A multi-resolution collaborative architecture for web-centric global manufacturing

Mihaela Ulieru *, Douglas Norrie, Rob Kremer, Weiming Shen

Intelligent Systems Group, Faculty of Engineering and Faculty of Science, University of Calgary, 2500 University Drive, Calgary, AL, Canada T2N 1N4

Abstract

We propose a recursive multi-resolution collaborative architecture (MRCA), based on multi-agent coordination mechanisms as a solid foundation for the development of web-centric cooperative applications in global manufacturing. The architecture consists of three layers: a low-level internetworking communication-support layer; a coordination layer – managing inter-agent cooperation through intelligent conversation/communication mechanisms; and an agent layer consisting of five categories of agents: interface, collaboration, knowledge management, application and resource agents. From a functional perspective, two distinctive hierarchies shape the multi-dimensionality of this architecture: a vertical hierarchy of agents and a horizontal hierarchical communication-coordination platform. The versatility of the proposed architecture (which supports practically any kind of collaborative application) and its recursive replication at all levels of resolution within the collaborative application are illustrated on a supply-chain example. © 2000 Elsevier Science Inc. All rights reserved.

Keywords: Recursive multi-resolution collaborative architecture; Multi-agent systems; Web-centric inter-enterprise cooperation; Task decomposition; Mediator-centric organization; Virtual clustering; Global resource allocation

* Corresponding author. Fax: +1-403-282-6855; web.: www.isg.enme.ucalgary.ca.
E-mail addresses: ulieru@ucalgary.ca (M. Ulieru), Norrie@enme.ucalgary.ca (D. Norrie).
1. Introduction

The tremendous progress in networking technologies has challenged dramatically the way enterprises do business by replacing traditional off-line information processing methods with on-line real-time connections to the global economy. From e-commerce and marketing intelligence to custom-based product design and ordering, more and more transactions flow over the internet making distances between business partners measurable in “network intelligence power” rather than in thousands of miles. Advances in network-enabling software (such as Java with its versatile network “hook-up” technology Jini) have changed the cash-payment into a dial-up from cellular phones to one’s bank account and the usual grocery shopping into an on-line order from the fridge-touch-screen when we run out of supplies. Not only has networking technology changed our ways of doing daily routine activities, but also it has tremendously challenged our ways of doing business as its synergistic merging with the latest advances in distributed artificial intelligence has led to breakthrough methodologies that enable on-line inter-enterprise collaboration in the global economy. For more than 10 years, the Intelligent Systems Group (formerly the Intelligent Manufacturing Systems Group) at the University of Calgary has researched and developed multi-agent systems for manufacturing applications [1–5]. The essence of all this research effort has been encapsulated into a novel, versatile and generic recursive multi-resolution collaborative architecture (MRCA) that supports web-centric global manufacturing applications. The strength of this architecture stems from its applicability/replication at different levels of resolution as we zoom in from general to specific into the global application. From the highest level of global inter-enterprise collaboration, where the strategic goals related to supply-chain management or e-commerce are specified, down to the intra-enterprise tactical and execution levels (order fulfillment, accounting, production management, machine control), the same generic structure and coordination mechanisms are used, with the proposed architecture revealing a nested hierarchy of holonic [27] sub-architectures inheriting similar properties [6,7].

2. The MRCA

2.1. Communication – collaboration perspective

To endow the MRCA with generality and versatility, while simultaneously keeping it simple, we have made the following assumptions relative to the constituent agents:
- An agent is any entity that can send or receive a message.
- An agent can be hardware or software or both (and can be a human).
An agent can be simple or complex with any desired functionality. Regarded from the inter-agent communication and collaboration perspective, the architecture naturally reveals three main levels, Fig. 1.

2.1.1. Agent level

To accomplish a task efficiently, agents specialized in doing different jobs should be able to easily communicate. As in societal organizations, agent specialization is an important feature of this architecture. On the basis of experience with distributed applications, we have identified five categories of agents that can do specific activities individually while together can support any kind of global collaboration process.

(a) **Collaboration agents** – sometimes named “yellow pages” – are in charge of finding partners for collaboration on a specific topic of interest.

(b) **Application agents** – are specialized in a particular application (e.g. product design agents, assembly coordination agents, planning and scheduling of production, material resource planning, billings, generic transaction processing, banking, machine grouping, etc.).

(c) **Resource agents** – are machines and software programs that directly execute the elementary jobs involved in fulfilling the high-level user request (e.g. CNC machines, robots, computers, query languages, application software packages, etc.).

(d) **Knowledge management agents** – are able to retrieve information for a specific application from knowledge bases and databases via intelligent querying mechanisms such as data mining and knowledge discovery. They are also able to store, transform, and in some cases (e.g. mobile agents) transport information.
Interface agents – create a user-friendly interface for each particular application, depending on the user’s needs and preferences. To generate screen representations of the information received by the interface, we use concept-mapping techniques [8]. An essential feature of a global collaborative agent system for industrial or commercial application is an effective human/system interface. Therefore, we incorporate collaborative interface agents as an integral part of the system design. An interface software layer acts as a “wrapper”, so that the interface appears to the system as just another agent. Constraint graph techniques [9] provide support for manipulation of visual items and enable interface agents to interpret (the meaning of) such user actions. Categories of visual representations are selected from recommendations of high-priority usage put forward by the industrial partners involved in the proof-of-concept prototypes.

2.1.2. Communication coordination level

This level is the key to a successful collaboration, as it manages the conversations (i.e. high-level communication) between the agents involved in each particular application. The basic communication and cooperation services are developed on top of the Internet and some lower-level network communication primitives, Fig. 2. Inter-agent collaboration is coordinated by a multi-layered communication platform consisting of messaging management, communication services management and conversation management layers, which constitute the horizontal dimension of the functional perspective on MRCA, Fig. 3.
2.1.3. Network level
Is the basic infrastructure for global inter-enterprise communication founded on message transmission between agents. It consists of any existing network communication infrastructure (internet, intranet, extranet, virtual private network, etc.).

2.2. Functional perspective

From the perspective of its functionality in solving a particular task, the proposed architecture reveals its multi-dimensionality through a double hierarchy, Fig. 3. The vertical agent hierarchy, manages the inter-enterprise cooperation through five layers of agents. The horizontal (in-depth) communication platform hierarchy is the communication coordination level in Fig. 1, expanded as the three layers which detail the services involved in managing inter-agent (and subsequently inter-enterprise) communication: messaging management, communication services management and conversation management layer.

2.2.1. Vertical (agent) hierarchy
When agents collaborate to solve the assigned task the message-passing goes top-down and back through this hierarchy, analogously to the open system interconnection (OSI) seven-layer network communication protocol/architecture.
The first three layers constitute the *interware*, which facilitates the emergence of collaborative groupings as needed for solving current tasks and sub-tasks. Once a goal has been set for the global collaborative enterprise system, first mediator agents (which are meta-application agents in charge with task decomposition and distribution over the available resources) [10] split it into sub-tasks and sub-sub-tasks (Fig. 4). Then the interware identifies partners for collaboration on solving the multi-resolution-decomposed sub-tasks and the mediator agents proceed to distribute the sub-sub-tasks accordingly over the global resources. The interware’s interface, collaboration and knowledge agents are involved in the high-level, strategic aspects of an application, such as directing a customer order to the best manufacturer for the respective product, and distributing the order processing across internal departments/plants and adequate suppliers. The *interface agents* facilitate human collaboration on the particular application. Using constraint graph techniques, agents in the intelligent visual interface of this layer select the appropriate category of visual representation and manipulate visual items for the specific user action (request). When designing the interface layer, all human partners will need to agree on a common set of visual representations associated with their meaning within the collaborative context.

Once a request is received via the interface layer, the *collaboration (yellow page) agents* answer questions such as “who should collaborate on...” and “who knows about...?” and further directs the request to appropriate agents in the *knowledge management layer*. Agents at this layer provide specific knowledge on the collaboration topic (e.g. product design knowledge, knowledge on who can produce different parts of a product, who sells a specified product, who provides a particular service, etc.).

![Recursive task decomposition](image-url)
When the collaborative partners have been identified via the interware and their resources allocated to sub-tasks by mediators, Fig. 4, application agents dispatch the scheduled operations on each resource accordingly. While mediator agents are in charge of resource allocation to sub-tasks, application agents are in charge of planning activities on the resources allocated to a certain task. This is the highest level of resource control. Fig. 5 shows the patterns of collaboration for the case when group technology planning techniques are used [11]. At the resource-agent level we distinguish two more resolutions of control [11]: the execution control (EC) level, where the control functions for each job are designed, and the control execution (CE) level, which downloads the code for the execution of each control function directly to the physical resource [23].

2.2.2. Horizontal (communication platform) hierarchy

2.2.2.1. Conversation management layer. In global collaborative applications, generating a solution to a problem is a multi-step and incremental process, which involves a conversation among agents using information at multiple levels of abstraction. It is a challenge to keep the conversation coherent and to enhance efficiency in related reasoning. Designing a flexible and robust computational conversation model is fundamental. For this purpose, at the conversation management level, a schema-based approach to specifying conversation policies (CP) is used to manage inter-agent communication.

We regard conversations as successions of messages involving two or more agents, which collaborate to achieve a particular goal [12]. Agents can rely on the patterns in the world to make their problem-solving more efficient. When communicating with others, agents can also rely on communication patterns
and group behavior [13]. We define a conversation schema as a pattern of conversation interactions specifying CP centered on one or more conversation topics. Topics are extracted from application domain at the domain analysis stage. A schema, Fig. 6, can be constructed by identifying conversation acts, the internal state changes and the external information exchange related to the conversation topics. A goal-directed schema is a schema in which the pattern of interaction is directed towards achieving a specified goal of participating agent(s). We use Colored Petri Nets (CPN) [14] as a schema specification tool at the design stage because of the advantages of formal specification and executability. A schema is detailed by speech acts [15] and conversation acts [16]. For more details on the implementation formalism please see [17].

An agent can have several active conversations at the same time. A conversation control mechanism is provided to allow agents to suspend current conversations, while waiting for others to reach certain stages, as a dynamic sequence of conversation scenarios form a conversation hierarchy. If an agent with an intention initiates a conversation with others by choosing a schema, then the other agent(s)' pattern of response is also limited to that schema. The control mechanism will only perform the “intro-schema decisions” by executing CPN transition-firing rules [14].

2.2.2.2. Communication services management layer. This layer manages inter-agent communication through the mechanisms described in Fig. 2.

Cooperation domain server (CDS). The server acts as a central “hub” for multi-agent conversations, such that all participants may send messages directly to the server for point-to-point, multi-cast, and broadcast communication within the cooperation domain (a group of agents working together on some task). Agents within a cooperation domain may also use the server to store persistent data that will permanently be associated with the conversation, giving the conversation a lifetime beyond the transient participation of the
agents, as is often required. Servers may also store transaction histories for future playback of the chronological development of the conversation artifacts. Servers may perform all these tasks because all messages use KQML [18], which flexibly provides a basic semantic “wrapper” on messages that may be otherwise domain specific. Both the utility of generic services and the efficiency of domain specific languages are therefore provided.

The CDS is the receiver of all messages sent by the agents in the cooperation domain. Thus, the CDS can determine the ultimate destination and some of the high-level semantics of the message. The CDS takes care of forwarding the message transparently to its destination, whether it is an individual agent, a list of agents, all agents of a particular service type (role), or all agents in the cooperation domain. In addition, the CDS may record all transactions to enable history playback and “group undo” operations.

A collaborative agent system may have one or more CDSs. The CDS listens to a specified port for new connections. Other agents may send requests to this port to create a new cooperation domain, or join an active or inactive cooperation domain. The server may allow this request by creating a new CDS (in the case of a new cooperation domain or a join to an inactive cooperation domain) and replying the agent with the port number that the CDS will use. In addition, the server offers data storage services to the cooperation domain – storing and retrieving named stream data on request.

Area and local area coordinators (LAC). An area is a convenient quasi-physical division of the network that can be controlled by a LAC. The area may be a single computer, some arbitrary division of a single computer, or a cluster of computers. A local area coordinator acts both as a representative of the area to the outside world, and a manager for the local agents within the area. There is exactly one LAC per area, which is responsible for local coordination and tasks, such as “waking up” a local agent on behalf of a remote agent. Any agent may send a message to the LAC to request that the LAC ask one of its local agents to join a particular cooperation domain. The LAC will invoke the agent if necessary, and pass along the request to the agent. The LAC also facilitates the local agent’s communication to the outside world by providing an interface service to the outside world. Instead of each agent having to keep a model of network locations of servers and yellow pages, the LAC provides a central cache of such data. Local agents may query the LAC to obtain yellow page services; the LAC may, in turn, pass these requests to one or more known yellow pages and broker the responses. This simplifies network-modeling requirements for individual agents.

Yellow page servers. They allow agents to look up other agents by the services they offer.

Agents. All application agents reside in one or more areas.
**Communication protocols.** The following protocols are defined at the Communication management layer:

- **Registration**: for registering an agent as part of an area;
- **Advertisement**: for advertising agent services;
- **Search**: for querying about advertised services;
- **Service provision**: for requesting services from other agents;
- **Cooperation domain subscription**: for joining a cooperation domain;
- **Cooperation domain invitation**: for requesting an agent to join a cooperation domain.

**Ontology servers.** In global manufacturing applications the collaborative companies often use different ways to represent information, different application software packages and different ontologies. Therefore, an ontology server is needed to integrate the repository of ontologies as well as an inference and query engine and different translators. The ontology server structure and related mechanisms are initially domain independent. It becomes domain-specific when it is filled with domain-specific ontologies for a specific application, e.g., mechanical design, or production planning, etc. Of course, it is not difficult to define the structure and mechanisms for an ontology server, but it is extremely difficult to develop and complete an efficient ontology server for an application domain.

2.2.2.3. **Messaging management layer.** Messages are needed to support interactions among agents. Messages are generically defined either as request, reply or inform messages, where requests are used to ask for the provision of a service, replies to answer requests, and informs to notify agents without expecting a response. Since these messages are too ambiguous for the definition of interaction protocols, other more meaningful message types are derived from general definitions of messages. For example, an advertisement_request is derived from request for an agent to request to be registered (with a yellow pages agent) as providing some particular agent service. We are using low-level conventions named interaction protocols to offer a reliable and efficient transmission of messages. As an interface provider between the coordination and the network level (Fig. 1) we use version 4 of the package Voyager developed by ObjectSpace (www.objectspace.com) which provide message transmission between agents according to the particular network requirements.

3. **Prototypes and mechanisms for global application development**

Several mechanisms support global application development and implementation using the proposed architecture.
3.1. Agent-based mediator-centric organization

To better manage global distribution of tasks and sub-tasks across the global application, this one is first organized into a federation of agent groups/clusters using a mediator-centric organization mechanism [1]. Federation multi-agent architectures require a substantial commitment to support intelligent agent interoperability through mediators. A generic model for the design of mediators, based on the specification of various meta-level activities, was proposed and implemented in [19].

Mediators are collaboration agents specialized in grouping agents able to solve a particular task (regarded as a recursive sub-task of a higher level task) and to distribute and schedule the sub-tasks across the global application. Once a task has been assigned to the global-distributed application, mediators decompose it into sub-tasks (Fig. 4). Mediators can use brokering and recruiting communication mechanisms to find related agents for establishing collaborative sub-systems (also called virtual clusters), across the globally distributed partner organizations Fig. 7. Mediator agents coordinate activities both within clusters and across clusters [10]. Each sub-task is subsequently distributed according to the best solution plan. Mediators learn dynamically from the agent interactions and identify coalitions that can be used to establish

Fig. 7. Resource allocation and grouping through virtual clustering.
distributed searches for the resolution of tasks. Sub-tasks are then further decomposed, and allocated through negotiation among resource agents [19,20].

3.2. Virtual clustering mechanism

To work cooperatively, agents may form clusters that bond dissimilar agents into harmonious decision groups, Fig. 7. Multi-stage negotiation and coordination protocols efficiently maintain the stability of these clusters. Each agent has its individual representation of the external world, goals, and constraints. Therefore, diverse heterogeneous beliefs interact within a cluster through distributed cooperation models. Through the virtual clustering mechanism, agents can be dynamically contracted to participate in a problem-solving group (cluster). In the situation where the agents in the problem-solving group (cluster) are only able to partially complete the task, the agents will seek help outside their cluster. This results in sub-clusters being formed for sub-tasks. This process is repeated, with sub-clusters being formed and then sub-sub-clusters, etc. as needed, within a dynamically inter-linked structure which underlines the multi-resolution of our architecture. As the respective tasks and sub-tasks are solved, the related clusters and links are dissolved. However, mediators will store the most relevant links with associated task information for future re-use. This clustering process, as implemented, provides scalability and aggregation properties to the system. Multi-resolution virtual clustering is the main mechanism for task distribution over global application.

3.3. Multi-resolution

It is important to underline the dynamic structure of MRCA, which is a cluster of clusters of agents (and so on, recursively) collaborating at different levels, from general to specific, in solving a particular application. At the highest level of resolution, the agents are clustered over the distributed organizations involved, then down to departments, sections, and machines within these organizations. Agent coordination is managed both locally, within each of the involved organizations and as well globally, over all involved organizations to establish collaborations which allow solving the particular tasks. At any level of resolution for each resource, the order in which tasks are to be executed is established by a collaboration control mechanism. In manufacturing, if desired, this can allow continued use of group technology methods, Fig. 5 [22].

3.4. Partial agent cloning mechanism

Through this mechanism, resource agents can be partially cloned as needed for concurrent information processing. These clone agents can then participate in virtual coordination clusters, where agents negotiate with each other to find
the best solution for a production task. A detailed description of this mechanism can be found in [9].

3.5. Adaptation and learning

Two fundamental learning mechanisms have been implemented to enhance the system’s performance and responsiveness, with mediators playing an essential role in both mechanisms. First, a mechanism that allows mediators to learn from history is developed at the resource mediator level to capture significant multi-agent interactions and behaviors. Second, a mechanism for propagating the system’s behavior into the future is implemented to help mediators ‘to learn from the future’. A ‘learning from history’ mechanism based on distributed case-based learning approach was developed for capturing agents’ behavioral patterns at the resource mediator level and storing these in its knowledge base. Such knowledge is then reused for later manufacturing requests, through an extended case-based reasoning mechanism. The main purpose of ‘learning from the future’ is to modify promissory schedules at the resource-agent level for otherwise unforeseen perturbations and changes in production priorities on the shop floor. The forecasting process simulates the behavior of the virtual model that emulates the shop-floor activities. By partially projecting ‘unpredictable behaviors’ and agent interactions, the agent-based manufacturing system is able to correct its real-world model and provide more accurate plans [10].

3.6. Self-organization

Recently, we have developed a methodology for automatic grouping of agents allocated to sub-tasks into virtual clusters, by fuzzy modeling the inter-agent information exchange. This approach [7,21] considers the multi-agent system as a dynamical system in which agents exchange information while acquiring more knowledge about the assigned sub-task. Knowledge is reflected in the level of organization of the information across the multi-agent system, such that the sub-task is accomplished in an optimal manner. The Shannon Entropy measures the degree of order in the information spread across the multi-agent system (which reflects the degree of knowledge). To cope with the multi-dimensional aspects of uncertainty we use the Fuzzy Shannon Entropy. We consider the agent interaction being influenced by the “information field” which acts upon the agents governing the agent clustering in a manner which optimizes the goal achievement while minimizing the entropy. In other words, entropy minimization of the information spread across the multi-agent system, with respect to assigned task, leads to automatic formation of clusters of agents meant to cooperate on solving that particular task optimally under the particular restrictions of the problem. This automatic clustering
mechanism – implemented within the methods used by mediator agents, induces self-organizing capabilities to the MRCA opening perspectives towards fault-recovery and on-line re-configuration of the global solution in case of unexpected perturbing events. For a detailed presentation of the self-organization mechanism please see [6]. Using this method mediators are able to re-configure the MRCA on-line when unexpected changes occur, such as when another order, more critical, was received and resources have to be re-allocated or when a resource fails, transfer its workload over other available ones.

4. Task distribution over the global application

As already presented previously, once assigned, the task is decomposed into sub-tasks that are distributed across available resources by the interware, Fig. 4. Any entity involved in collaboration for solving a particular task at any level of resolution (global inter-enterprise or local, intra-enterprise tactical, operational/execution level) – is regarded as a resource. So, resources can be considered machines, departments, and sections within an enterprise – at the intra-enterprise level as well as whole organizations and multi-national corporations at the global inter-enterprise level. At a logical structural level, a resource agent, Fig. 7, represents each resource. So one distinguishes besides the physical resource level also a logical resource-agent (resource clone) level. Task distribution and re-distribution is achieved via the grouping of resource agents into virtual clusters, Fig. 7. For each sub-task a virtual cluster is formed, which contains agent-clones of the resources needed for that task’s accomplishment. One resource may be cloned as resource-agent in many virtual clusters. A mediator agent solves conflicts when the resource is allocated to more than one sub-task.

Dynamic virtual clustering of resource agents at the logical level is the enabling mechanism for self-organization. Resource agents are representing the resources as needed for each particular task. Each virtual cluster is dissolved as soon as its task has been accomplished. New clones and clusters are thus being created and dissolved for the tasks as needed. The task decomposition and virtual clustering processes can be recursive down through further levels of decomposition/resolution as sub-tasks are further decomposed and allocated to sub-organizational entities.

5. Application example: global supply-chain management

5.1. The scenario – global resource allocation

Here we illustrate how the architecture supports collaborative applications, using the example in Fig. 8. Notice the top-down flow of information (mes-
sages) through the vertical agent hierarchy. We assume that Customer “C” wants to order a particular product, say X and places an order for this product to its interface agent IAc. The interface agent IAc contacts the collaborative (yellow page) agent C1 who knows that knowledge agent KA1 has knowledge about enterprises who make the respective product. When being contacted by C1, KA1 informs the customer’s interface agent IA1 that Production Enterprise P produces the respective product at the closest specification with the ones desired, and the most convenient price. IA1 contacts the interface agent of the production manager at the Production Center of the Enterprise P, IAp placing an order for n products X with due date D. IAp alerts the Production Manager about the order through, for example, icon-blinking associated with specific sound (which can be e.g. via the screen of its cellular phone/pager or palm PC). At the same time, IAp contacts collaboration (yellow page) agent C2 asking “who knows about products of type X?” C2 knows that knowledge agent KA2 has such knowledge, so contacts KA2 who replies to IAp that C4 has knowledge on who can collaborate on product design. IAp contacts C4 (placing as well a relevant message on his user’s screen). C4 identifies agent KA3 as having knowledge on the design of product X. KA3 retrieves the respective knowledge from its associated knowledge base (DB1) and finds that product X consists of two parts of type Y and three parts of type Z, and as well that collaborative agent C3 can find out who can produce these parts. C3 is contacted by IAp and finds out that Factory 1 produces parts of type Y and

---

Fig. 8. Establishing global collaboration via the interware.
Factory 2 produces parts of type Z. At this point, C3 initiates creation of a collaboration template mediator (TM) agent [1] to manage cooperation between Factory 1 and Factory 2 on producing product X. TM creates in turn a dynamic mediator agent (DM1) which intermediates the bidding process from factories 1 and 2 via the virtual cluster V1. Suppose Factory 1 replies (via Iaf1) that it can deliver with specified due date all necessary parts Y. However, suppose Iaf2 responds that it cannot deliver the full order, but only a few parts with due date D, the rest later. Then, DM1 contacts Iaf2 asking it to find a solution for delivery of the remaining parts with due date D. Iaf2 contacts collaboration (yellow page) agent C5 asking for knowledge on who can deliver parts Z with due date D, etc. Once the order is confirmed with Factories 1 and 2 (and any other suppliers) another dynamic mediator (DM2) is created, to be in charge of the resource agents’ coordination. DM2 acts as an Application Agent, controlling resource agents RA1, RA2 and RA3 who produce part Y. For this, DM2 will form dynamic virtual clusters (as indicated in Fig. 7) – one for each task – which will group the resources involved in each task at a logical level.

5.2. The internet-enabled implementation

Fig. 9 presents the communication infrastructure for the supply-chain example. From this perspective, according to Sections 2.1–2.2.2.2, the involved organizations are grouped in areas managed by LACs which take care of the intra-organizational security and coordinated inter-organizational exchange of information.

For each collaborative application, the collaborative partnering organizations are connected via CDSs. The supply-chain application involves two distinctive aspects of cooperation:

- one in product ordering, between the customer and the multi-national corporation which takes over the order; CDS1 is used to facilitate the cooperation and negotiation between customers and the global corporation’s marketing service (e-commerce collaborative support);
- one in production, consisting of all the organizations participating in the fulfillment of the order; CDS2 facilitates cooperation between the factories and their parts and materials suppliers (business-to-business collaborative framework, or e-business).

At the highest level of application distribution, the virtual clustering mechanism transcends the local areas grouping organizations needed for collaboration on a certain domain. In our example, the clustering of organizations involves Factory 1, Parts Suppliers 1 & 2 and Materials Supplier 1 are clustered to deal with parts and materials supply negotiation needed for the fabrication of the ordered products). The collaborative interface agents CIAs manage the interfacing with the human managers involved in the global collaboration.
Collaborative agent YP1 works for the customer to find the best company who delivers the requested products, while YP2 works for the corporation to find suppliers for the specified order.

- The ontology server facilitates the information exchange between the different organizational entities involved in the two distinct cooperations, by providing translation support during the conversation among different agents through the CDSs.
- The HCA functions as a coordinator of Factories 1 and 2 for coordinating production planning.

6. Discussion and conclusions

This paper describes a high-level architecture for multi-agent coordination in global inter-enterprise collaboration that encapsulates 10 years of experience with distributed manufacturing systems. The strength of this MRCA stems from its versatility, generic layout, and self-organizing properties as well as the power of its coordination–communication platform and its domain independence.

There are many types of architectures that have been described for multi-agent systems [23–26]. These involve deliberative, reactive, or hybrid approaches that incorporate various mechanisms for task decomposition, formation of coalitions, and coordination of activities. Some of these architectures have proven very successful in particular applications.
What has been attempted in the architecture described in this paper is to combine recursive patterns of decomposition, coalition forming, and coordination into a “generic” dynamically self-organizing structure. In the work carried out so far, the desired patterns of activity have been demonstrated successfully experimentally, using simulation. In principle, the architecture is scaleable and upgradeable, but the extent to which this proves to be successful will only be determined when the architecture is applied to large real-life complex systems.

Acknowledgements

The contributions of the members of the Intelligent Systems Group (People-link at http://isg.enme.ucalgary.ca) are acknowledged with appreciation.

References