Proceedings of the Workshop on "Knowledge Grid and Grid Intelligence" (ISBN 0-9734039-0-X),
held at the 2003 IEEE/WIC International Conference on Web Intelligence / Intelligent Agent Technology,
October 13 - 16, 2003, Halifax, Canada, pp. 195 - 206

On the Architectures of Complex Multi-Agent Systems

Huaglory Tianfield *, Jiang Tian

School of Computing and Mathematical Sciences
Glasgow Caledonian University
70 Cowcaddens Road, Glasgow G4 0BA, UK

Xin Yao

School of Computer Science
University of Birmingham
Edgbaston, Birmingham B15 2TT, UK

** Phone: +44 141 331 8025, Fax: +44 141 331 3608, E-mail: h.tianfield@gcal.ac.uk*

Abstract

This paper presents a literature review on the architectures of complex multi-agent systems. Firstly, the concept of general architectures and three orientations of systems architectures, i.e., information flow oriented, control oriented and role oriented ones, are presented. Secondly, a literature review is presented upon the architectures of complex multi-agent systems in terms of control oriented and role oriented architectures. Finally, a novel autonomic architecture is put forward for complex multi-agent systems.

1. Introduction

1.1 General Architectures

A system has multiple perspectives. Specifically, a system has functions (i.e., services) as viewed from its external appearance; geometry as viewed from its internal construction; information flows/data relations as viewed from its flesh in the internal construction; behaviors (workflow, process, discrete-event dynamics and real-time dynamics) as viewed from its internal acting; and operational infrastructure as viewed from its subsistence.

We define architecture as the identification, description and specification of the decomposition inter-relationship among system components along the multiple perspectives and the performance of a system. Architecture is an effective means of tackling and reducing the complexity of systems.

While sometimes architecture appears in other terminologies such as structure, framework, and frame, they all have the same meaning, namely specifying the components in a system, the inter-relationship among these components of the system.

The structure of an organization defines the roles of various intentional components (actors), their responsibilities (in term of tasks and goals they are assigned), and resources (they are allocated), and defines how the activities of various actors are coordinated and how they depend on each other [20].

ICT platform architecture is the architecture of the generic resource layer, which describes the machines, networks, peripheral, operating systems, data base management systems, user interface frameworks, system services, and other resources that will be used as a platform for the construction of ICT system for the enterprise [2].

Software architecture involves the description of the components (from which systems are built), interactions among those components, patterns (that guide their composition and constraints on those components) [33]. Software architectures describe a software system at a macroscopic level in terms of a manageable number of subsystem/components/modules inter-related through data and control dependencies [20].

Frames can be seen as the answer to the question “what is going on here?”. Frames provide “framing” information about a particular class of interaction situations that will allow the participant to act appropriately. In multi-agent systems (MAS) context, frames are viewed as data structures that contain sufficient knowledge to structure the interaction for the individual [34].

Generally speaking, systems architectures have three orientations, i.e., information flow oriented, control oriented and role oriented ones. For instance, in software engineering, information flow oriented architectures have three classic models, i.e., repository model, client-server architecture, and abstract machine model. Control oriented architectures have two classic models, centralized control and event-driven control, which further have call-return model and manager model, and broadcast model and interrupt-driven model, respectively [1].

1.2 Architectures of Multi-Agent Systems (MAS)

MAS architecture is a typical system architecture that has intelligent agents as the system components and social communication as the interactions. Structuring the MAS architecture is the process in which the roles of agents and the relationships between agents are defined.

In a MAS, agents can undertake learning, perception, reasoning, judgment and decision in term of their knowledge about themselves and their environment. Moreover, agents can cooperate and coordinate by communication. All these functions depend on the architecture of the system. The architecture reflects the capability of the system to deal with complex problems, and determines operation mechanisms.

MAS architectures can be categorized in numerous ways according to different criteria. For example, in term of the mechanism, there are pipes-and-filters architecture, event-based architecture and layered architecture [20]. In term of their characteristics, MAS architectures can be classified into hierarchical, distributed, open, re-configurable, and mobile architectures, as well as fault-tolerant architectures.

However, these categories do not completely describe the common characteristics and the essence of MAS. In a general view multi-agent system architectures can be classified as information-flow oriented architecture, role-oriented architecture and control-oriented architecture inter of the orientation of MAS.

An information-flow oriented architecture focuses on the information flow in the system. As far as MAS are concerned, information flow oriented architectures reflect the interactions and communication in MAS, and are about the inter-agent communications.

In practice, an overwhelming majority of the architectures of MAS in literature is role oriented ones, which is based on the functional decomposition in application domains. Role-oriented architectures of MAS rely upon the knowledge of application domains. These architectures focus on meeting the generic functional requirements of domains. Therefore, role-oriented architectures of MAS are specific to different application domains. Suppose the general functions can be decomposed into many sub-functions. Different agents play different roles and these roles correspond to the sub-functions. Both mapping between agents and roles and that between roles and sub-functions are multiple to multiple. Role oriented architectures can be further classified as layered and flatted, and generic and non-generic.

Role oriented architectures have three major characteristics. Firstly, role oriented architectures are domain specific and are built on rich knowledge of application domains. Secondly, role oriented architectures basically follow the understanding that the overall goals of the system are decomposable into a set of separate functions and each of the functions can be fulfilled by the roles played by one or more agents, and that the interactions among the agents will be homomorphic to the relationships among the separate functions. Thirdly, the MAS built up by a role oriented architecture will be definitely cooperative, as the whole idea behind the architecture of the MAS is the top-down functional decomposition of the overall goals of the system.

Control-oriented architecture is to represent the intensive control flows of the system, especially to deal with the unexpected events in the environment based on agent's perception, and to make fuzzy decisions in an autonomic way. Control-oriented architecture has rigid control hierarchy, namely, the system can be divided into various layers to achieve the task. Each layer is controlled by its higher layer. However, so far there are only a very few control-oriented architectures of MAS. Maybe the information-flow oriented architectures and role-oriented architectures have already met the current requirements upon MAS. Basically the design of architectures is necessary only when the systems are sufficiently complex. If information flow oriented architectures and role oriented architectures are for complex MAS, then control oriented architectures are mainly aimed at very complex MAS. It may be the case that most of the systems dealt with by MAS so far are not very complex and thus the use of control oriented architectures was not particularly needed. Another view is that information flow oriented architectures are mainly for information flow extensive systems, role oriented architectures are in line with structured system analysis and design methodology, and control oriented architectures are for control intensive systems. In the existing research of architectures of MAS, most of the analysis and design of MAS has implicitly followed the structured system analysis and design methodology, and most of the systems concerned are not heavily control intensive. With the great challenge from large complex systems, proper control-oriented architectures have to be devised.

2. Architectures of MAS ----- A Literature Review

2.1 Control-Oriented Architectures of MAS

Hierarchical architecture is a typical control-oriented architecture of MAS. The hierarchical solution to a global problem is built up of modules which form stable sub-solutions, and allow one to construct a complex system out of less complex components [11]. Classical hierarchical architecture lies in the fact that complex tasks can be decomposed into several levels of abstraction and assigned to distinct hierarchical levels. The specialization of each level facilitates the implementation and management tasks [32]. Hierarchy is an essential paradigm to tackle complexity. Hierarchy well matches

human organization/management procedures to solve complex problems, and provides effective feedback and control ways.

However, a major drawback of hierarchical architectures is that they can be overly rigid and consequently difficult to adapt to unanticipated disturbances [32]. Typical hierarchical architectures are holonic architecture, nested architecture.

Holonic multi-agent system architecture is suitable for dealing with complex tasks that can be divided into a set of subtasks, for each of which an agent can be assigned to [12]. There are N layers in a holonic architecture, and at the same layer there are numerous distributed agents. Each layer is controlled by its higher layer. Moreover, the number of layers depends on the complexity of the task to be resolved.

The essence of nested architecture is also hierarchy. In a nested architecture agents at lower layer are nested to higher and from lower layer. A nested hierarchy system is defined as an independent agent that has nested agents, providing a group of actions at each level. The lowest level in this hierarchy consists of a group of agents that carry out only simple actions [33]. The agents that are at the same nesting level communicate with each other through a shared system functional history. Each complex action comprises various simple actions. When an agent carries out a complex action, it is composed of various simple agents each of which carries out the individual actions [33]. So the occurrence of the execution of the complex action is stored in the system functional history of the main agent.

A three-level nested-workflow is proposed to manage the inter-related workflows of the Virtual Organization [46]. It is a nested workflow, and corresponds to the federation agent workflow for virtual organization. It consists of super-federation level, federation level, and agent level. A level is nested to its super and from lower level.

2.2 Generic Role-Oriented Architectures of MAS

2.2.1 Re-configurable Architecture

Re-configurable architecture is based on hierarchical architecture, and characterized with mediator agents and recovery agents at every layer. When some error happens in the system, the system will find the error and recover to establish a new architecture. This architecture possesses following characteristics [32].

- The reconfigurable architecture is characterized by loosely connected agents.
- Error recovery can be handled at different levels.
- The hierarchical treatment of errors facilitates the design of sensor strategies. And
- Re-configurability of cluster of agents provides flexibility and adaptability.

For example, there is a hybrid hierarchical intelligent-agent-based system with error recovery agents [32]. This architecture is based on the hierarchical control and MAS architectures. The production agents are autonomous, namely the last decision on the execution of a task is made by the production agent itself. The diagnostic and error recovery functions are delegated to specialized external agents. Production and error recovery agents will be reconfigured into temporary clusters, and the reconfiguration will always preserve a generic hierarchical structure. So agents can only be reconfigured across their particular hierarchical level and focus on adaptability to disruptions at that level. This MAS consists of three structures, i.e., production structure, recover structure and mediator agent structure.

2.2.2 Open Agent Architecture (OAA)

The open agent architecture (OAA) is a research framework for constructing agent-based systems. The software services are provided through the cooperative efforts of distributed collection of autonomous agents. OAA is a domain-independent framework, from which a wide variety of systems can be built. The communication and cooperation between agents in OAA are brokered by one or more facilitators. OAA minimizes the effort involved in creating new agents and wrapping legacy applications, which are written in various languages and operating on various platforms. In OAA the components include facilitator, application client agents, meta-agents, and user interface, and OAA has general mechanism for questing by triggers [8]. The main characteristics of the OAA are openness, distributedness, extensibility, mobility, collaborativeness, multiple modalities, multimodal interaction, etc. [30].

2.2.3 CORBA

Common object request broker architecture (CORBA) is a typical distributed MAS architecture. CORBA specifies a way in which developer can achieve seamless integration and interoperability of distributed objects [19]. CORBA is composed of client, object implementation and object request broker (ORB). The client is the entity that wishes to perform an operation on the object; the object implementation is the code and data that actually implement the object; and the ORB is responsible for all of the required mechanisms to find the object implementation for the request, to prepare for the object implementation to receive the request, and to communicate the data that makes up the request.

2.3 Domain-Specific Role-Oriented Architectures of MAS

2.3.1 E-commerce

A framework for Intelligent-agent-based e-commerce consists of market maker, transaction level, activity agents level and other supporting tools. These levels are used to support e-commerce at three different levels, i.e., market, contract and activity, respectively [22]. Market maker serves as a window between the user and other agents. It can select the proper trade type based on its internal knowledge and activates proper agents at the lower level. At transaction level there exist multiple agents, and each of them is responsible for performing a certain type of transactions. Barter, bargaining, bidding, auction, clearing, contract agents correspond to six trade types in electronic commerce, respectively. Activity agents perform certain activities for different trade agents, such as search agent, generator agent, evaluator agent, decision agent, transaction agent and service agent. Other supporting tools are useful in supporting e-commerce, e.g., the electronic meeting agent may be used to support bargaining.

A MAS with object linking and embedding (OLE) was used in shoe industry. OLE automation server is utilized to transfer the Internet protocol data into the server side agent. Moreover, the OLE automation server motivates the client agent automatically if needed. The architecture of MAS using OLE server and object request broker (ORB) consists of client agent, server agent, storage agent, monitor agent, selection agent and negotiation agent [19].

A personalized brokering multi-agent system comprises many components such as the mobile buyer, the buyer's agent, the multi-attribute resource intermediary (MARI) server, the Wherehoo server, seller agents, and physical sellers [41]. User and sellers interact with their agents through Internet. Buyer agent and seller agents undertake brokering through MARI server. MARI server is a generalized platform for the specification and brokering of heterogeneous goods and services. MARI server makes it possible for both buyers and sellers to specify relative preferences for the transaction partners, as well as for the attributes of the product in question, making price just one of the many factors influencing the decision to trade. Wherehoo is a search engine optimized for location-specific searches, with a web-based front-end for the addition of new records and maintenance of existing records.

Time-Bound Negotiation Framework is one where agents negotiate through messages with commitment duration [21]. When a message has a zero-length commitment duration, the agent has no commitment, namely zero-time commitment. On the other hand, when a message has an infinite-length commitment duration, the message has eternal commitment. When a message has a finite-length commitment duration, the message is valid for the specified duration.

Online Dynamic Bargaining System Framework consists of three agents, i.e., dynamic price-issuing agent, pattern generalization agent, and pattern-matching agent, two settings, i.e., front-end settings and back-end settings, and a web interface [23]. Through the web interface, dynamic price-issuing and pattern-matching agents operate in front-end settings, interacting with buyers in real time. Pattern generalization agent operates in back-end settings, processing transaction data periodically to generate bargaining patterns in a batch mode.

The architecture of federation agent workflow for virtual organization consists of two levels, i.e., virtual level and virtual support level [46]. Virtual level includes federation organization (agents and low-level federation), class repository (such as passive objects, interaction and the common rules), and communication media. Virtual-support level includes run-time support mechanisms and behavior service repository. Behavior is managed and performed by run-time support mechanisms which are for driving, monitoring and managing the virtual organization.

2.3.2 Logistics

A MAS for Coordination of Just-In-Time Production and Distribution consists of three types of agents [9], i.e., consumer agents, redistribution agents and producer agents. Consumer agents continuously make predictions of future consumption by the corresponding consumer, and monitor the actual consumption, and send information about this to their redistribution agents. Redistribution agents continuously make prediction for the cluster and send these to the producer agent, and monitor the consumption of resource of the consumer in the cluster. It is for each cluster of consumers. Producer agents receive predictions of consumption and monitor the actual consumption of consumer through the information it receives from the redistribution agents.

An AGENT-DOSS (Distributed Omni-Search Strategy) System for Vehicle Routing Problems aims at the problem how to economically assign vehicles to deliver productions to the user. This system is implemented using DOSS [42]. The MAS consists of two main agents, i.e., auctioneer agent and vehicle agent, and various classes.

Each agent has environment class. This class maintains the necessary information for communicating with other agents. It maintains the information about all the other agents in the system and the spaces in which they are present. In the class of communication, all communication between agents is implemented. In skills class, each agent can obtain skills either by being programmed or by learning over a period of time. The skills, which are possessed by the agents, are unique with respect to an agent and are maintained in this class. In identity class, the

identity of an agent is maintained. In customer class, information about the customers or depots with respect to the agent is maintained in this class.

2.3.3 Design and Manufacturing

A design of structured objects (DESO) Framework consists of an agent template, multi-level communication model, and generic multi-agent system architecture [37]. Firstly, the agent template has 3-layer architecture, i.e., mental layer, behavior layer, and communication layer. Secondly, multi-level communication model has four levels, i.e., the first level defines the high-level strategy and the negotiation structure between agents; the second level defines message semantics, request type and content and is based on KQML; the third level defines the standard mechanism of logical component interactions, is used to map data and operation by converting events and message to the set of KQML performatives; the fourth level is the low level communication stacked by transmission control protocol/internet protocol (TCP/IP). Thirdly, the generic multi-agent system architecture is DESO high-level communication architecture. It supports coordination, synchronization and collaboration of agents and non-agent software components. It consists of CCDA (component integration and concurrent configuration design advisor), facilitators, Database & knowledge base, DESO components, CAD tools and external CAD tools. Moreover, in CCDA, there are design agents, advisor agents, and constraints.

A Cooperative and Collaborative Framework for design, namely, an assembly co-design framework, consists of three layers, i.e., application layer, core system layer and communication layer [45]. The application layer, namely upper layer, is for specific tasks: designer, planner, evaluator, or a system console to be used only by the manager. The core system layer, the intermediate or control layer, is concerned with the correct treatment of all kind of objects in a specific assembly design and planning session. It performs the core of the computer supported cooperative work. The communication layer, the lower layer, is responsible for providing the correct message handling among the users. Moreover, each agent in the framework is internally designed as a multi-layer service, where each layer provides a refined service to its superior layer. Different kinds of agents interact through the network in order to provide a cooperative/collaborative service to a certain class of applications. The overall system consists of many kinds of cooperative agents: design agent, planning agent, evaluation agent, simulation and animation agent, user agent, manager agent, and coordinator agent.

The agent-based collaborative service support system provides a service to the manufacture firms within an enterprise information network [44]. The system consists of four components, i.e., the user interface, the task management system, the information update and the system database. The user interface is designed to include the push technology concept. It is the communication "outlet" between the users and the network system. The user can navigate through the system with ease, preferably using only "point-and-click" procedures. The task management system is the "brain" of the whole system. It is characterized by the integration of a rule-based inference mechanism with object-oriented technology. The rule-based inference mechanism unit produces the list of basic tasks required to meet the service request, based on the result of the reference process; as the inference mechanism becomes part of the object-oriented programming environment, the list of tasks being generated is sent directly to the task control subsystem; an "agreement" is worked out by building the rules for determining which agent is responsible for which job, regarding which fundamental task is meat for which agent. The information update enables system administrators to update the information in the repository that stores the facts, rules and templates for the expert system, and data-files for the agent objects. The system database is the central data storage system for keeping the data which are to be used by the task management system. Information update and system database components work together to ensure that the most updated and accurate information be kept in the system database.

A SCEP (supervisor, customers, environment, and producers) Multi-Agent Model is a reactive scheduling system. It consists of customer agent, producer agent, supervisor agent and environment [4]. It is a distributed model, and introduces an indirect co-operation between manufacturing order agent (representing the set of customers, denoted C) and machine agent (representing the set of producers, denoted P). This cooperation is performed synchronically via the environment E and is controlled by the supervisor agent S. The scheduling is achieved after a defined number of cycles, and each cycle corresponds to an activation of customer agents. The environment E is composed of a set of objects O (that evolve according to the influence that they receive from the customer and producer agents). Each customer agent has an intervention domain, and each producer agent has an intervention domain.

A MAS with Supervisor for production scheduling and control consists of scheduling supervisor agent, part agent, resource agent (machine, operator, and transporter), local spreading centre agent, and transport manager agent [7]. Each type of agent is equipped with the information and the decision-making modules to carry out its own task. The information module contains data and the structure of information, and the decision-making module contains the agent's decision-making logic. Two phases, i.e., scheduling and real-time control phases, are involved. In

scheduling phase, the mechanism in this architecture used for the scheduling of the system is the negotiation driven by the intervention of a supervisor. The negotiation process is not activated in parallel, but in a sequential way. It is the supervisor agent to select the job with the highest priority among the applicants. The negotiation modes are similar to those in the contract-net models. The supervisor agent negotiates with the resource agents, trying to optimize an objective function. The single machine will sell a service in order to fill its future workload. In the real-time control phase, real-time control is accomplished by the part agent which is the direct user of the service supplied by all the machines assigned during the scheduling phase. In this system, the function of local spreading centre agent is to lighten the communication traffic among the agents. They route the task announcement messages to the resource agent. Moreover, the supervisor is given the faculty to modify the assignments built up during the scheduling phase. Whenever critical events or conflicts among jobs occur, the supervisor agent can relax the previous contracts and carry out a new scheduling starting from the jobs which currently have the highest priority.

A Market-Like MAS is based upon information processing and decision-making process distributed in a market-like hierarchical system. It is inspired from the negotiations carried out in a marketplace where sellers aim to sell their goods at the highest price, while buyers try to buy goods of the desired types and of the required quality at the lowest prices [7]. This system consists of two main types of agents, i.e., part agent and resource agent. Part agent is the control module of a production batch or a single manufacturing job. It contains all manufacturing data, the main managerial information and the decision-making rules. Resource agent is the logical representation of any of the production resources of the shop-floor system. It collects all the manufacturing and managerial information related to the negotiation tasks of the resource itself. The control strategy is carried out through a contract-net protocol with a combination of price and objective mechanism. On the one side, each part agent tries to fulfill the processing requirements to achieve its own set of objectives. It is given a certain amount of currency representing its own purchase capacity in order to acquire the needed service from the single production resource. On the other side, a production resource sells a service at a fixed price, according to its current capacity and the global state of the system to maximize its individual profit or system performance criteria.

A Three-Dimension Framework for chemical plant is along three dimensions, i.e., cooperative development, knowledge sharing and computer-aided support [15]. Cooperative development of interface, protocols and architecture is domain-independent. This means that the interface should allow users to create quickly new systems of software agents, and these agents communicate on the internet regardless of application-specific knowledge representation. Most agent systems can be communicated by knowledge query and manipulation language (KQML). Knowledge sharing among systems that maintain their own specialized knowledge bases and reasoning mechanism can be accomplished to represent domain-specific knowledge. KQML provides a standard frame for knowledge interchange, and knowledge interchange format (KIF) plays a role of the contents as a neutral format. Computer-aided support for the negotiation and decision-making that characterizes concurrent engineering can be achieved by algorithm for decision management among different agents. The algorithm should be able to track design changes and their relations.

A hybrid MAS for motor gearbox test control consists of eight agents which aim at different tasks to be accomplished [40]. There are two basic agents, i.e., the supervisor and the speed profile agent. The supervisor agent is to maintain time signals, start and halt the whole system and manage emergency signals from all the other agents. The speed profile agent is providing target values such as look ahead along speed profile for the control task. And the speed profiles are arranged in database. Moreover, the basic parameter agent and the basic controller agent supply the simulation environment. The basic parameter agent manages parameters for the motor gearbox model and the environment simulation. Its main task is to simulate the change. The basic controller agent contains a state detection mechanism which decides, depending on target values and state variables of the simulation, what control strategy has to be taken and which subsystem will control the motor and gearbox.

ComPASS (competence promoting multi-agent system for support) was developed to assist operators in the process of locating and remedying malfunctions occurring at a CNC machine-tool [27]. ComPASS consists of two deliberative, adaptive, and loosely coupled agents, i.e., horizontal agent and vertical agent. The horizontal agent searches along the functional dependencies between component parts and assemblies of a machine. And the vertical agent searches for fault dependencies at a single assembly or component. Both agents have relevant knowledge, use diagnostic strategies, and offer solutions to machine operator, respectively.

Manufacturing agent-based emulation system is an open framework for design and analysis of discrete manufacturing systems. The MAS has two system paradigms, i.e., distributed agents and synchronous collaboration [17]. Distributed agents include customer agent, process centre agent, process agent and stack agent. When the customer agent sends requests, or pulls signals, the receiving agent propagates the signal upstream to other agents in the process line with the goal to “pull” the resources required for completing the product.

2.3.4 Traffic Management

A MAS is designed for traffic congestion management [25]. It consists of two interacting, real-time problem-solving agents, i.e., freeway agent and arterial agent. Both agents are decision-support systems for a traffic operations center, and communicate with each other through a fast TCP/IP-based real-time protocol. Freeway agent supports incident management operations for a freeway sub-network and interacts with a human operator at the traffic operations center of a freeway management agency. Arterial agent supports operations for the adjacent arterial network, and interacts with an operator at the local city traffic operations center.

TRYS is an agent-based environment for building intelligent traffic management system (ITMS) applications for motorway networks [16]. The typical TRYS is the integrated TRYS (inTRYS) and TRYS autonomous agents (TRYSA2). InTRYS system relies on a set of knowledge-based traffic control agents, each of which is responsible for traffic management in one area. It receives local control proposals from the traffic control agents, resolves conflicts between them, and sends the resulting globally consistent local signal plans back to the traffic agents. TRYSA2 system is controlled by autonomous traffic agents which coordinate laterally, based on a mechanism called structural cooperation. And control devices "belong" to certain agents, and the corresponding mutual dependence provides a potential for cooperation.

2.3.5 Domestics

A MAS was designed for intelligent buildings [13]. In this system, each room contains an embedded agent, which is responsible, via sensors and effectors, for the local control of that room. All embedded agents are connected via a high level network, thereby enabling collaboration or sharing of information.

A MAS for the Management of Home Device Network consists of several agent spaces and many agents. It is controlled by Space Interconnection while each Agent Space controls the home device by many agents. All Agent Spaces broadcast all operation using IEEE1394's asynchronous stream. Agent spaces are interconnected to each other based on tuple space model. An agent can autonomously manage a specific home device or several other agents using the agent space and interconnect. All agents are loosely coupled with each other, because one agent can easily obtain information of another agent by reading the information from the agent space, and agents can also announce their information to all the other agents and advices by including it into the agent space [29].

A Mobile Agent Space System (MASS) for Home Device consists of MASS place, message server and Agent Proxy server [43]. Agent Proxy Server stores various agents, thus MASS Place or MASS Client can load them via network. And it can provide the appropriate authorization for users. Message Server is responsible for sending message to the MASS Place. MASS Places must register to the Message Server if they want to join the space. Both MASS Clients and MASS Places can connect to Message Server to obtain the information from other spaces by exchanging information between two Message Servers.

A Collaborative Society Framework was developed in water supply domain. It applies knowledge-engineering philosophy to specific scenarios [10]. It consists of a number of task-specific agents such as information agent, constraint agent, data mining agent, and predictor agent. All of these agents collaboratively manage the decision-support tasks and share the Decision Enabling Database. In this framework, the Interface agent accepts tasks from and supplies solutions to the user in a readily understandable format. The Information agent is responsible for the internal description of any given task and ensuring the Data Warehouse agent responds to the needs of the other agents. Moreover, this framework differentiates between data exchanges, for example differentiating the transfer of data from the databases to the task specific agents, and agent communication. Agent communication, which is more encompassing than data exchange, encapsulates the information of data exchange between any of the agents and databases.

2.3.6 Military

A Simulation Based on Software Agents and the High Level Architecture consists of different Participant Simulated theatre under the monitoring of different command and control information systems (C2ISs) [26]. High level architecture is a software interoperability framework evolving under the guidance of the Defense Modeling and Simulation Office Architecture Management Group. It provides a set of application programming interfaces (APIs) for data interchange service, pre-runtime templates and tools for reconciling data exchange, and rules for proper use of these services and rules. Simulated theatre is a digitalization of the battlefield of a participant. These different theatres interact through the high level architecture. Moreover, in every theatre, there are two types of agents, i.e., event-agents and participant-agents. The event-agent materializes the update in the simulated theatre with a specific symbol. And it triggers the condition-action template that corresponds to the event. The participant-agent implements the actions of the events that the event-

agent has detected. Several participant-agents are required to satisfy the requirements of an event.

A command agent system of object architecture for command and control in simulations has six main components, i.e., Comms, Collector, Planner, Promulgator, Recognized Picture, and Plan [28]. Comms provides communications facilities allowing the agent to exchange information (orders, report, and requests) with other agents. Collector encapsulates the G2 process, covering data collection, data fusion, construction and maintenance of the recognized picture and intelligent assessment. Planner encapsulates the G3 process, covering decision-making and planning. Promulgator encapsulates administrative processes for managing the output of the agent. It also handles the reporting cycle and promulgates reports/requests on demand. Recognized Picture encapsulates the agent's knowledge of the outside world. The Planner does its planning on the basis of the Recognized Picture. Plan is namely the planner's output. It defines the tasks that are to be designed to subordinate agents in order to achieve this agent's mission goals.

2.3.7 Information Retrieval and Multimedia

An Agent-based e-catalogue framework consists of the user agent, the facilitator, and the resource agent [24]. The user agent connects the system to the client. The resource agent is responsible for retrieving data from a particular resource. The facilitator performs routing, resource discovery, data collection and translation. The facilitator comprises five components, i.e., control agent, broker agent, category model agent, virtual catalogue agent and domain agent. Virtual catalogue agent is applied to manage the user profile and dynamically construct a personalized virtual catalogue; broker agent constructs a directory information tree (DIT) to maintain the resource-level metadata and category-level metadata.

An Intelligent Retrieval System with distributed heterogeneous data sources is composed of five agents, data sources, and a user profile base [36]. The agents include Intelligent User Information Agent, Query Enhancing Agent (enhancing user's query based on the user profile), Search and Routing Agent (sending and retrieving information from distributed heterogeneous data sources), Filtering Agent (filtering the raw data from SRAgent), and Analysis and Synthesis Agent (using the filtering information to enhance decision-making).

Agent-based imagery and geospatial processing architecture (AIGA) is a novel approach for developing and supporting large-scale query-driven multimedia information systems [38]. AIGA uses a two-level architecture. At the bottom level is an AIGA domain, and at the top level, AIGA domains are connected to other AIGA domains by connecting together Collaboration Switch to other Collaboration Switches (of other AIGA domains) via a spanning tree. An AIGA domain consists of a Collaboration Switch, analysis agent, and IGS (imagery and geospatial process system) agents. Collaboration, communication and knowledge sharing within AIGA domain are achieved by reading and writing AIGA pages in a common shared page space. Each AIGA page is composed of a Query, a Baseline Representation, the Computational Steps (which are necessary to provide results to this query), and a Processing Strategy (which configures the computational steps into the appropriate sequence and takes advantage of any opportunities for parallelism). In this system there are three types of AIGA agents, i.e., analysis agent, collaboration agent and IGS agent. Analysis agent serves as the primary interface between AIGA users and the underlying system. Each analysis agent has the ability to manage the process of analysis, and to form and control distributed search and query. Collaboration Switch serves as a resource agent for the other agents in the system. It provides a "yellow page" of other AIGA services. The entire class of IGS agents is capable of locating, indexing, processing, and transmitting geospatial data.

2.3.8 Software Engineering

A Mobile Agents system is designed for checking software engineering documents [39]. This system carries out incremental consistency checks between software engineering documents. This architecture of the system expands upon development of static consistency checker by capitalizing on the distribution of documents, consistency rule, and consistency links. It includes domain agents, resource interface agents, and mobile consistency checking agent.

Hector's System is designed to provide the infrastructure to control parallel programs during their execution and monitor their performance. It consists of master allocator, slave allocator, and local message-passing interface tasks [35]. In the system, the master allocator plays the central decision-maker and control process roles. The slave allocator, i.e., a distributed agent, is running on each candidate platform. Slave allocator gathers performance information from the tasks and executes commands issued by the master allocator. Message-passing interface programs are linked with a special library to interface with the Hector runtime system.

2.3.9 Network Management and WWW

Agent-based community oriented routed network (ACORN) architecture is a distributed MAS architecture for the search, distribution and management of information across networks [6]. It consists of server agents (such as Main server agent, Directory server agent, and Anonymity server agent), cafe (dynamic café and

static café), user information, and fat/thin agents. The user enters the ACORN through web browser and links to other ACORN by Network.

A MAS for Active Networks is a new networking paradigm with advanced services such as application-aware routing, information caching, packet filtering, encryption, compression, data format change [31]. The architecture of this system comprises four agent classes, i.e., request agent, quota agent, messenger agent, and configuration agent. Request agent (request agent) is responsible for periodically gathering and aggregating the requests from the end hosts. Quota agent is responsible for collecting information about resource availability and pricing within the domain. Messenger agent is the interface of the MAS with other domains. Configuration agent is responsible for configuring the individual sessions, by computing a Labeled Switched Path going through the nodes that satisfy the requirements of each session (in term of QoS parameters and price).

A Mobile Agent Platform (MAP) for network management is used for the development and the management of mobile agents [54]. It is compliant with the mobile agent system interoperability facility (MASIF) standard. This system consists of region registration, i.e., MAFFinder, and many MAP servers, i.e., MAFAgentSystem, through object request broker (ORB). MAFFinder provides the methods needed for locating and identifying agent and agent system by considering a central component for their registration. MAFAgentSystem is concerned about the management, the transfer and the execution of agents.

A MAS of managed domain in multimedia networks has two levels, i.e., domain level and system domain [5]. Each managed domain is under the control of a high-level intelligent agent, referred to as the domain agent. The lower architecture layer is controlled by a set of intelligent agents, referred to as switch agent. Switch agents are located at every ATM switch and monitor the switch behavior (e.g. buffer queue length). They automatically adjust appropriate thresholds depending on directives received from the domain agent. Since switch agents have partial knowledge of the system, they only act on behalf of the domain agent to collect and filter pertinent state information.

A General MAS for Intelligent Websites consists of user, employee and four classes of agents, i.e., customer (human agents), personal assistant agents (software agents), website agents (software agent) and employees (human agents) [18]. Personal assistant is involved as a mediating agent in all communication between its own user and all Website agents. Website agent communicates not only with all Personal assistants, but also with each other and with employee. Customer only communicates with his/her own Personal assistant, and it serves as an interface agent.

2.3.10 Healthcare

A multi-agent system is designed for monitoring medical protocols [3]. The monitoring services are provided by a secure communication interface and some autonomous system agents (ASAs). The ASAs include certification agent, supervisor agents, and the medical protocols server agent (MPSA). Certification agent acts as a certification authority. Supervisor agent track interactions between specialized domain agents (SDAs) and verify their validity. MPSA performs two main system services. MPSA distributes the list of available medical protocols to SDAs depending on their hospital roles, and it also distributes medical protocol specifications to supervisor agents. The monitoring process carried out by ASAs is divided into two phases. The first phase involves agent certification, agent addresses distribution and medical protocols delivery. In this phase, certification agent is the central component, and all agents must contact with certification agent to obtain their certificates. MPSA firstly contacts with certification agent, secondly the certification agent stores the MPSA address. Supervisor agent contacts with certification agent in order to obtain the MPSA address, and it allows us to dynamically distribute the system monitoring load depending on the number of supervisor agents. SDA contacts with the certification agent through an inter-agent for obtaining the MPSA address. The second phase corresponds to the medical protocols monitoring process. In this phase the inter-agent contacts with MPSA for obtaining the list of available medical protocols, and the SDA decides which medical protocol has to be executed. When all SDAs involved in the medical protocol have contacted with the MPSA, the MPAS assigns a supervisor agent for monitoring the medical protocol execution, and delivers to inter-agents the SA address.

A Reasoning MAS of LifeStyle Assistant was designed for elder independence. It consists of device layer, adapter and functional layer [14]. Device layer comprises both sensor and actuator. Adapter layer translates device message. Functional layer comprises four categories of capabilities, i.e., sensing, situation assessment, response planning, and response execution and actuation. Situation assessment predicts ramifications based on evidence. Response planning creates general response plan based on situation. Response execution and actuation talk to devices, format the presentation, send it to the device, and monitor for its successful delivery. Moreover, situation assessment includes clustering, validating, and intent inference. Clustering combines multiple sensor reports into a single event. Validating increases confidence of patterns, eliminates false positives, and weighs competing

hypothesized patterns. Intent inference puts multiple events together, and infers goals of actors.

3. A Novel Autonomic Architecture of MAS

The hierarchical decentralized control architecture (HDCA) [47-49] is a unique architecture of large complex systems. The distinctive features of HDCA include (a) the identification of the intrinsic attributes of large complex systems, (b) the distributed and nested structuring of the perception-decision links, (c) the constructive definition and formulation of hierarchical levels, and (d) the formalization of the architecture. These are far-reaching for large complex systems architecture research.

HDCA will be able to endow a system with advanced intelligence, including high degree autonomy on the system's own activities and high degree adaptation to the external hostile environment, etc. Such advanced intelligence can be typically illustrated by the intelligent machines enabled by HDCA, where machine is of a very broad meaning, such as machine-tool, grinder, vehicle, digger, transporter, loader, reactor, distillator, especially robot. Intelligent machines can emulate the high-level functions of human beings such as perception, recognition, logic reasoning and decision-making, etc. For instance, an intelligent machine-tool enabled by HDCA would be able to automatically recognize fixtures and parts, automatically plan machining sequences, automatically execute the sequences, and what is more, be able to automatically detect/diagnose and correct the faults, resume to normality and re-plan machining sequences while faults occur in the halfway execution of the originally planned machining sequences.

Based on HDCA, here a novel autonomic architecture of MAS is put forward [50-51]. It is a random holonic multi-agent architecture. Firstly, as to the multi-agent based encapsulation, each loop in HDCA, composed of perception cube and decision spheroid, can be realized by one or more agents. An agent is an autonomous entity that is packaged of a set of capable computational entities, three of which, i.e. for internal scheduling, problem solving and social communication routing, respectively, are normative and others are optional [52-53].

Secondly, as to the holonic property, the agents at a level provide the agents at the lower level with guidance, control, goal direction and constraints [50]. Under the holonic multi-agent architecture, the autonomy of agents is relative and never absolute, called holonic autonomy. The scopes that the agents at a level can autonomously act are specified by the agents at the upper level, called the spaces of autonomy. Within the corresponding space of autonomy, the agents are fully autonomous, and outside the space, the agents from this space are nested to an agent at the space of greater radius.

Thirdly, as to the randomness, the inter-level connections, in both top-down and bottom-up directions, are discrete event driven [47-48]. The randomness between two adjacent levels in the architecture is manifested in two ways. On the one hand, a level gets the statistic perception of the world from its adjacent lower levels. On the other hand, a level receives and executes the commands from its adjacent higher level in an opportunistic sense. For instance, a traffic radio station broadcasts to on-board receivers the advisory information of traffic jams and throughways, but vehicles do not certainly do as advised. While most vehicles may do so, some may not. This forms an opportunistic situation for the execution of the commands from its adjacent higher level. In some cases the inter-level connections are certain in their structures, but their contents, i.e., the commands from higher to lower levels or vice versa are linguistically fuzzy and stochastic. Random hierarchical architecture is very complicated and challenging.

4. Summary

The above overview shows that at least two important aspects in the different architecture studies need breakthroughs. Firstly, existing role oriented architectures of MAS are too dependent upon the domain knowledge and their generality is very limited. Secondly, control oriented architectures of MAS and their applications are very few and need to be significantly expended.

References

[1] I. Sommerville. Software Engineering, 6th Edition, Addison Wesley, August 2000.

[2] Aerts, A. T. M., Szirbik, N. B. and Goossenaerts, J. B. M. (2002). A flexible, agent-based ICT architecture for virtual enterprises. Computer in Industry, Vol. 49, 2002, pp. 311-327.

[3] Alsinet, A., Béjar, R., Fdernàndez, C. and Manyà, F. (2000). A multi-agent system architecture for monitoring medical protocols. Proceedings of Agents 2000, Barcelona, Spain, pp.499-505.

[4] Archimede, B. and Coudert, T. (2001). Reactive scheduling using a multi-agent model: the SCEP framework. Engineering Applications of Artificial Intelligence, Vol. 14, 2001, pp. 667-683.

[5] Boutaba, R., Iraqi, Y. and Mehaoua, A.. A multi-agent architecture for QoS management in multimedia networks. Journal of Network and Systems Management, Vol. 11, No. 1, 2003, pp. 83-107.

[6] Carter, J., Ghorbani, A. A. and Marsh, S. (2002). Just-in-time information sharing architectures in multi-agent system. Proceedings of AAMAS, 2002, Bologna, Italy, pp. 647-654.

[7] Cavalieri, S., Garetti, M., Macchi, M. and Taisch, M. (2000). An experimental benchmarking of two multi-agent architectures for production scheduling and control. Computer in Industry, Vol. 43, 2000, pp. 139-152.

[8] Cheyer, A. and Martin, D. (2001). The open agent architecture. Autonomous Agents and Multi-Agent Systems, Issue 4, 2001, pp. 143-148.
[9] Davidsson, P. and Wernstedt, F. (2002). A multi-agent system architecture for coordination of just-in-time production and distribution. Proceedings of SAC 2002, Spain, pp. 294-299.
[10] Davis, D. N. (2000). Agent-based decision-support framework for water supply infrastructure rehabilitation and development. Computers, Environment and Urban Systems, Vol. 24, 2000, pp. 173-190.
[11] Fischer, K. (1999). Agent-based design of holonic manufacturing systems. Robotics and Autonomous Systems, Vol. 27, 1999, pp. 3-13.
[12] Gerber, A., Kammenhuber, N. and Klusch, M. (2002). CASA: A distributed Holonic multi-agent architecture for Timber production. Proceedings of AAMAS, 2002, Bologna, Italy, pp. 551-552.
[13] Hagras, H., Callaghan, V., Colley, M. and Clarke, G. (2003). A hierarchical fuzzy-genetic multi-agent architecture for intelligent buildings online learning, adaptation and control. Information Sciences, Vol. 50, 2003, pp. 33-57.
[14] Haigh, K. Z., Phelps, J. and Geib, C. W. (2002). An open agent architecture for assisting elder independence. Proceedings of AAMAS, 2002, Bologna, Italy, pp. 578-586.
[15] Han, S. Y., Kim, Y. S., Lee, T. Y. and Yoon, T. S. (2000). A framework of concurrent process engineering with agent-based collaborative design strategies and its application on plant layout problem. Computers and Chemical Engineering, Vol. 24, 2000, pp. 1673-1679.
[16] Hernández, J. Z., Ossowki, S. and García-Serrano, A. (2002). Multi-agent architectures for intelligent traffic management systems. Transportation Research Part C, Vol. 10, 2002, pp. 473-506.
[17] Ivezic, N., Potok, T. E. and Pouchard, L. (1999). Multi-agent framework for lean manufacturing. IEEE Internet Computing, 1999, pp. 58-59.
[18] Jonker, C. M., Lam, K. A. and Treur, J. (2001). A reusable multi-agent architecture for active intelligent websites. Applied Intelligence 15, 2001, pp. 7-24.
[19] Kim, T. W., Ko, C. S. and Kim, B. N. (2002). An agent-based framework for global purchasing and manufacturing in a shoe industry. Computers and Industrial Engineering, Vol. 42, 2002, pp. 495-506.
[20] Kolp, M., Giorgini, P. and Mylopoulos, J. (2002). Organisational multi-agent architectures: A mobile robot example. Proceedings of AAMAS, 2002, Bologna, Italy, pp. 94-95.
[21] Lee, K. J., Chang, Y. S. and Lee, J. K. (2000). Time-bound negotiation framework for electronic commerce agents. Decision Support Systems, Vol. 28, 2000, pp. 319-331.
[22] Liang, T. P. and Huang, J. S. (2000). A framework for applying intelligent agents to support electronic trading. Decision Support System, Vol. 28, 2000, pp. 305-317.
[23] Lin, F. R. and Chang, K. Y. (2001). A multiagent framework for automated online barbaining. IEEE 2001, pp. 41-47.
[24] Liu, D. R., Lin, Y. J., Chen, C. M. and Huang, Y. W. (2001). Deployment of personalized e-catalogues: An agent-based framework integrated with XML metadata and user models. Journal of Network and Computer Applications, Vol. 24, 2001, pp. 201-228.
[25] Logi, F. and Ritchie, S. G. (2002). A multi-agent architecture for cooperative inter-jurisdictional traffic congestion management. Transportation Research Part C, Vol. 10, 2002, pp. 507-527.
[26] Maamar, Z. (2003). Design of a simulation environment based on software agents and the high level architecture. Information and Software Technology, Vol. 45, 2003, pp. 137-148.
[27] Marzi, R. and John, P. (2002). Supporting fault diagnosis through a multi-agent-architecture. Mathematics and Computers in Simulation, Vol. 60, 2002, pp. 217-224.
[28] Mason, C. R. and Moffat, J. (2001). An agent architecture for implementing command and control in military simulations. Proceedings of the 2001 Winter Simulation Conference, pp. 721-729.
[29] Moon, J. C. and Kang, S. J. (2000). A multi-agent architecture for intelligent home network service using Tuple Space Model. IEEE 2000, pp. 791-794.
[30] Moran, D. B. and Park, S. (1997). Multimodal user interfaces in the open agent architecture. Proceedings of IUI, 1997, Orlando, Florida, USA, pp. 61-68.
[31] Mostagir, M. and Decker, K. (2002). A multi-agent system architecture for active Networks. Proceedings of AAMAS 2002, Bologna, Italy, pp. 1417-1418.
[32] Odrey, N. G. and Mejía, G. (2003). A re-configurable multi-agent system architecture for error recovery in production systems. Robotics and Computer Integrated Manufacturing, Vol. 19, 2003, pp. 35-43.
[33] Paderewski-Rodríguez, P., Rodrígez-Fortiz, M. J. and Parets-Llorca, J. (2003). An architecture for dynamic and evolving cooperative software agents. Computer Standards and Interfaces, Vol. 25, Issue 3, June 2003, pp. 261-269
[34] Rovatsos, M., Weiss, G. and Wolf, M. (2002). An approach to the analysis and design of multi-agent systems based on interaction frames. Proceedings of AAMAS 2002, Bologna, Italy, pp. 682-689.
[35] Russ, S. H., Reece, K., Robinson, J., Meyers, B., Rajan, R., Rajagopalan, L. and Tan, C. H. (1999). Hector: an agent-based architecture for dynamic resource management. IEEE Concurrency, 1999, pp. 47-55.
[36] Shaw, N. G., Mian, A. and Yahav, S. B. (2002). A comprehensive agent-based architecture for intelligent information retrieval in a distributed heterogeneous environment. Decision and Support Systems, Vol. 32, 2002, pp. 401-415.
[37] Sheremetory, L. B. and Smirnov, A. V. (1999). Component integration framework for manufacturing systems re-engineering: agent and object approach. Robotics and Autonomous Systems, Vol. 27, 1999, pp. 77-89.
[38] Simon, R., Nolan, J. and Sood, A.. (2002). A light-weight agent architecture for collaborative multimedia systems. Information Sciences, Vol. 140, 2002, pp. 53-84.
[39] Smolko, D. (2001). Design evaluation of the mobile agent architecture for distributed consistency management. IEEE, 2001, pp. 799-800.
[40] Sturm, M., Eder, K., Brauer, W. and Gonzàlez, J. C. (1997). Hybridization of neural and fuzzy systems by a multi agent architecture for motor gearbox control. Fuzzy Sets and Systems, Vol. 89, Issue 3, 1997, pp. 309-319.

[41] Tewari, G., Youll, J. and Maes, P. (2002). Personalized location-based brokering using an agent-based intermediary architecture. Decision Support Systems, Vol. 34, 2002, pp. 127-137.

[42] Thangiah, S. R., Shmygelska, O. and Mennell, W. (2001). An agent architecture for vehicle routing problems. Proceedings of SAC 2001, Las Vegas, NV, pp. 517-521.

[43] Tsai, C. F. and Wu, H. C. (2002). MASSIHN: A multi-agent architecture for intelligent home network service. IEEE 2002, pp.505-514.

[44] Tso, S. K., Lau, H. C. W., Ho, J. K. L. and Zhang, W. J. (1999). A framework for developing an agent-based collaborative service-support system in a manufacturing information network. Engineering Applications of Artificial Intelligence, Vol. 12, 1999, pp. 43-57.

[45] Zha, X. F.. A knowledge intensive multi-agent framework for cooperative/collaborative design modeling and decision support of assemblies. Knowledge-based Systems, Vol. 15, 2002, pp. 493-506.

[46] Zhuge, H., Chen, J., Feng, Y. and Shi, X. Q.. A federation-agent-workflow simulation framework for virtual organization development. Information and Management, Vol. 39, 2002, pp. 325-336.

[47] H Tianfield. An innovative tutorial on large complex systems. Artificial Intelligence Review, An Intl Science and Engineering J, Vol. 17, No. 2, Apr 2002, pp. 141-165.

[48] H Tianfield. Formalized analysis of structural characteristics of large complex systems. IEEE Trans on Systems, Man, and Cybernetics, Part A: Systems and Humans, Vol. 31, No. 6, Nov 2001, pp. 559-572.

[49] H Tianfield. Structuring of large-scale complex hybrid systems: From illustrative analysis toward modelization. J of Intelligent and Robotic Systems: Theory and Applications, Vol. 30, No. 02, Feb 2001, pp. 179-208.

[50] H Tianfield (2003). Multi-agent based autonomic architecture for network management. Proceedings of the IEEE International Conference on Industrial Informatics (INDIN'03), Banff, Alberta, Canada, August 21-24, 2003, 8 pages.

[51] H Tianfield (2003). Multi-agent autonomic architecture and its application in e-medicine. Proceedings of the 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT'2003), Halifax, Canada, October 13-17, 2003, 4 pages.

[52] H. Tianfield (2001c). Agentization and coordination. International Journal of Software Engineering and Knowledge Engineering, Vol. 11, No. 5, pp. 1-5.

[53] H. Tianfield (2001d). Towards advanced applications of agent technology. International Journal of Knowledge-based Intelligent Engineering Systems, Vol. 5, No. 4 (a), pp. 258-259.

[54] Papavassiliou, S., Puliafito, A., Tomarchio, O. and Ye, J. (2002). Mobile agent-based approach for efficient Network management and resource allocation: Framework and applications. IEEE Journal on Selected Areas in Communications, Vol. 20, No. 4, June 2002, pp. 858-872.