We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

13

Intelligent Collaboration Environment in Multi-Agent System Enabling Software Dynamic Integration and Adaptive Evolving

Qingshan Li, Lili Guo, Xiangqian Zhai and Baoye Xue
Xidian University
China

1. Introduction

Along with the quick development of Internet, network techniques and software techniques, computing development has the remarkable characteristics of distribution, maneuverability and heterogeneous nature. How to expand a system quickly and how to integrate a legacy system into a new one have been more and more concerned. System integration is combined with independent units into a system with harmonious function and intimate connection, rather than simple accumulation of equipments. For lack of integration standards, some problems are brought to the systems inter-connectivity and interoperability. So the information of one system could not be shared by others, the services provided also could not be used and people have to develop a new one. Component-based technique is one of the main means for integrated problem, but it can not meet the need of autonomy, intelligence, dynamic property and domain-oriented extensibility.

The rapidly changing environment of internet demands that system integration development should dynamically adapt and actively react to the changing requirements and contexts. However, in the traditional component-based integration technologies, the integrated components are statically bound, and the collaboration mode among them is fixed so that it can not be adjusted and modified especially when system integration is performing. Therefore, these technologies are unable to adapt to the frequently changing requirements and environments.

Though the traditional component-based integration approach has brought some solutions towards system integration of information systems, the integrated components is statically bound and the collaboration mode of them is single and fixed, it can not be adjusted and modified especially when the system is running. As above described, in the Internet computing environment, the requirements are variable and the computing contexts are dynamic and changeable , which demands that the collaboration mode of the integrated systems can be dynamically and flexibly modified to rapidly respond to those changes. Therefore, the traditional component-based integration approach can not meet this demand to handle those changes.

With the emergence of agent technology, a new approach based on agent was proposed in system integration flied which can resolve the problem above well which the traditional component-based integration approach can not handle. Through wrapping legacy systems

and newly-developed systems into agents which has the characteristics of intelligence, autonomy, and adaptability, the system integration becomes the integration of agents which can flexibly handle the changes. In this paper, based on agent technology, the agent dynamic discovery and collaboration mechanism is attained. Additionally, through the application of Contract Net Protocol into the cooperation among agents, the dynamic bidding of agent is implemented and the adaptive integration which can adapt to the variable environment is gained. Finally, with script-described integration requirement strategy, the flexible requirement transition which does not interrupt the running system is achieved by script switching, which accomplishes dynamic evolution of system integration. Based on above, the dynamic and flexible integration approach is achieved which perfectly handles the environment and requirement changes.

Agent technology provides a new way to resolve the above problems.Agents are intelligent, flexible and autonomous so that they are capable of adapting intelligently to the environment changes. In this chapter, the agents are applied to system integration to achieve dynamic system integration and adaptive evolving, through packing the sub-systems into the agents and the communication and collaboration among the agents, the software dynamic integration and adaptive evolving is implemented. More specifically, the agents in intelligent collaboration environment are based on Agent Model proposed in first section which makes the agents active and intelligent, so that the overall system is able to dynamically and immediately perceive the presence and disappearance of agents. The agents' collaboration mechanism is based on Contract Net Protocol and the related algorithms which enables the agents to cooperate more flexibly and intelligently through bidding and tendering. Furthermore, the integration logic which depicts the integration requirement is expressed as the script , with the interpretation of the script, the agents collaborate with each other in terms of the script and thus the system integration is performed dynamicall, especially, when the integration requirement is changed, the new script is designed after that, with the the interpretation of the new script, the new system integration is accomplished and the new requirement is satisfied, the script-based integration control mechanism makes the flexible integration transition which copes with the frequently changing requirements perfectly.

Agent technologies and multi-agent systems have also been applied to some actual systems, i.e. Agile Manufacturing system, Air traffic management system (Rabelo et al., 1998) and e-commerce systems. After the original CNP was proposed in (Smith & Davis, 1980), many researchers had made much effort to refine and extend it with more flexibility, adaptability and better performance. Some most typical works as shown here: T. Sandholm proposed a formal model (Sandholm, 1993), whose pricing mechanism generalizes CNP to work for both cooperative and competitive agents. By focusing on the issue of self-interest agents in automated negotiation system, Sandholm extended CNP to Leveled Commitment Contract (Sandholm & Lesser, 1996), which allows agent to end an ongoing contract if an outside offer is more profitable than current contract. Besides, there were some endeavors on formalization of CNP, e.g. M. d' Inverno and M. Luck (Inverno & Luck, 1996), Werner (Werner ,1989), M. Fisher & M. Wooldridge (Fisher & Wooldridge, 1994). By adding rejection and confirmation communicative action, FIPA made the original CNP an important standard for a whole range of prominent agent platform implementations. The newest extension for this standard is FIPA iterated CNP (ICNP). It allows multi-round iterative bidding . In order to enhancing CNP, T. Knabe (Schillo, Fischer, Kray,2000) proposed CNCP which is another helpful extension of CNP.

In the actual application, agents are being placed in dynamic network environment, and their kn- owledge, capability and overload will be undergoing dramatic changes with the task execution. Howev- er, a single agent cannot get a global knowledge of the task complexity and the states of other agents. Additionally, the manager should coordinate the relationship among agents according to the agents' ability. When selecting an agent as the successful bidder, the manager should consider some agent's successful experience of the task execution and give chances to those agents who lack experience from the task execution. Based on the above considerations, we introduce the perception coefficient and the degree of credibility into the traditional contract net protocol and propose the Contract Net Model based on Agent Initiative Perception (CNMAIP). CNMAIP is applied to the Agent-based System Integration Tool (ASIT), and has been tested with the Border and Coast Defense Simulation System. The result demonstrates that, CNMAIP not only increasingly self- adapts dynamically to the change of the agent's ability and the environments, but also improves the negotiation efficiency on the basis of sufficient ensuring negotiation quality.
Based on the above model, strategies and mechanism, the dynamic integration and adaptive evolving architecture is designed and the intelligent collaboration environment in multi-agent system is accomplished which involves Agent Wrapper, Script Design Tool and Run-time Support Platform and so on. The environment provides support not only for the integration development phase but also for the run-time management phase of system integration. Finnaly, the intelligent collaboration environment is demonstrated well by the integration simulation experiments of Border and Coast Defense System and the experiment result indicates that the dynamic software integration and adaptive evolving is achieved.
This chaper is divided into three sections: Agent Model, Dynamic Integration and Adaptive Evolving Architecture,Agent Collabration Mechanism and Algorithms. Agent Model section introduces the definition of Agent(includes Function Agent and Service Agent) and inner mechanism, the integration units is wrapped into the agents which is active and intelligent, Agent Model is the foundation of Agent Wrapper. Dynamic Integration and Adaptive Evolving Architecture section depicts the whole architecture and integration development environment , which consists of several tools and platforms. The last section decribes the Agent Collaboration Mechanism and Algorithms which enables the intelligent collaboration among the agents.

2. Agent model

Traditional agent models are divided into three types: reactive agent model, rational agent model and a hybrid of the above. Brooks first developed the idea of reactive agent model (Brooks, 1991) and proved its practical utility with robots experiments. The task strategy of reactive agent model is hierarchy. The low level tasks are atom behaviours which are similar to the human stress response. The high level tasks are composed task and can be divided into low level tasks. The reactive agent model is quite efficient in task execution. Although the reactive model embodies the process of reaction to its surrounding, it is often passive and is low in adaptability and portability. The rational agent model, originated from artificial intelligence, focuses on formal logic to describe agent (Cohen & Levesque, 1990). The most influenced is Anand S. Rao and Michiael P. Georgeff's BDI model (Rao & Georgeff, 1991). BDI model has a sound theoretical foundation, and many researchers and scholars, such as Shi Chunyi (ZHANG & SHI, 2002) and Shi Zhongzhi (DONG et al.,2004) in China, have made expansion and application on it. Core idea of BDI is its internal goal

oriented reasoning and planning process which is expressed in Belief, Desire and Intention. BDI model performs well in agent intelligence, but it is difficult to implement and has a certain distance for practical application.

In recent years, Michael Wooldridge put forward the VSK agent model (Wooldridge & Lomuscio, 2000). It describes agent in the concepts of Visible, Perceive and Know. The VSK model also belongs to the rational agent model. However, it does not specify its internal structure, while it places emphasis on agent semantics under the MAS environment.

In our view, an agent is a collection of primitive components that provide a focused and cohesive set of capabilities. To obtain the domain-oriented extendibility and smart features, we design two categories of agent, Service agent and Function agent, which are short as SA and FA correspondingly. FA is a concrete entity (functional component) which possesses some basic capabilities while SA is an abstract entity which organizes the FAs together in terms of certain collaboration logic to provide a high level service. The semantic model is showed as Agent=<*Aid, Type, Mod, Des, Caps, Acq, Mq, Mh, Co, Rules*>.Of which:

Aid=<*ip, port, name*> is defined as the identifier, including the host machine IP and communications port, as well as the agent's name;

Type- represents the type of agent, namely, Service and Function, which are equivalent to SA and FA correspondingly;

Mod- represents the functional components whose functions are its capacities;

Des- describes the basic information of the agent;

Caps- describe the set of agent's capacities;

Acq- describes the agent's acquaintances. Each of the acquaintances includes its aid, and capability information. This attribute aims at SA, the acquaintances are its organized FAs. FA does not own this attribute.

Mq- represents message queue, which caches the received messages;

Mh- represents message processor, which analyses and encapsulates message content;

Co- represents agent collaboration engine, which mainly interprets and implements the service process of Service Agent;

Pe- represents agent sensor, which senses the related ability and new published tenders on CMB;

Rules=<$rule_1$, $rule_2$,…, $rule_n$> expresses the rules database of agent, which stores the integration rules of system integration. The behaviour of Agent is determined by the cooperative relationship with other Agents described in the integration rules.

In the model above, *Caps* and *Acq* behalf the behavior of agent.The *Caps* contains two types of capability: atomic capability and composite capability, corresponding to the capabilities provided by FA and SA. An atomic capacity can accomplish a specific task without the collaboration with the others, while a composite capacity is the combination of atomic capabilities that are not all provided by the same agent. The *Acq* describes the agent's relationships with other agents in the society, and its behaviours about the capabilities and addresses of those agents.

There are two message queues: incoming-message-queue and outgoing-message-queue corresponding to two threads respectively. *Mh* processes incoming messages from *Co* or *Pe,* putting them into the outgoing-message-queue. For each message awaiting dispatch, *Mh* queries the message object for the intended recipient. For each message in the incoming-message-queue, *Mh* extracts its content. If the content is simple, e.g. the response to the registration of an agent from CRC (Capability Register Center) or AMS(Agent Management Service), it is handled by the *Mh* itself. Other messages, e.g. tasks to be executed, are put into the task queue or the buffer in *Pe*.

The Rules and the *Co* are cores of an agent. The role of the former is to construct action sequences that achieve desired input goals. The role of the latter is to manage its problem solving behaviours, particularly those involving multi-agent collaboration.
The *Pe* can perceive the changes in the external environment. It can realize what new capabilities have increased in CRC and perceive tenders issued by CMB (Common Message Blackboard). With the help of the *Pe* in the control platform, it enhances the pro-active of agent to obtain more acknowledge about agent in the system, so as to make a better decision.
The message handler processes incoming messages from Co-ordination Engine or Sensor, putting them into the outgoing-message-queue. For each message awaiting dispatch, the communicator queries the message object for the intended recipient, and looks up a local address book for the recipient's address. If the address is found, the message is put into the incoming-message-queue. On the contrary, the communicator stores the message object onto a holding buffer, and queries known agents for the required address. Once the message recipient's address is received, the communicator removes the relevant message from the holding buffer and proceeds to dispatch the message. In the event that no address is found or network communications fails, a suitable error message is generated, which the communicator adds to the reader's incoming-message-queue to be processed as a normal incoming message.
For each message in the incoming-message-queue, the communicator extracts its content. If the content is simple, e.g. the response to the registration of an agent from CRC or AMS, it is handled by the communicator itself. Other messages, e.g. tasks to be executed, are put into the task queue or the buffer in Sensor.
Actually the message handler is an agent's internal message sorting office, continually checking the incoming-message-queue for new messages, and forwarding them to the relevant components of the agent..
The role of an agent's Co-ordination Engine is to manage its problem solving behaviours, particularly those involving multi-agent collaboration. The role of the Planner is to construct action sequences that achieve desired input goals. It controls a single-agent planning mechanism. Currently, a planner based on a plan library which provides logical descriptions of planning tasks known to the agent is provided where plans are hierarchical skeletal structures the nodes of which may be either new subplans, or executable subtasks.
The Co-ordination Engine and the Planner are cores of an agent. They contain the desire and intention of an agent. When processing a task, first the Co-ordination Engine invokes the Planner to decompose the task into subtasks, and then the Co-ordination Engine manages the execution of the subtasks. If one subtask cannot be finished by an agent itself, the agent asks a favour of its acquaintances. Once one acquaintance is found to aid in completing the subtask, the agent will send a message to it to request executing the subtask; otherwise a tendering is initiated by the agent. At the same time, the subtask is put into the internal buffer to prevent the subtask thread from running.
The Sensor can perceive the changes in the external environment. It can realize what new abilities has increased in CRC, if some of the ability is related to its capability, it will put the owners of the ability as its acquaintances into AD. In addition, it also can perceive tenders issued by PIB, get the tender and measure its capability, then decide to bid or not. The application of Sensor will increase the agent's initiative, and therefore the agent can get more knowledge of other agents in the system, so that the agent can make a better decision. With the help of Sensor and PIB in the control platform, the Contract Net Model based on Agent Initiative Perception introduced in next section is realized.

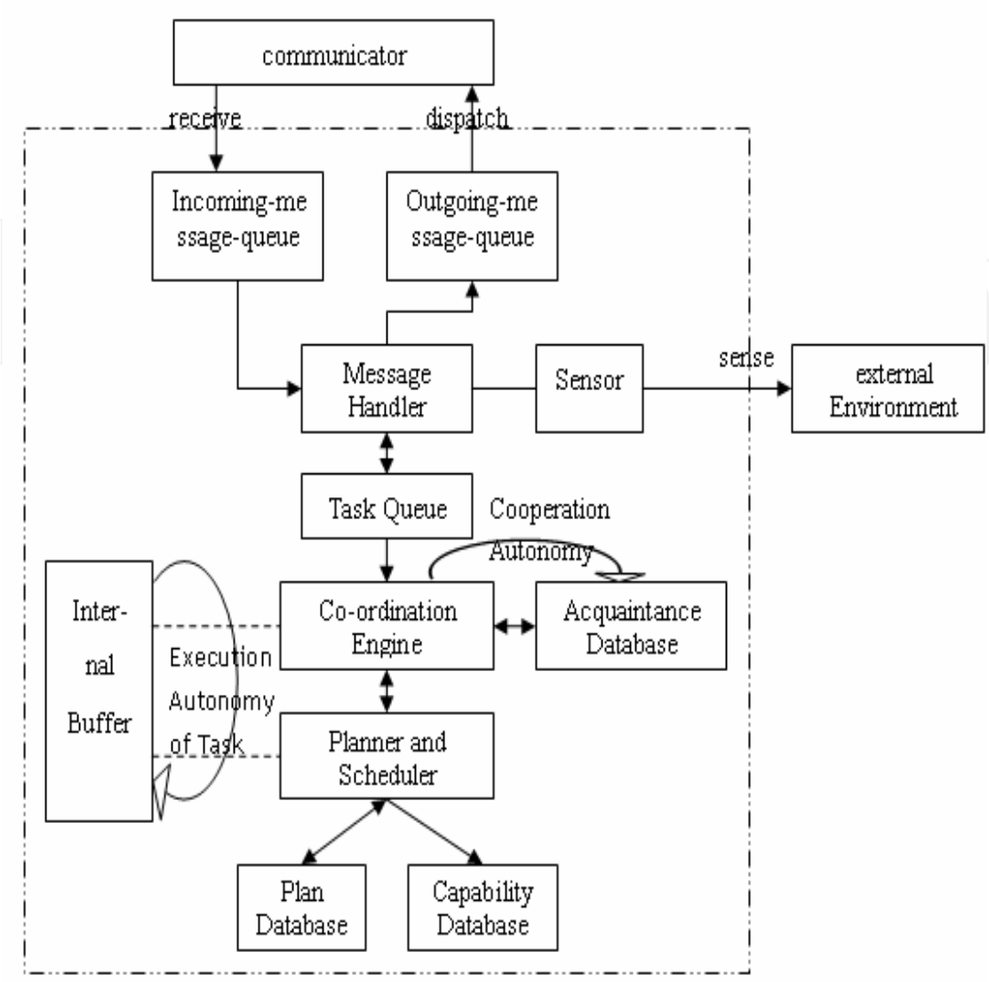


Fig. 1. Agent Model

Agent unit have the characteristic of intelligence and initiative, that's can work independently on its own knowledge, it also has the ability to actively find company. When the agent receives a task, it will call its own planner/scheduler to decompose capacity into several sub-capacities. When it find some sub-capacity do not exist in its own capacity database, the agent will give the sub-capacity to its co-ordination engine. Firstly the co-ordination engine find the acquaintances in the acquaintance database whether they possess this sub-capacity, if don't ,it will request collaboration to other agents through the contract net protocol which was widen by a conversational method, and ultimately complete missions with other agents by horizontal collaboration. The collaborative manner reduces the difficulty of realization of the requirement of integration and improves the intelligence of the system integration.

3. Dynamic integration and adaptive evolving architecture

3.1 Agent-based integration framework

Referring to FIPA platform specifications, this paper designs an agent-based integration framework in Fig. 1, which supports the dynamic integration in the distributed systems.

When the model is applied, only one computer acts as the control platform and the other computers as non-control platforms. Their coordination accomplishes the system integration. The CRC (Capability Register Center), AMS (Agent Management Service), CMB (Common Message Blackboard), and CA (Control Agent) just exist on the control platform, as special agents to provide services to the other Agents; there is an MTS (Message transfer Service) and an Agent Database on the control platform and each non-control platform.

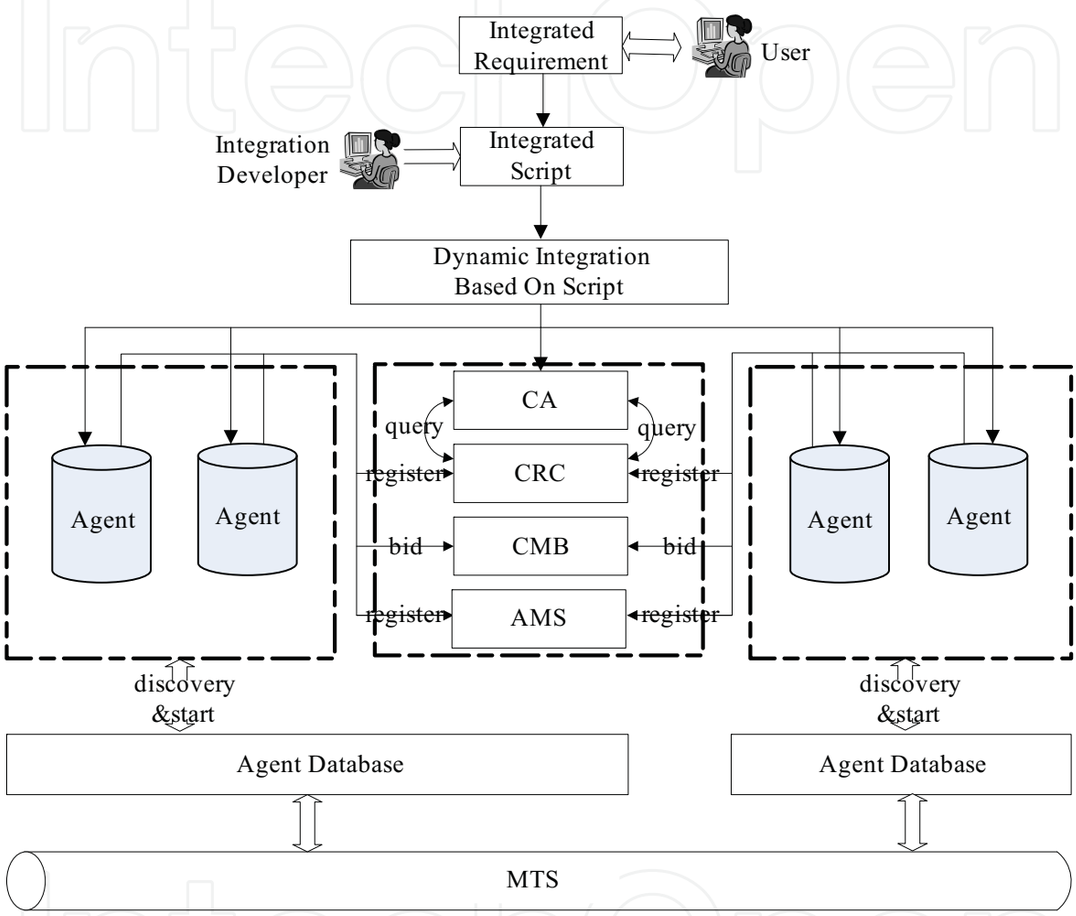


Fig. 2. Dynamic Integration and Adaptive Evolving Architecture

Among those agents, CRC is one of the core modules in the framework and is responsible for the registration of all the agents' capabilities in the platform and the query of these capabilities information; AMS performs real time monitoring on each state and address of the agents (idle, running, waiting and resume), queries the information according to the condition; CMB as the intermediate information service module, provides initial and response service for biding negotiation; CA is not only an intermediary serving but also controls the script interpretation by which integration rules is obtained to achieve System Dynamic Integration; as the relationship between integration units, MTS provides synchronous or asynchronous communication for the interactive and communication among platforms and modules; Agent Database is used to perceive the Agents on the platform, and once one Agent is perceived, it will be added into the local address book that exists in the Agent Database and starts running. Then the agent registers capabilities to CRC

and addresses and states to AMS. Also, it can load and upgrade the definition files and capability components in run model to manage static maintenance for Agents.
In the function of this framework, in terms of the wrapper standard, integration developers obtain the definition files and capability components. They are loaded by Agent Database in run mode. Once one Agent is perceived, it will be added in the MAS. After users raise integration needs, integration developers call integration logic by logically abstracting those needs, and then it can be mapped to script. Therefore, CA interprets the script. When running into a simple capability of the script, it will send a message to CRC looking for agents with the capability. After receiving the searched result from CRC, it will obtain the integration rules and send messages to them to request executing the task. Finally, Agents will collaborate with other agents to realize system integration according to the contract net protocol.
It effectively uses the Pro-active, Autonomy and Collaborative of agent, combines the distribution control and concentration control, and achieves the aim of dynamic system integration and expansion in system integration. The problem of load balance is solved by playing a role of the Perceptive, Autonomy and Collaborative of Agent.
In the traditional component-based integration development, the relationship between traditional distributed components is static, which is difficult to adjust the interactive relationship among components dynamically on the fly. The agent-based system integration approach is applied to resolve the dynamic automation. In the approach, the dynamic discovery of agent is the critical part and premise to accomplish system integration. On finding agents that meet needs, it guarantees that agents can be combined and collaborated to accomplish tasks, which realizes system integration. The process illustrates that the dependence, between requirement and agents, is dynamic, but not static.

3.2 Dynamic discovery based on semantic ontology

The dynamic discovery of agent is the automatic location of agent that provides the special capability and accords with limited conditions. For example, assuming that it needs a member of interface display to show the situation, the common method is to redevelop it by requirement or to inquire the existed model base by keywords. But the method can not be used to inquire component and is difficult to meet the rapid construction of application system. Its reason is that facing the mass of components, they are common on the interface specification, but they greatly differ in the semantic meaning of different domains. So it is difficult to locate agent and guarantee the reuse and combination of agent correct and effective.
The core and critical of dynamic discovery of agent is to construct the effective matching algorithm. Mili, et al consider that it is necessary to do the accuracy of initial inquiry by domain knowledge and make more effective inquiry by reasoning based on case and other artificial intelligence technologies.
As the critical port of application system, besides the basic information of general components of function interface and parameter information, agent includes the rich information of model semantics and linguistic context that serve as heuristic information in the agent-matching algorithm. Because of the inefficiency of the pure keyword matching, this paper proposes an accurate matching based on semantic ontology to enhance the accuracy and efficiency of agent-matching algorithm, which is divided into two matching stages, domain location and ability orientation. At the stage of domain location, related domain ontology of an agent can be determined by the algorithm; and then at the stage of

ability orientation, it realizes the agent location by the interface information of agent in the domain ontology.

$$DoC_a = \frac{\sum_i^n t_i \bullet v_i}{\sum_i^n v_i} \tag{1}$$

In the algorithm of domain location, the basic information of agent may be same as that of others and they are relevant with the concept of different domain-ontology, as a result it is necessary to definite the most relevant domain-ontology. We define the set of domain-ontology as AO={ao1,ao2,…,aon}, of which each domain-ontology aoi={a1,a2,…,an}, and ai is the set of agent involved in aoi. If there exists mapping relationship between the domain of agent in integration requirement and the concept of the domain of aoi ∈ AO, aoi is the domain-ontology of the agent. By the domain location it can filter a mass of agent irrelevant with integration requirement.

In the algorithm of ability orientation, all the capabilities consist the capability set of Container={ Cf1, Cf2,…, Cfn}, where for each Cfi (1≤i≤|Container|), if there exists inclusion relation between the domain-ontology ao and integration requirement, it holds that all the capability of the agent are put into the set of candidate capability – Candidates. According to the semantic of agent, it makes the accurate matching between agent and integration requirement. In other words, it adjusts whether the capability of agent in Candidates, meets the demand of agent involved in integration requirement by semantic matching.

3.3 Dynamic integration and adaptive evolving

General Magic's Telescript system is developed based on the integration technology of script-interpretation, providing system integration with a new idea. However, the interaction among agent is limited to the local method invocations in Telescript, which adopts the static binding and unable to meet the changes of internet and integrated requirement. Therefore, making full use of the pro-active and collaborative of agent, improving the Contract Net Protocol (CNP) proposed by Reid G. Smith and Randall Davis, it implements a combination of bidding and script switching, to achieve the dynamic.

As is shown in the Fig. 3, when an SA wants to find a contractor through negotiation, it announces the task to be allocated by sending a Bid to CMB. Therefore, the SA is considered as an Initiator. Receiving an announcing from the Initiator, CMB gets the potential contractor (agent) according the announcing. Then CMB inquires CRC to locate agent dynamically. For example, the task in the announcing is type ti, according to the classification of task. If Agentj has an ability of type ti, Agentj is a potential contractor of the Initiator and will be a participant of the negotiation in the bidding. CMB will send calls for proposals to the participants. The main difference from CNP is that CMB sends calls for proposals to participant according the agent ability information it maintains, not to each agent in a broadcast way in CNP.

Meanwhile, CMB monitors the new register and update information. Once an agent is able to be a participant, no matter whether it is a new agent just deployed on to the platform or ability updated, CMB will send the call for propose to the agent if the announcing is still before the declared deadline.

$$DoA_a = \frac{1}{NCFP} \tag{2}$$

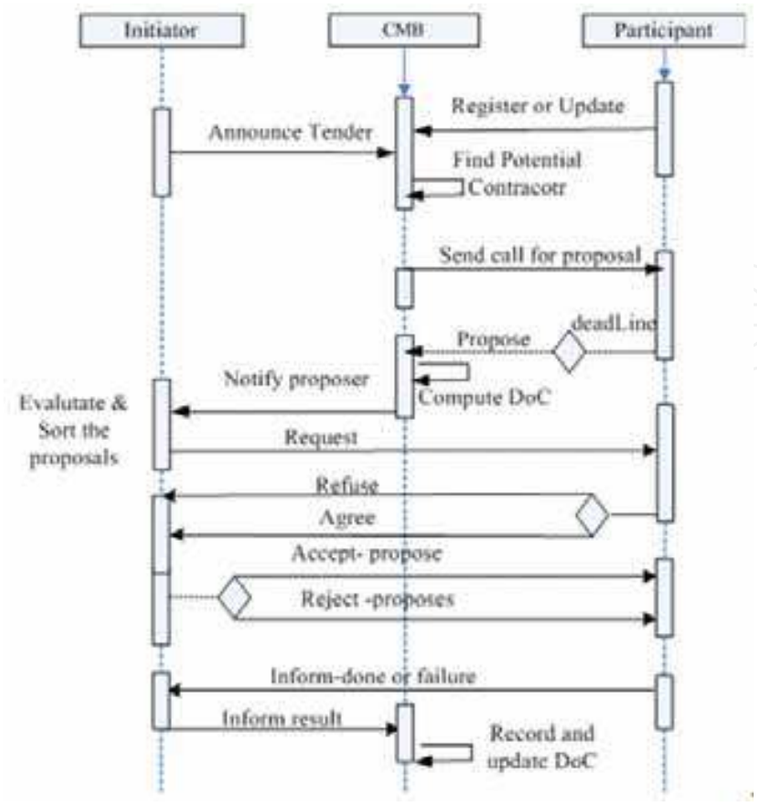


Fig. 3. Dynamic Bidding Process

The participant reads the call for propose sent from CMB and makes the decision whether to bid. The participant may send proposal to bid or refuse to propose by its own willingness, mainly including its ability, status and self-interest. The participant's propose includes the preconditions that the participant is setting out for the task, which are the cost and the number of call for proposals it has received etc. The information of call for proposals is helpful to evaluate the degree of availability of the participant and it is an important factor in awarding decision.

When CMB receives a propose from a participant, it computes the credibility of the participant (Degree of Credibility , Briefly DoC) to complete the task according negotiation history record in CMB. CMB will inform the Initiator with the information of the participant's proposal and its DoC.

Once the deadline passes, the Initiator evaluates and sorts received proposals. The Initiator will sort the received proposals by its cost, DoC and DoA. The higher of DoC and lower of DoA, the earlier the participant may receive the offer. The initiator will send a request to the participant in descending order of the computed value. The process of awarding can be described below.

If a participant receives the offer and agrees, the Initiator will send an accept-proposal message to the participant. The participant will be the contractor and begin to execute the specific task. When the task is done or failed, the contractor will inform the Initiator the result of task execution.

The Initiator will evaluate the result of task execution and inform CMB. CMB will record the negotiation and adjust the DoC of the participant.

The centralized control is the common way to realize the integrated control. It is easy to realize but it will result in the sharp decrease in efficiency of the integrated system. While the autonomy and intelligence of the agents make it interact with other agents initiatively by learning. We can make use of those characters of agents and adopt the control way which combines the distributed and centralized way, in which each agent controls its own behavior. This way not only avoids the sharp decrease in efficiency which is caused by centralized control, but also dynamically changes part or all of agent behaviors at run mode and eventually realizes dynamic system integration.

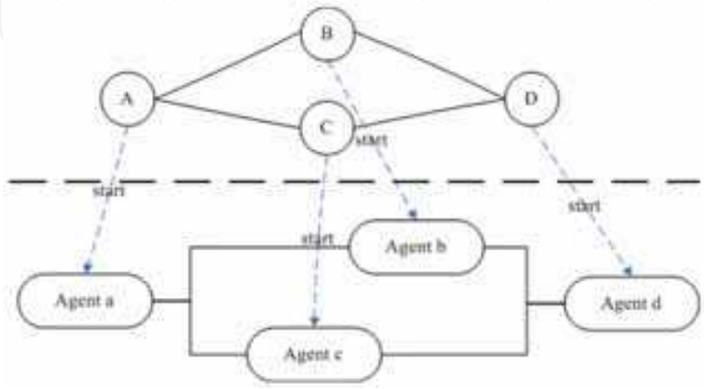


Fig. 4. Mapping Relation on Integrated Rules

Scripted-control integration can be illustrated easily as "once-interpretation and multi-execution". Once-interpretation refers to interpreting the integration scripts once, from which the description of interaction relationship among services is obtained and the integration rule is generated and distributed to the agents which participate in the application system integration. Multi-execution means that the participating agents consider the integration rule as a kind of knowledge and control the cooperation with other agents. As is described in Fig. 4, integration script describes the collaboration relationship among Task A, B, C, and D, aimed at accomplishing integration requirements. When the script is interpreted, the collaboration relationship between tasks is obtained: A→B, A→C, B→D, C→D. By inquiring CRC, the obtained Task A, B, C, D are accomplished respectively by the corresponding service which is provided by Agenta, Agentb, Agentc, and Agentd. The integration rule generated is distributed to the corresponding agent, and the collaboration relations are formed as is illustrated in Fig. 4. When Agents receive their corresponding rules, they will complete integration needs autonomously.

The script depicts the integration logic abstracted from integration requirements, so it has one to one correspondence with integration requirement. Therefore, Integration logic is bound to change when integration requirement needs, which makes integration developers to redefine the script. But in the traditional component-based integration development, it may stop the system via changing the integration logic, which decreases the overall system performance sharply. By switching the script dynamically, the system just replaces integration rules to achieve the flexible integration.

Script Switching, in essence, alters the integration logic smoothly in the process of system integration, under the charge of dispatching Integration Rules and the Agents' internal support. Integration Rules Dispatch is responsible for emptying and replacing the rules, and the Agents' internal support the Script Switching for storing and upgrading the rules.

There exist two situations during dispatching the rules. One is that Service Agenti is not in the pre-script; the other is that Service Agent Agentj is in the pre-script, but its integration logic is not the same. For the former, the rules need clearing; for the latter, the rules must be re-substituted. But both are to replace the integration rules in fact, and their difference lies in the implementation modes. In the Agent internal, storing integration rules provides the foundation for the system running, and updating rules ensures the consistency of integration logic.

When integration requirements change, first of all, the integrated developers abstract the requirements to obtain the integration logic, which is mapped to a script. In the fly, the integration logic is obtained by interpreting the script, and dispatched to the related agent. When receiving the logic, every Agent stores it in temporary zone. Once receiving the "Start" command, Agents substitute the rules of Rule Base with it in a split second. Combined with the flow defined in the Scomp of the related Agent, it forms entire task logic, which controls the system integration.

```
obtain the action and integration rules from message
If(action is "Clear_Rule")
{
assign the state of Agenti to idle and clear the rules base of Agenti.
}
if(action is "Dispatch_Rule")
{
search the capabilities by the agent's name;
if(the searching result is not empty){
clear the rules base of Agenti;
obtain the new rule, and write to the memory.}
}
send the result to CA via MTS
}
```

Algorithm 1. Script Switching Algorithm

4. Agent collaboration mechanism and algorithm

4.1 Partition and cooperating process of tasks

In the process of collaboration between agents, the first problem to be solved is task partition and allocation. The purpose of task allocation is to enable the system complete a task with as less cost as possible in its execution and communication, achieve both local and global objectives while ensure no conflicts occur between agents. At present, some typical works are heuristic allocation algorithm and queuing theory scheduling algorithm. Heuristic allocation algorithm merges the task partition and sub-task allocation as a single linear programming problem. This method is better in handling complicated tasks due to its use of focus coordination mechanisms and distributed collaborative mechanisms. However, when applied in closed-end MAS of small and medium-sized size, the cost of this algorithm is very high. If the completion of the task submitted by user just need one agent, the method of queuing theory is more suitable. The main idea of queuing theory is that each agent has a task queue and the agent performs the task in its task queue base-on the principle of FCFS (first come first served).The main algorithm for task distribution includes the smallest queue

algorithm , the shortest waiting time algorithm as well as historical information algorithm. The specific algorithm for a task allocation should be chosen according the needs of the task. In the field of system integration, the complexity of the task is uncertain because of the uncertainty of the task. If just taking the heuristic allocation algorithm, it would be of high risk. And the queuing theory scheduling algorithm could not be just taken for it could complete the complex tasks. To solve the problems above, FSMAS makes uses of the federal structure of itself, and presents solution: MAS allocates task of high complexity to service agents and service agents partition the complicated tasks to simple tasks and assigns the simple tasks to an appropriate function agent, where the algorithm for task partition and task allocation is used. In addition, there will be a process of collaboration between agents after task allocation.

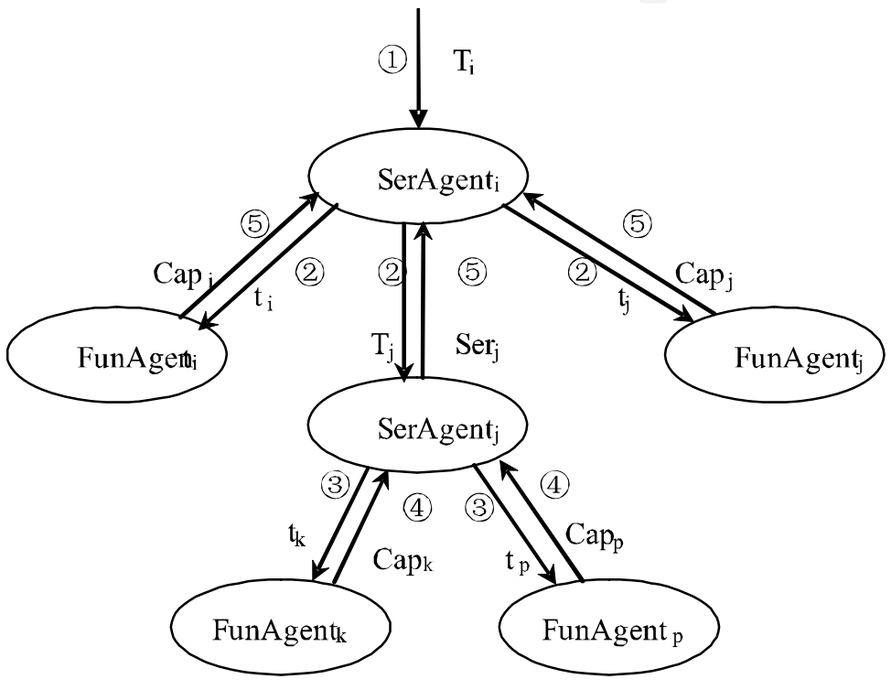


Fig. 5. Partition and Cooperating Process

At first, a whole task is divided by MAS into sub-tasks in a plan. Sub-tasks to be executed are allocated to some federal. The federal will make a partition according the information in the knowledge base of the service agent after getting the task. The sub-tasks will be distributed to and completed by Agent federations. After completing the sub-tasks, the federations collaborate with each other to finish the overall task. Tasks in FSMAS are divided into two major categories, the tasks which can be completed independently by a function Agent are atomic tasks, which are record as t. actually, a function Agent can only complete atomic tasks. Another kind of tasks completed by the collaborations among Agents, are collaborative tasks, record as T. It is defined as T= {Ti, Tk, Tk...ti,tj,tk,…}.

Fig.5. shows the flow chart as an instance of partition and collaboration of an overall task. The main steps are as follows.

① SerAgenti receive a certain task Ti;

② Ti is supposed to partitioned into ti, tj and Tj by SerAgenti, and distributed to FunAgenti, FunAgenti and SerAgentj;

③ Tj continues to be partitioned into tk and tp, which will be distributed to FunAgentk and FunAgentk;
④ FunAgentk and FunAgentk provide Capk and Capp to SerAgentj , SerAgentj organizes the capabilities to provide Serj;
⑤ SerAgenti organizes Capi , Capj and Serj to provide Seri. This organization has been able to complete the overall task Ti.

The specific algorithm for task partition is as follows.

```
Queue PartitionTask(Ti){
Queue taskqueue;
Ti →{task1, task2,….taskk}
taskqueue.pushback({task1, task2,….});
for(int i=1; i!=end; i++) {
If( ∃ taski |  taski ∈ T){
Queue subtaskqueue=PartitionTask(taski);
        taskqueue.pushback(subtaskqueue);
}
}
return taskqueue;
```

Algorithm 2. Task Partition

When a service Agent gets a collaborative task, it partitions the task according to the definition file and would get a task queue. If there still exists some collaborative tasks and no member service Agents could complete those tasks, the original service Agent would continues to partition those tasks until all of the tasks in the task queue could be completed by the members in its federation. At last, the original service Agent would return the task queue.

MAS take the simple algorithm for task allocation. The tasks which are allocated to service agents are determined by integration control-logic defined by user. One simple implementation is to assign the task to the first service agent found by query. For function agents, there are two ways for task assignment, one of which is service agents assign task to specific function agents by its knowledge and the other is task is assigned to specific function agents through biding with contract net protocol. In the filed of system integration, a function agent usually just serve as a function supply unit. There usually would not be many agents for an integration function unit. Therefore, the queuing theory allocation algorithm can not significantly improve system efficiency while the calculation of information would cause unnecessary system costs. The adoption of simple task allocation algorithm in FSMAS avoids the calculation of the length of task queue and waiting time and can meet the task allocation requirement in system integration filed.

4.2 Agent collaborative algorithm

The typical work of collaboration in MAS is Contract Net Protocol. The main procedure includes task announcement, sending call for proposal, bidding, evaluation and awarding and contract execution. Distributed Sensing System of MIT and expert union system UNIONS of Chinese Academy of Science Mathematics were both typical system based-on contract net protocol. However, contract net protocol also has its problems such as frequent communication, waste of system resources and limited scope of application. Blackboard

Collaboration is a more widely used collaboration method. The main idea is the Blackboard provides sharing solve space when a number of agents collaborate to solve a problem. All the agents involved can "see" the information of the blackboard and record the intermediate result to it until the problem is resolved. In this way, agents are independent of each other and the communication load is reduced. What agents need to do is to response the blackboard and to use the same communication language.

As the problems of CNP mentioned in the introduction, a new collaborative algorithm "acquaintance first base on CNP" is proposed in this paper. Its main idea is: service Agents choose the Agents in their acquaintance base to complete tasks first, if the Agents in acquaintance base can not complete the task, the service Agent would initiate a bid to get the Agents needed.

As Algorithm 3 shows, when a service Agent receives a task, it partitions the task by the partition algorithm has been mentioned. First, the Agent gets the Agents' information which would be organized according to the definition files. Second, the service Agent would search the Agents' information in CRC, the information would show which Agents are the acquaintances of the Service Agent and which are not. Third, if all the Agents involved are acquaintances, the service Agent would organize them and return a service; if not, then initiates a bid. When the service Agent gets all the Agents, it would organize them and return a service.

```
SerAgentExeTask(Ti){
taskqueue=PartitionTask(Ti);
taskqueue×CRC→Agentqueue;
    if((∀ agent | agent ∈ Agentqueue)→(agent ∈ Ab))
      Ser={Ser1,…Serm, …Cap1, …Capn}
|Seri ∈ Serbi, Capi ∈ Capbi;
    else {
            Bidagent | agent ∈ Agentqueue && agent ∉ Ab;
            Bidagent×BidMod→bid;
            Initiate(bid);
            Wait();
            Bidagents=Accept(bid);
Ser={Ser1,…Serm,…Cap1,…Capn,Bidagent1,…Bidagentk}
|Seri ∈ Serbi, Capi ∈ Capbi, Bidagenti ∈ Bidagents;
}
    return Ser;
```

Algorithm 3. Service Agent Execute A Task

Algorithm 4 shows how CMB process a bid when receives it and how the Agents wins the bid. When a service Agent initiates a bid, it would send a message about the bid to CMB, CMB would analyze the message and get some information about the bid after it received the message, then it would query CRC according to the message and get an AgentSet, next, it would send Msg which records the bid's information to all the Agents in AgentSet and wait for reply. CMB would send the information WinMsg of first replying Agent to the initiator Agent. And to every one in AgentSet, when they got the message, they would check their stat first and send WinMsg if the stat is idle. Then it would be waiting for invoking.

```
Bid(bid)
{
bid×CRC→AgentSet;
for(Agenti=First(AgentSet); Agenti!=Last(AgentSet); i++)
{
      SendMsg(Agenti, Msg);
}
Wait();
Send(WinMsg) | WinMsg is sent to initiator;
}
WinBid(bid)
{
      if(state==idle)
SendMsg(WinMsg);
```

Algorithm 4. Bidding Algorithm

Using "acquaintance first based on CNP" collaborative algorithm has the advantages as follows: 1. choosing acquaintance to collaborate could greatly enhance the efficiency of the collaboration; 2. the bidding mechanism could ensure the completion of tasks and the success rate of coordination; 3. adding acquaintance automatically after bidding, which prepares for the next collaboration.

4.3 Collaboration exception handling

Several exceptions always appear in the collaboration process. If those exceptions could not be handled appropriately, not only the robustness of system would be reduced significantly, but also the system would be attacked easily and even cause the system breakdown. Collaboration exceptions means any of a deviation from a normal execution process (such as resources, conflicts, overtime, etc), or that means a deviation from the best design. Collaboration exception could be divided into task partition exception, task allocation exception and collaboration executing exception. If the exceptions belong to the first two categories, the handling method is allocating the tasks again. This paper focused on the approaches which handle the exceptions appear in the task collaboration. These exceptions are divided into internal and external exceptions. Internal exceptions could be handled by the Agents who generate them. And external exceptions are those ones could not be handled by the owner without external assistance.

For internal exceptions, the handling mechanism inside of the Agent could handle them appropriately. And to handle external exceptions, some collaboration between Agents needed. The strategy is not only to recover the normal collaboration between the Agents, but also have some measures to be taken to make the abnormal Agents produce smaller impact in late collaboration. In FSMAS, service Agents manages communication between federations, function Agents could not communicate with others. So within a federation, service Agent can be used as a trigger to detect the collaboration exceptions, and also could handle the exceptions by the handling mechanism of themselves. And to between federations, the exceptions would be handled by the Agents who collaborate with the abnormal Agent.

The process of handling exceptions is as Algorithm 5 (suppose that agent1 produces an exception).

```
if(agent1 has e )
{       // e  is an exception
if (agent1 can handle e in_model)
 {
Handle(e);
 if(agent1 is OK)
 continue;
      Else
          Hand e to funAgent1;   //funAgent1 manages
// agent1
}
else
      Hand e to funAgenti;  // funAgenti collaborates
// with funAgent1
```

Algorithm 5. Exception Handling Algorithm

5. Agent-based intelligent collaboration environment

5.1 Agent-based system integration platform and tools

Based on above, the agent-based system integration architecture is designed, which supports the dynamic integration and adaptive evolution in the distributed system. Based on the architecture, Agent-based System Integration Development Environment (ASIDE) is implemented which involves Agent Wrapper, Agent Warehouse, Script Design Tool and Run-time Support Platform.

Agent Wrapper is responsible for packing the integration units (legacy systems or newly developed systems) into Function Agents and Service Agents which have the unified interfaces and possess some specific capabilities (functions). Script Designer is the tool for designing the script in DCISL and it has Text Mode and Graphic Mode. Moreover, the semantics, syntax and integration logic checking are also involved in Script Designer to ensure the correctness, reliability and integrity of the integration logic. With the data integration development tool, the data conversion regulation files are generated in terms of the data conversion relations (as shown in Fig.12 and Fig.13), which are the plug-ins. After being deployed in the platform, the data conversion regulation files are extracted the related information by the engine to perform the data conversion among the agents.

With the run-time management tools, the intelligent and dynamic integration process is obtained in the run-time support platform. AMS, Integration Control Tool and CMB are the three critical tools among the run-time management tools to implement this process. AMS is in charge of managing the overall agents which exists in MCP and NCP. When some new agents are loaded into platform, AMS will perceive them immediately and allocate resource to them, and then the new agents will join the running integration process dynamically and initiatively if needed. Additionally, the related status information about certain agent can be looked up and displayed in AMS real-timely. Integration Control Tool is used to load and interpret the script-based integration control logic to control the system integration action. The interpreted integration control logic is distributed to the related agents who store the

logic as their own regulation and act in terms of the regulation to accomplish the integration task. When the integration requirement changes, the integration control script is altered accordingly and then loaded to perform the new integration without interrupting the running system. Thus it achieves a smooth and flexible integration logic switch. Therefore, it is the strategy that can deal with the integration requirement transition flexibly and dynamically, while the traditional component-based system integration method can not.

Agent Warehouse takes charge of discovering and managing all the agents in the Run-time Support Platform. Script Design Tool is used to design the integration scripts in terms of integration requirements. And Run-time Support Platform provides service for the execution of system integration, which involves Platform Monitor Tool, Platform Configuration &&Management Tool, AMS, CRC, CMB and so on. Platform Monitor Tool timely monitors and displays the communication and collaboration amongst all the Agents which exist in the distributed system, and Platform Configuration&&Management Tool configures and manages the related platform information, such as IP and Ports of the computer. AMS is in charge of querying, managing and displaying the overall agents' information, CRC maintains a list of the agents' capabilities and can dynamically perceive the newly-loaded agents so as to record their capabilities; CMB provides service for the agent bidding mechanism based on Contract Net Protocol.

In the system integration process, the collaboration among agents adopts the federated acquaintance-first strategy. Once the capabilities of the agent can not accomplish the distributed integration task, it will try to find the appropriate agent through inviting public bidding based on Contract Net Protocol. The other agents who have the required capability will bid initiatively and provide service if chosen. It achieves the dynamic and intelligent collaboration among the agents. The biding process is monitored and the related agent information is displayed in CMB.

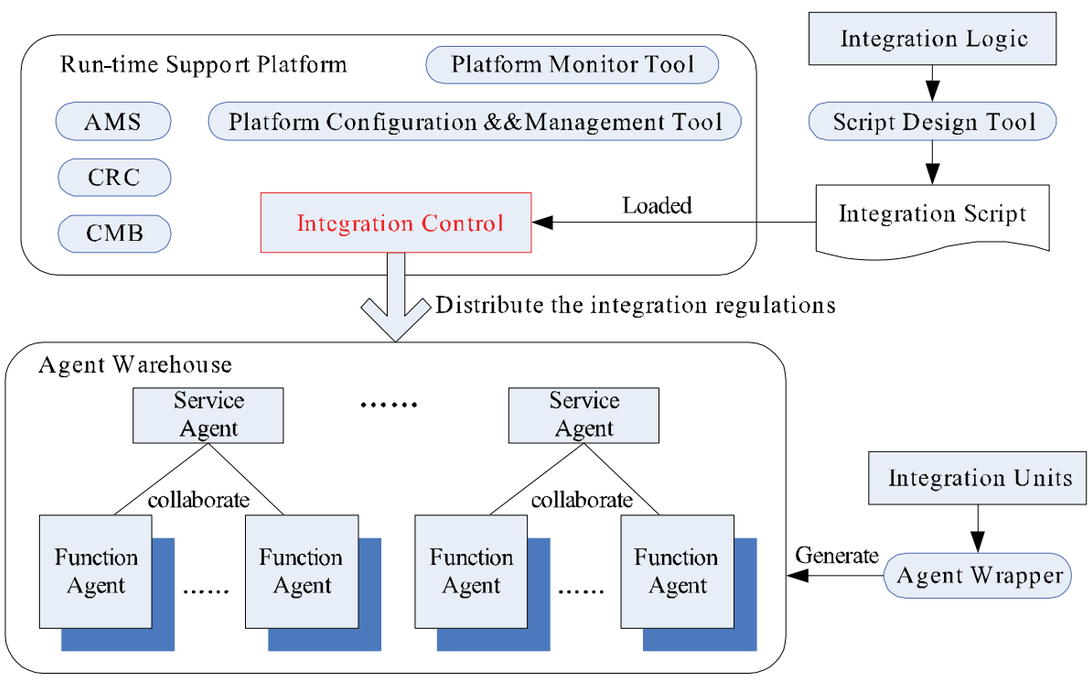


Fig. 6. Architecture of ASIDE

In ASIDE, the agents in Agent Warehouse can be discovered and perceived dynamically. With the help of CMB, the agents can gain or provide the required service in terms of bidding mechanism. Furthermore, when the integration requirement changes, the new integration logic is expressed as script, and without interrupting the running integration activities, the new script can be loaded into ASIDE to carry out the new integration. The dynamic switch of scripts leads to the integration transition, and it adapts to the frequently changing requirements flexibly and eventually achieves dynamic system integration.

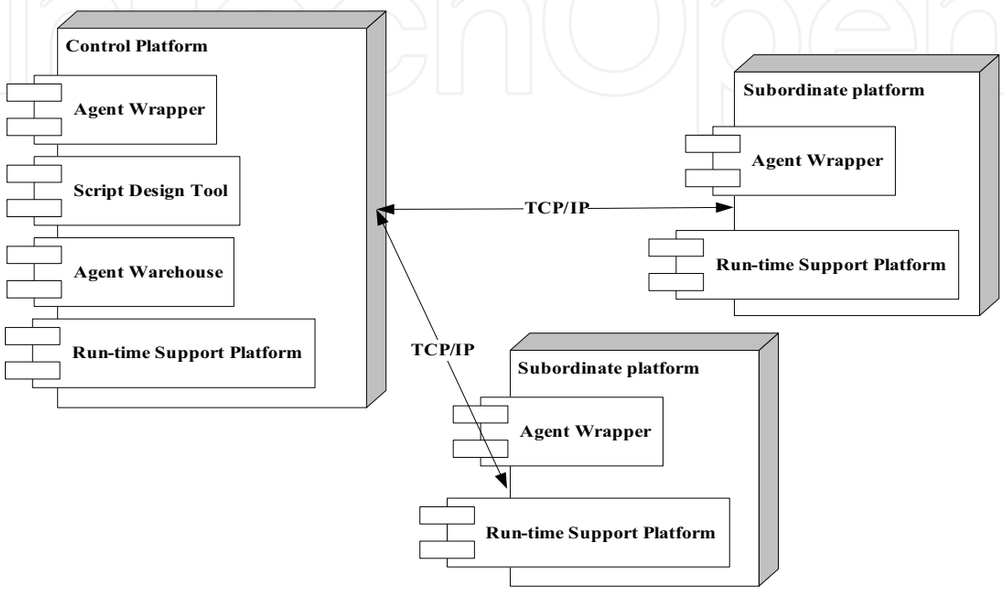


Fig. 7. Deployment Diagram of ASIDE

In addition, to support the distributed system integration, Control Platform (CP) and Subordinate Platform (SP) are developed, which are deployed in distributed computers. One Control Platform and several Subordinate Platforms are needed in the distributed integration and different tools of ASIDE are installed in them to support the integration, the deployment diagram is shown in Fig 7. ASIDE are deployed in Control Platform while only Agent Wrapper and Run-time Support Platform are deployed in Subordinate Platform. During the integration process, the agents are loaded and deployed in Control Platform and Subordinate Platforms, and the agents' information (especially the information of their capabilities) is immediately detected and recorded by CRC in Control Platform. After the integration script is loaded in Control Platform, the related agents are dynamically controlled and organized to accomplish the integration.

5.2 Agent-based system integration development process

The integration development process is divided into three phases as shown in Fig.8. Integration unit division is accomplished during requirement analysis phase, in integration design phase, agent design and integration logic design is completed. During run-time integration phase, the agents are deployed in Agent Warehouse and managed by Run-time Support Platform. After the script is loaded, the interpretation of script will lead to the occurrence that the integration regulations is distributed to the related agents, when the agents receives the regulation, they will immediately act in terms of the regulation to carry

out the integration. In this phase, once the required capabilities by an agent are lost with the owner or lacked, the bidding mechanism will work to find the agent who is idle and owns the same capability, Meanwhile, the agents who satisfy the bidding demand will initiatively make a bid. Eventually the agent will choose the best one from the bidders. The bidding mechanism copes with the environment changes (the occurrence and disappearance of agents in the platform) dynamically and flexibly, which achieves the adaptive integration. Additionally, when the integration requirement changes, the interpretation of the new script controls the collaboration of the related agents to perform the new integration, this process does not need to interrupt the running system, and thus it achieves the dynamic and flexible adaptation to the requirement changes, which is the dynamic integration presented in this paper.

The system provides two types of interfaces, one faces to the general users, and the other faces to the integrated development users. The Integrated development users though the analysis of specific areas extracted the business processes of the field, use script language to define integrated rule according to the business processes, and add the rules into integration rules storage of this field, simultaneously abstracts the relatively complete functional module in the business processes to use as agent. For general users, firstly should input the integration needs in the man-machine interface, then the integrated development users invoke the integrated rules of the related field, map the integrating demand into integrating script .The control integrated agent controls service agent collaboration though explanting facility of integrated script, and achieves the goal of the system dynamic integration.

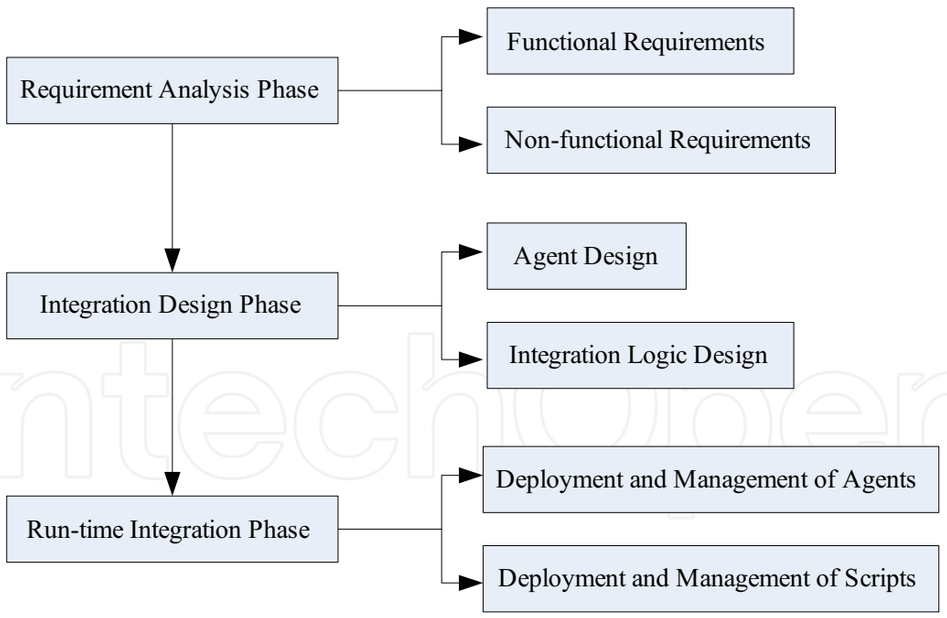


Fig. 8. Integration Development Process

After users input their requirement in human-computer interface, the integrated development staffs will call the integrated rules from corresponding field and map to integrated script based on the users' demands. The centralized control agent as a special agent calls the script interpreter to explain the integrated script. When running into a simple capability of the script, the centralized control agent will send a message to CRC looking for

agents with the capability. After receiving the searching result from CRC, the centralized control agent will send messages to them to request executing the task. If the agent holding the capability cannot complete the task by itself, it will request collaboration with other agents through the contract net protocol based on agent active perception. After the agent completes the task, it will inform the centralized control agent of the result for continuous execution of the next task. In this framework, the centralized control agent, CRC, AMS, PIB as well as other agents communicate with each other through information mechanism, which can support synchronous and asynchronous communication, and can bring about convenience to build distributed systems.

5.2.1 Requirement analysis phase

Requirements analysis is critical to the success of a development project , Requirements are a description of how a system should behave or a description of system properties or attributes. The functional requirement of system integration is integrating the sub-systems to construct a new system which accomplishes a certain function and the non-functional requirement is that the integration should be dynamic and flexible so that it can cope with the changing integration requirement and environment.

5.2.2 Integration design phase

Two types of Agents are designed, which are Function Agent and Service Agent. Function Agent is a concrete entity which possesses some basic capabilities while Service Agent is an abstract entity which organizes the Function Agents together in terms of certain collaboration logic to provide a high level service.

After analyzing the system integration requirement, the basic function modules (if not exist) are designed and implemented to provide a specific function. Afterwards, With Agent Wrapper, single function module is packed into a function agent (as shown in Fig.9 and Fig.10) and then several function agents are packed into a service agent by being defined as the acquaintances of the service agent. In addition, the service flow which depicts the collaboration relation of the organized function agents is also defined in service agents. Moreover, the service flow checking is also accomplished in the wrapping process to ensure the reliability and integrity.

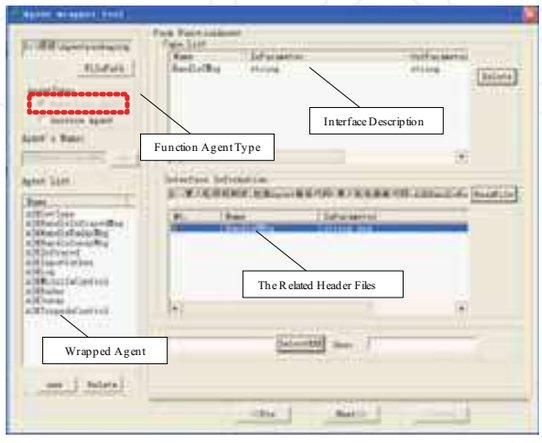


Fig. 9. Wrapping of Function Agent

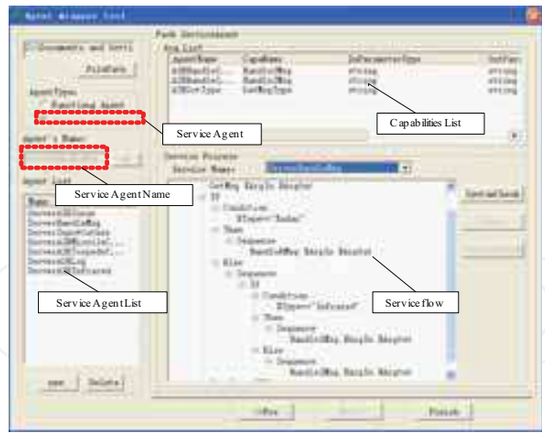


Fig. 10. Wrapping of Service Agent

DCISL (Dynamic Control Integration Script Language) is introduced in ASIDE, which depicts the integration control logic which will control the interaction and collaboration of the agents to accomplish the integration. Script Designer is the tool for designing the script in DCISL and it has Text Mode (as shown in Fig.11 and Fig.12) and Graphic Mode. Moreover, the semantics, syntax and integration logic checking are also involved in Script Designer to ensure the correctness, reliability and integrity of the integration logic.

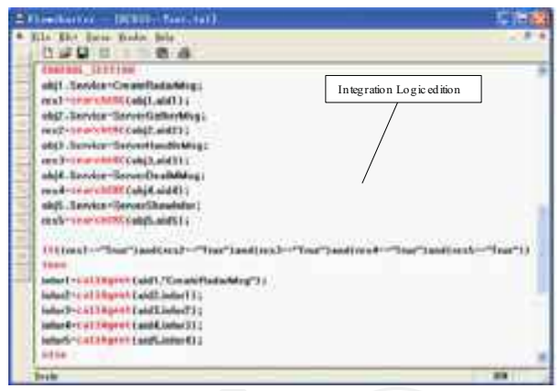


Fig. 11. Editing Script in Text Mode

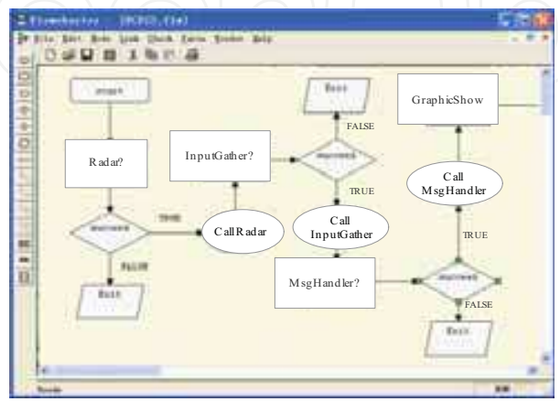


Fig. 12. Editing Script in Graphic Mode

Features like naming conflict, different format, and dissimilar structure of the data in different agents might cause inconvenience during the system integration process. To deal with these issues, Data Integration strategy is proposed in ASIDE, which involves data integration engine and plug-in, the extended functions are implemented by plug-ins which is produced by the data integration development tool.

In the system design phase, the data conversion relations are gained by analyzing the interface information of agents. With the data integration development tool, the data conversion regulation files are generated in terms of the data conversion relations (as shown in Fig.13), which are the plug-ins. After being deployed in the platform, the data conversion regulation files are extracted the related information by the engine to perform the data conversion among the agents.

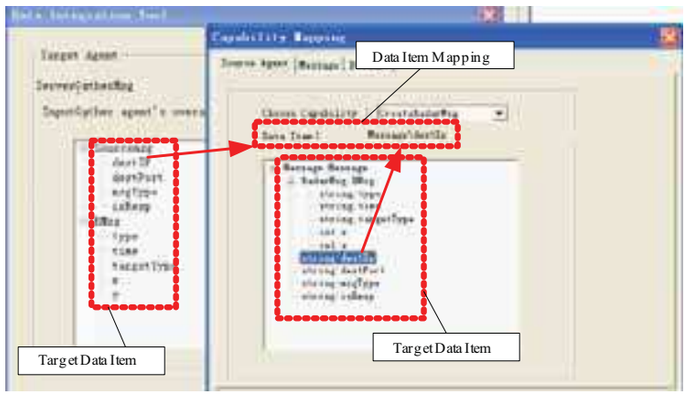


Fig. 13. Data Integration Development Tool

5.2.3 Run-time integration phase

With the tools in Run-time Support Platform, the intelligent and dynamic integration process is obtained. AMS (as shown in Fig.14), Integration Control and CMB (as shown in Fig.15) are the three critical tools among these tools to implement this process.

AMS is in charge of managing the overall agents which exists in CP and SP. When some new agents are loaded into Run-time Support Platform, AMS will perceive them immediately and allocate resource to them. Additionally, the related information of the agents can be queried real-timely in AMS.

Integration Control is used to load and interpret the script-based integration control logic to control the system integration. The interpreted integration control logic is distributed to the related agents and they store the logic as their own regulation and act according to the regulation to accomplish the integration task. When the integration requirement changes, the integration script is altered accordingly and then the new integration regulation is distributed to the agents, it replaces the old one and conducts the agents to accomplish the new integration.

In the integration process, the federated acquaintance-first strategy is applied in the collaboration among agents. Service Agent firstly obtains service from its acquaintance (function agent), if the required capability by the Service Agent is not possessed by its acquaintances, it will try to find the appropriate agent through inviting public bidding. The other function agents who have this capability will bid initiatively and provide service if chosen. The biding process is monitored and displayed real-timely in CMB.

Additionally, Monitor Tool monitors and displays the communication and interaction amongst all the Agents which exist in CP and SP. CRC deployed on CP maintains a list of overall Agents' capabilities. When the new agents are loaded, the related capabilities information is registered in CRC. The query function is also provided to inquire the ability information of certain agent which is loaded in certain platform.

Fig. 14. AMS

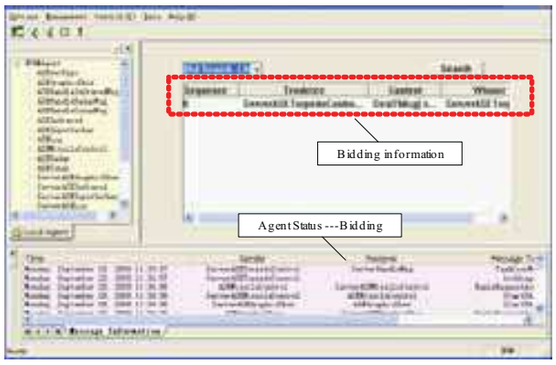


Fig. 15. CMB

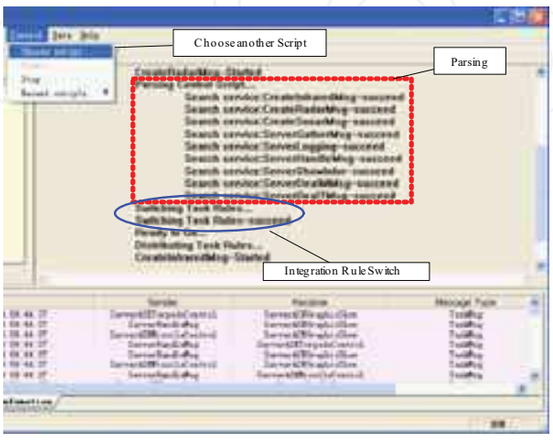


Fig. 16. Integration Logics Switch

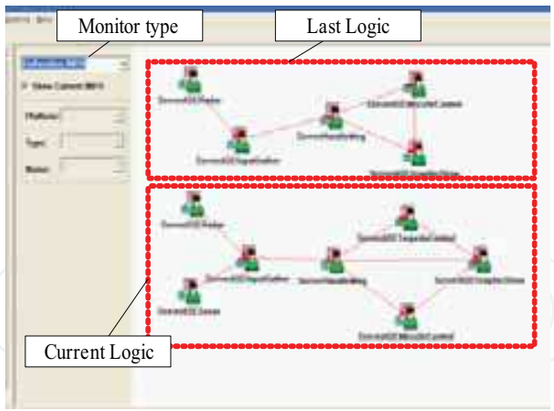


Fig. 17. Monitor the Scripts Switch

By choosing the first integration requirement script and loading it into the Control Platform, the script is interpreted and distributed to the agents who collaborate with each other in terms of script, as shown in Fig.16. With the system keeping running, the second script can be loaded into it and the new integration will be performed, the related switch information is shown in Fig.16.

When switch to the next script, the absence of ASETorpedoControl agent results in the suspending of the integration process. Therefore, the service agent ServerASETorpedoControl will bid for the capability of DealMsg, the bidding information is shown in Fig.15. After loading the ASETorpedoControl function agent, the required capability is provided by ASETorpedoControl agent and the process is activated to continue the integration. The monitor center displays the collaboration relation before and after switching in Fig.17.

6. Conclusion and future work

With the rapid development of computer software and hardware technology, the system scale is more gigantic and the function is more intricate, how to reuse and integrate the existing system efficiently to construct large-scale system is the emphasis of many researches. Though the traditional component-based system integration strategy brings some solutions towards this issue, it lacks flexibility and dynamic especially when the integration requirement keeps changing. To address this problem, the Agent-based software integration approach is put forward and implemented in this paper which applies the agent technology into the system integration to intelligent the integration process. With the agent wrapper, the function modules are wrapped into agents who are self-determined and self-adaptive. With the script-based control integration logic, the integration requirement switch is realized flexibly. With the bidding mechanism based on Contract Net Protocol, the required capabilities are provided by the agents who possess the ones so as to continue the integration process.

The contract net protocol is a simple and dynamic scheme but is communication intensive due to the broadcast of task announcements. Its performance degrades drastically when the number of communicating agents and the number of tasks announced increases. This limits its usability in a large-scale multi-agent system. In this thesis, the Contract Net Model based on Agent Initiative Perception is proposed. The functions of the perception coefficient and

the degree of credibility are introduced to study the changes of environment and agent itself. The parameter for the perception coefficient represents the current overload of an agent, and the parameter for the degree of credibility shows the completion history of tasks having been finished by the agent. The contract tender decides to bid or not in terms of the bidding strategy, and the contract manager consigns tasks to agents based on the tendering strategy, that improve the system efficiency. However, when the number of tasks is small, the set of the intersection of these agents' capabilities is nearly empty. Each agent in the system will perceive tenders that increases the communication overhead of the system. As part of the future work, we intend to go further improving CNMAIP and using CNMAIP in the Agent-based System Integration Tool. At the same time, we will also further study how to motivate an agent when the tasks are finished by the agent successfully and how to punish an agent when the tasks are not complete on time.

7. Acknowledgement

This work is supported by the Defense Pre-Research Project of the 'Eleventh Five-Year-Plan' of China under contract number 513060401 and the Fund of C4ISR National Defense Science and Technology Key LAB under contract number 9140C8301011001; In addition, it is supported by the 'Fundamental Research Funds for the Central Universities'.

8. References

Rabelo, R. Camarinha-Matos, L., Afsarmanesh, H. (1998).Multi-agent Perspectives to Agile Scheduling. *Intelligent Systems for Manufacturing*, pp. 51-66. Kluwer, Netherlands.

Shen, W., Norrie, D. (1998). An agent-based approach for dynamic manufacturing scheduling. In *Proc. of the 3rd International conference on the practical applications of agents and multi-agent systems*, pp. 533-548. London, UK.

R.G. Smith, R. Davis. (1980) The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Tran. Computers*, 1980, C-29(12): 1104—1113.

T. Sandholm. (1993) . An Implementation of the Contract Net Protocol Based on Marginal Calculations. In *Eleventh National Conference on Artificial Intelligence*, pages 256-262, 1993.

T. Sandholm, V. R. Lesser. (1996).Advantages of a Leveled Commitment Contracting Protocol. *In AAAI*, 126-133. 1996.

M. D'Inverno, M. luck. (1996). Formalizing the Contract Net as a Goal-directed System. In *Proc. 7th MAAMAW*, Eindhoven, Netherlands, LNAI Series, 1038, pages 72-85,.

E. Werner. (1989). Co-operating Agents: A unified theory of communication and social structure, In *Distributed Artificial Intelligence*, II, M. Huhns and L. Gasser, Eds. Pages 3-36. Kaufman and Pitman, London,.

M. Fisher, M. Wooldridge. (1994).Specifying and Executing Protocols for Cooperative Action. In S. M. Deen, editor, CKBS-94 --- *Proceedings of the Second Internateional Working Conference on Cooperating Knowledge-Based Systems*. Springer- Verlag,.

R. Brooks. (1991). Intelligence without Representation, *Artificial Intelligence*, 47(1-3), 139-159.

Rao A S, Georgeff M P. (1991). Modeling rational agents within a BDI-architecture. *Proceedings of the 2nd International Conference on Principles of Knowledge*. Representation and Reasoning, 473-484.

ZHANG Shuang-Min and SHI Chun-Yi. (2002). Modeling Agents Combined with the State of Environment. *Computer Research and Development*. Dec 2002, 39(12), 1587-1591.

WANG Yi-Chuan, SHI Chun-Yi. (2003). A Concurrent BDI-Agent Model. *Jounal of Software*. 2003, 14(3), 422-428.

DONG Ming-Kai, Zhang Hai-jun, SHI Zhong-Zhi. (2004). An Agent Model Based on Dynamic Description Logic. *Journal of Computer Research and Development*. May 2004, 41(5), 780-786.

Wooldridge M, Lomuscio A. (2000). Multi-Agent VSK logic. In: Ojeda-Aciego M, deGuzman IP, Brewka G, Pereira LM, eds. *Logics in Artificial Intelligence*: European Workshop, JELIA 2000. LNAI 1919, New York: Springer-Verlag, 300-312.

HS Nwana, DT Ndumu, LC Lee, JC Collis. (1999).ZEUS: a toolkit and approach for building distributed multi-agent systems. *Applied Artificial Intelligence*,.

Jennings N R, Sycara K, Wooldridge M. (1998). A Roadmap of Agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1998:1275-306.

Wekman K.(1991).Using negotiation and coordination systems. *Proceeding of 12th International Joint Conference on Artificial Intelligenc*e, 187-198.

R.G. Smith, R. Davis.(1980).The contract net protocol: High level communication and control in a distributed problem solver[J]. *IEEE Tran. Computers*, 29(12):1104-1113.

Haishun Yun, Qingshan Li ,Dan Jiang, He Liu, Shaojie Mao, Yuping Li. (2009). A Contract Net Protocol Based on Information Intermediary Service in Multi-agent System. In *proceedings of 2009 International Conference on Artificial Intelligence and Computational Intelligence*, IEEE-CS Press. pp.

Yingqiang Wang, Qingshan Li, Chenguang Zhao, Hua Xing. (2008). A Multi-Agent System Frame Model for Dynamic Integration. In *Proceeding of the 9th International Conference for Young Computer Scientists*, ICYCS 2008. IEEE CS Press. Zhangjiajie, China. p.1163-1168.

Dan Jiang, Qingshan Li, Haishun Yun. (2010) .Collaboration Algorithm of FSMAS. In *proceedings of the International Conference on Swarm Intelligence* (ICSI 2010). Springer LNCS, Vol. 6145, June 12-15, Beijing, China. pp.390-296.

Qingshan Li, Weihua Sun, Lili Guo. (2010). An Agent-based Dynamic Integration and Adaptive Evolving Model in Software Development. *Journal of Communications of SIWN*. June-July 2010. Vol.8-9.

Meisheng Wang, Qingshan Li, Chenguang Zhao. (2008). A contract net model based on agent active perception. In *Proceeding of the 32nd Annual IEEE International Computer Software and Applications Conference*, COMPSAC 2008, Turku, Finland. pp.511-516.

Chenguang Zhao, Qingshan Li, Meisheng Wang, Yingqiang Wang, Yuping Li. (2008). An agent based wrapper mechanism used in system integration. In *Proceeding of the 2008 International Conference on e-Business Engineering*. ICEBE 2008. IEEE CS Press. Xi'an, China. p.637-640.

Weihua Sun, Qingshan Li, He Liu, Lei Wang, Shaojie Mao, Yuping Li. A Scripted-Control Integration Strategy in Multi-agent System. In *proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence* (AICI'09), IEEE-CS Press. Nov. 2009, Shanghai, China. pp. 263-267.

Shanshan Hao, Qingshan Li, Meisheng Wang, Hua Xing, Chenguang Zhao. A Systems Integration Oriented Multi-Agent Hierarchy Cooperation Model. *International*

Conference Computer Science and Software Engineering, CSSE 2008. IEEE CS Press. Wuhan, China, 2008. 452-455

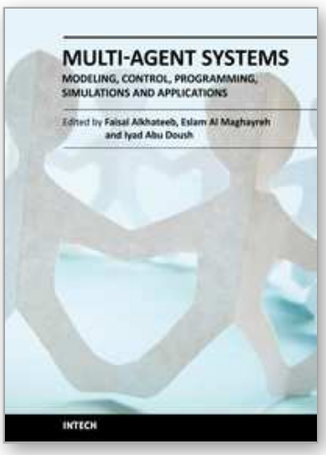
Wooldridge M, Jennings N R. Intelligent Agent: Theory and Practice. *Knowledge Engineering*, 1995, 10(2).

Ed Ort. Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools [M]. *SunMicrosystems*. 2005.

Cheyer A, Martin D. The Open Agent Architecture, *Journal of Autonomous Agents and Multi-Agent Systems*. March 2001. 143~148.

Chu WJ and Tolone, et al. Integrating Manufacturing Softwares for Intelligent Planning Execution: a CIIMPLEX Perspective.

Shen W, Norrie D H. A Hybrid Agent-Oriented Infrastructure for Modeling Manufacturing Enterprises. In: *Proc. of KAW'98*, Banff, Canada, 1998(Agents-5: 1~19).



Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications

Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-174-9
Hard cover, 522 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Qingshan Li, Lili Guo, Xiangqian Zhai and Baoye Xue (2011). Intelligent Collaboration Environment in Multi-Agent System Enabling Software Dynamic Integration and Adaptive Evolving, Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications, Dr. Faisal Alkhateeb (Ed.), ISBN: 978-953-307-174-9, InTech, Available from: http://www.intechopen.com/books/multi-agent-systems-modeling-control-programming-simulations-and-applications/intelligent-collaboration-environment-in-multi-agent-system-enabling-software-dynamic-integration-an

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen