and their interactions directly from the requirements statement. This 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 multi-agent system 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 & Champy, 1993; Davenport & Short, 1990; Hammer & Stanton, 1999) in real-world organizations. We propose that in order to retain realism and gain substantial benefits, the development of multi-agent systems 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 multi-agent supply chain system 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.

4. The MASCF framework

As illustrated in Fig. 3, the process of MAS development (like any other software development) involves the execution of four phases: the requirements collection, system analysis, system design (architectural and detailed), and implementation. Development of a software agent-component based framework that focuses on system design and predominantly on implementation was detailed in Govindu and Chinnam (in press).
In contrast, we develop MASCF that focuses predominantly on system analysis and design phases. This section presents the details of MASCF, its scope and limitations, and how SCOR improves the precision and efficiency of Gaia methodology in the development of multi-agent supply chain systems.

By design, MASCF creatively combines the features of SCOR to improve the precision and effectiveness of the Gaia methodology in the system analysis and design phases. Requirements collection phase is beyond its scope, and so is the implementation phase. As Fig. 4 indicates, 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.

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 Gaia-based analysis in MASCF 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 standard process definitions of SCOR. Once the analysis phase using Gaia is complete, it generates an output in the form of SCOR-based attri-
butes relating to environment, roles, interactions, and organizational rules for all the sub-organizations of the supply chain system. This output becomes an input to 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 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 generates an output in the form of agent and services models for the multi-agent supply chain system 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 multi-agent system. 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 multi-agent system, 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 multi-agent system 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 multi-agent supply chain systems would remain a complex and tedious task. Although individual developers will still have a major 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 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 analysis using Gaia methodology. This input helps Gaia to carry out 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 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 multi-agent system 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 multi-agent supply chain systems.

4.1. Scope and 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 multi-agent supply chain systems. Hence, the framework is:

- Generic – applicable to model a wide range of multi-agent supply chain systems.
- 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 supply chain operations, and deals with both macro (societal) level and micro (agent) level aspects of multi-agent system development.
- Neutral – with respect to the both target supply chain domain and agent architecture within the multi-agent system.
- Applicable for the development of large-scale real-world multi-agent (both open, and closed) supply chain systems.
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 other aspects (e.g., tactical and strategic) of the systems not covered by SCOR.

### 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 our discussion to only those additional aspects that supply chain 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 supply chain 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.

#### 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.

##### 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 supply chain system should comprise of. Ideally, one would like to have at least as many sub-organizations as the number of independent enterprise 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 supply chain 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 supply chain. In such cases where the entire supply chain 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 supply chain thread and process map using SCOR level 2 process categories.

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 supply chain 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 multi-agent system 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 an input “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 multi-agent system under development. In which case, this particular input has to be identified and become part of the environmental model. When the multi-agent system 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.

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 a multi-agent system. 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 supply chain 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 supply chain system. It is important to pick only the relevant and applicable generic elements/aspects based on the system requirements while developing a specific multi-agent supply
chain system. To elaborate more on this aspect, let us assume that an aggregated level (level 2 process categories of SCOR are relevant) supply chain 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 supply chain 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 supply chain domain using SCOR is presented here. The description of part of the system could be as follows: Let us assume that a supply chain 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.

As shown in Fig. 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

<table>
<thead>
<tr>
<th>Role Schema: DELIVER STOCKED PRODUCT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
</tr>
<tr>
<td>This primary role fills customer order for every time period.</td>
</tr>
<tr>
<td><strong>Protocols and Activities:</strong></td>
</tr>
<tr>
<td>ReceiveOrder, ReserveInventory, ShipProduct</td>
</tr>
<tr>
<td><strong>Permissions:</strong></td>
</tr>
<tr>
<td>reads CustomerOrder</td>
</tr>
<tr>
<td>changes InventoryAvailability</td>
</tr>
<tr>
<td>changes Shipment</td>
</tr>
<tr>
<td><strong>Responsibilities</strong></td>
</tr>
<tr>
<td><strong>Liveness:</strong></td>
</tr>
<tr>
<td>DELIVER STOCKED PRODUCT = (ReceiveOrder, ReserveInventory, ShipProduct)_{max}</td>
</tr>
<tr>
<td><strong>Safety:</strong></td>
</tr>
<tr>
<td>• shipment = min(customerorder, inventoryavailability)</td>
</tr>
</tbody>
</table>

Fig. 5. Schema for SCOR-based role “Deliver Stocked Product.”
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 supply chain modeling.

5.1.4. Preliminary interaction model

Interactions capture the dependencies between the various roles in the multi-agent system 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 this interaction as “Procurement Signal” and the output as “Shipment,” while providing an appropriate description at the same time. This completes the definition of the interaction protocol “Procure Products” as indicated in Fig. 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 multi-agent supply chain systems.

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.

5.2. Gaia-based architectural design in MASCFS

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

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.

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 supply chain 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.

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 multi-agent supply chain system that can be implemented using any multi-agent toolkit.

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 supply chain 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 1 identifies some of the informational requirements of the various Gaia models and how SCOR fills the needs. The table is self-explanatory.

Whenever a multi-agent system 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 supply chain operations reference model, for any given supply chain 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 supply chain 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 “Tamagotchi” case.

6. Case study

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

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 demand boom outpacing the ability to meet 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 product life cycle setting. We utilized the information provided in Higuchi and Troutt (2004) to setup our multi-agent system. The three stages of Tamagotchi system dynamics model were identified as the three supply chain stages – manufacturer, 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

Table 1
Information-based relationship between Gaia models and SCOR elements

<table>
<thead>
<tr>
<th>Gaia model</th>
<th>Aspects of Gaia/ MASCF</th>
<th>Relevant SCOR elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizations</td>
<td>Process map</td>
<td>SCOR process map: level 2 process categories</td>
</tr>
<tr>
<td>Environmental model</td>
<td>Variables and tuples</td>
<td>Inputs and outputs of level 3 process elements – in particular information exchanged across inter-organizational boundaries (e.g., “Source” and “Deliver” process related)</td>
</tr>
<tr>
<td>Role model</td>
<td>Roles</td>
<td>Either level 2 process categories or level 3 process elements</td>
</tr>
<tr>
<td></td>
<td>Activities</td>
<td>Level 3 process elements or their details</td>
</tr>
<tr>
<td></td>
<td>Protocols</td>
<td>Level 3 process elements or their details</td>
</tr>
<tr>
<td></td>
<td>Permissions</td>
<td>Inputs and outputs of level 3 process elements</td>
</tr>
<tr>
<td></td>
<td>Liveness responsibilities</td>
<td>Sequence of level 3 process elements</td>
</tr>
<tr>
<td></td>
<td>Safety responsibilities</td>
<td>Captured by the relationship between certain inputs and outputs of level 3 process elements</td>
</tr>
<tr>
<td>Interaction model</td>
<td>Initiator and Responder</td>
<td>Either level 2 process categories or level 3 process elements</td>
</tr>
<tr>
<td></td>
<td>Inputs and Outputs</td>
<td>Inputs and outputs of level 3 process elements</td>
</tr>
<tr>
<td>Organizational rules</td>
<td>Liveness rules</td>
<td>Sequence of either level 2 process categories or level 3 process elements</td>
</tr>
<tr>
<td></td>
<td>Safety rules</td>
<td>Inputs and outputs of level 3 process elements (of other roles)</td>
</tr>
</tbody>
</table>
the system requirements and was used in developing a multi-agent system. 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.

6.2. Analysis and design of Tamagotchi supply chain using MASCF

The objective of the first model of the analysis phase is to divide the Tamagotchi supply chain 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 supply chain – market, retailer, and manufacturer. This gives 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 supply chain model is to assist in planning and not real-time operations. Our model also treats the market level demand generation at an aggregated level, and considers it to be the responsibility of market-level sub-organization. It is also possible to consider multiple markets, multiple retailers, and multiple manufacturers. However, in order to keep the model simple following the aggregated approach of Higuchi and Troutt (2004), our model also considers only one organization at each level. The next step in MASCF is to carry out SCOR-based supply chain system analysis based on the identified system requirements. With the decision made on the sub-organizations consisting Tamagotchi supply chain system, a SCOR process map (shown in Fig. 7) is generated for the entire system using the applicable SCOR level 2 process categories. The process map is self explanatory.

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 involves identification of appropriate information required from 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.

Based on the nature of the system requirements, and also since the multi-agent system 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 supply chain 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:

![Fig. 7. SCOR process map for Tamagotchi supply chain.](image-url)
Table 2
SCOR-based roles for Tamagotchi supply chain system

<table>
<thead>
<tr>
<th>Supply chain level</th>
<th>SCOR-based roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Market</td>
<td>Source Stocked Product (S1)</td>
</tr>
<tr>
<td>Retailer</td>
<td>Deliver Stocked Product (D1)</td>
</tr>
<tr>
<td></td>
<td>Plan Deliver (P4)</td>
</tr>
<tr>
<td></td>
<td>Manage Product Inventory (ES.4)</td>
</tr>
<tr>
<td></td>
<td>Manage Finished Product Inventories (ED.4)</td>
</tr>
<tr>
<td></td>
<td>Source Stocked Product (S1)</td>
</tr>
<tr>
<td></td>
<td>Plan Source (P2)</td>
</tr>
<tr>
<td>Manufacturer</td>
<td>Deliver Stocked Product (D1)</td>
</tr>
<tr>
<td></td>
<td>Plan Deliver (P4)</td>
</tr>
<tr>
<td></td>
<td>Manage In-Process Products (EM.4)</td>
</tr>
<tr>
<td></td>
<td>Manage Finished Product Inventories (ED.4)</td>
</tr>
<tr>
<td></td>
<td>Make Stocked Product (M1)</td>
</tr>
<tr>
<td></td>
<td>Plan Make (P3)</td>
</tr>
</tbody>
</table>

• “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 Bandai 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 multi-agent system is too small with very few roles that 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 Tamagotchi supply chain system in Fig. 8.

Fig. 8(a) presents the role schema for a Tamagotchi role PLAN SOURCE at the retailer level. It performs an activity “Prepare Sourcing Plan” and interacts with other roles in executing “Receive Delivery Plan” and “Receive Inventory Availability” protocols. Fig. 8(b) presents an interaction protocol in 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 production plan and receive information feedback about actual production. The detailed design phase receives input in the form of 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 Tamagotchi system are listed in Table 3. This concludes multi-agent supply chain system analysis and design for Tamagotchi using MASCF.

6.3. Implementation of Tamagotchi supply chain system and results

The output of Tamagotchi supply chain system generated at the end of the analysis and design phases using 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). JADE supports development of distributed multi-agent applications.
based on the peer-to-peer communication architecture (Bellifemine, Poggi, & Rimassa, 2001). Several industrial applications of JADE were recorded in diverse application areas including SCM, holonic manufacturing, rescue management, fleet management, auctions, tourism (Bellifemine, Caire, Poggi, & Rimassa, 2003). JADE is the most popular among all the open source toolkits available for implementing multi-agent systems. 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

Fig. 8. An example Tamagotchi role schema and interaction protocol (a) role schema PLAN SOURCE (b) interaction protocol “Produce”.

Table 3
Agents and Services for Tamagotchi supply chain system

<table>
<thead>
<tr>
<th>Agents</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Market Agent</td>
<td>Place Market Demand</td>
</tr>
<tr>
<td>Retailer Sales Agent</td>
<td>Fill Market Demand</td>
</tr>
<tr>
<td></td>
<td>Establish Delivery Plans</td>
</tr>
<tr>
<td>Retailer Inventory Agent</td>
<td>Update Inventory</td>
</tr>
<tr>
<td>Retailer Procurement Agent</td>
<td>Source Products</td>
</tr>
<tr>
<td></td>
<td>Establish Sourcing Plans</td>
</tr>
<tr>
<td>Manufacturer Sales Agent</td>
<td>Fill Retailer Order</td>
</tr>
<tr>
<td></td>
<td>Establish Delivery Plans</td>
</tr>
<tr>
<td>Manufacturer Inventory Agent</td>
<td>Update Inventory</td>
</tr>
<tr>
<td>Manufacturer Production Agent</td>
<td>Production</td>
</tr>
<tr>
<td></td>
<td>Plan Production</td>
</tr>
<tr>
<td>Manufacturer Facility Agent</td>
<td>Capacity Expansion</td>
</tr>
</tbody>
</table>
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 supply chain system under execution is illustrated in Fig. 9. It shows various messages relating to interaction protocols getting exchanged between the agents comprising the system.

Since not being the focus of this paper, and due to space constraints, we omit the details of multi-agent implementation of Tamagotchi system using JADE and suggest the readers to refer Govindu and Chinnam (in press) for the same. We present partial results of a few multi-agent simulation runs in Fig. 10.

Fig. 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. 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. 10(c). The plot indicates total demand and 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. 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).

7. Conclusions and research extensions

Multi-agent systems introduce a new paradigm for modeling complex systems such as supply chains. Although they are gaining popularity, their implementation is confined to academic research to a large extent. This is true in particular for supply chain systems. This paper argued that in order for the real-world industrial strength multi-agent applications to proliferate, it is extremely important to simplify the development process of multi-agent systems. Although numerous methodologies exist for developing multi-agent systems, 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
multi-agent supply chain systems. Most of the multi-agent systems 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 paper showed the evidence of process-centered organizations replacing function-based organizations. We introduced the notion of process-centered organization metaphor and argued that multi-agent systems, especially for supply chains, be based on it. We proposed that a 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 multi-agent supply chain system development, we proposed a generic process-centered methodological framework. Instead of developing it afresh, a generic yet comprehensive multi-agent system development methodology, Gaia, and the widely popular industry standard for process reference, SCOR, are integrated. This paper provided the details of the framework, and showed how the framework operates with the help of supply chain domain examples. We validated the framework through the analysis and design of Tamagotchi multi-agent supply chain system. In order to demonstrate the effectiveness of MASCF 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 paper can be summarized as:

- Development and validation of generic methodological support (possibly, the first of its kind) for analysis and design of multi-agent supply chain systems.
- Creative adoption of SCOR to generic MAS development methodology, Gaia.
- Introduction of the notion “process-centered organization metaphor” for multi-agent systems.

Finally, we conclude this paper 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 multi-agent system. The development of necessary infra-
structure support in terms of building libraries of SCOR-based standardized roles and interactions have
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 to speedup the process of MASCF
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 multi-agent supply chain systems even further. Finally, since
the focus of SCOR is only on SC operations, integration of similar standards that address strategic/
tactical aspects of supply chains would make the framework much more comprehensive in terms of
standardization.

References


25, 603–618.

Kang, P. A. Fishwick (Eds.), Proceedings of the 2000 winter simulation conference (pp. 342–351).

Ganesan, & M. Magazine (Eds.), Quantitative models for supply chain management (pp. 197–232). Springer.

Interfaces, 25(1), 69–93.


281–294.

JADE, 3(3), 6–19.

Practice & Experience, 31(2), 103–128.


Research, 153, 704–726.


46(8), 1032–1048.

Systems, 36(2–3), 67–86.


artificial intelligence conference.

Zambonelli (Eds.), Methodologies and software engineering for agent systems: The agent-oriented software engineering handbook
(pp. 69–88). Springer.


