Study of SOA Component Dynamic Scheduling Based on Mobile Agent Coalition

Tao He[1], Bin Tang[2], Leqiu Qian[3]

Computer Science and Engineering School, Fudan University, Shanghai 200433, China
[1] eGrid@msn.com, [2] edesk@126.com, [3] iAgent@163.com

ABSTRACT

Service-oriented components differ greatly with the traditional ones in the Service-Oriented Architecture. The ways of scheduling components seamlessly according to the agile computing needs to fit the e-business requirements is the key technology in the highly distributed, paralleled environment. In this paper, Based on the Multi-Agent Coalition, a new service-oriented component dynamic scheduling model is proposed, including the Multi-Agent Organization to schedule and coordinate the component assembly, the design of virtual execution task list table and self-learning algorithm, the definition of the Services component model, and the mechanism of collaboration Agents to search, discovery, concurrent schedule, dynamic assembly when execution in an heterogeneous network environment. To a large extent, the thesis solves the traditional problem of over-emphasis on centralized control logic, which leads to lacking flexibility in e-Business computing presently, and helps e-business service-oriented components become more adaptive, mobility and intelligence.

Keywords: Service-Oriented Component, Multi-Agent, e-Business, Scheduling, Algorithm

1. INTRODUCTION

Automation is the most effective way to improve the efficiency of e-business. With the widely use of Service-Oriented Architecture software engineering, there is at least two problems emerged: too much service-oriented components which lack of automatic management system and too weak technology now-used which cannot support the dynamic and agile computing to meet e-business demand, especially to fully considerate the enterprise individuation. Multi-Agent technology as the research focusing on Artificial Intelligence and distribution computing, gives an effective way to solve these problems, which make business becomes more agility and interconnectivity.

A new service-oriented component-scheduling Model based on the Multi-Agent Coalition is proposed. In the architecture, every kinds of agent can enhance mobile agility by self-learning mechanism and by storage of personal information in every dynamic and distributed execution. This approach use a fully new method model to construct the virtual business activities by simulating transactions in the real business world, such as locate customs, negotiate with them, and so on. We use the model in the following by build a MAGE (Multi-Agent Environment) to put the conception into practice, and the process of e-business is simplified greatly.

2. DYNAMIC COMPOSITION

Agent-based component dynamic composition is the fundamental technique to shift from traditional components to virtual service-oriented ones. With the help of agents, components can aggregate or compose existing services and component into new styles. Some components are virtual ones when execution, in fact, they don't exist at all. Such would allow third–parties to provide value–added components, which is similar with the real world services business. Five new features of services component are defined here.

Dynamic Existence It will not only being an entity as used to be, but a function correlation with the time.

Autonomy The Agent-based Services Component has the ability to take its own decisions in an independent way, including selecting the best of the available options according to e-Business computing requirements, Self-adaptation to different interfaces and protocols, being flexible, polytypic and extensible.

Competitiveness It should be competitive for a candidate to integrate into the SOA. More efficient and accuracy should be achieved with the lowest costs, and the self-learning mechanism which enhance the performance of components after execution is vital.

Sociability Communication protocol of different Agents is needed to enable them interaction with each other.

Semantic matching Semantic matching of component descriptions will make composition more transparent.

There are two basic composition styles under the control of composition Agent. One is Hierarchical Composition, and another is dialogic Composition.

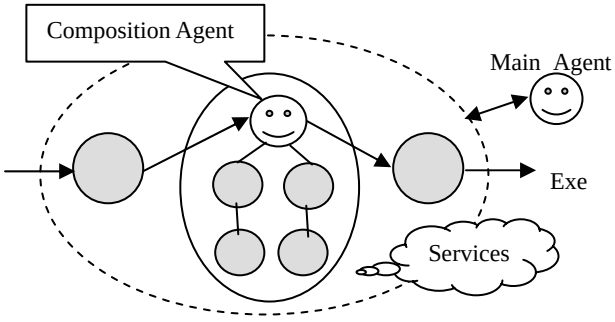

Fig. 1 Agent-based Hierarchical Composition

In hierarchical compositions, some related components cooperate under the agent coordination act as a new component at a higher level.

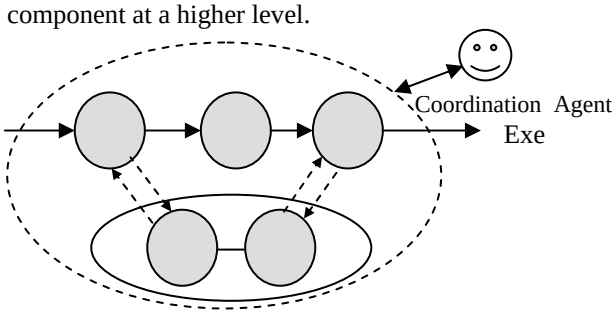

Fig. 2 Agent-based dialogic Composition

In the dialogic pattern, interacting components are viewed as peers. One component may make use of the other by exchanging data or control signals supervised by coordination Agent.

3. RUN-TIME ASSEMBLY MODEL

The service-oriented component dynamic scheduling and assembly model during the run-time should be predictable, applicable and manageable. We investigate the mechanism from three perspectives including Agent Coalition Perspective, Component Dynamic Assembly Perspective and the Inside Control Gene Perspective.

3.1 Agents Society Architecture in the Model

According to the dynamic assembly requirements, a main control-agent is introduced. It is an agent stored in migrating node with a virtual task list, and when the main agent runs in the migrating node, these service-oriented components can be assembled dynamically and the task table can be taken down after running at a previous node.

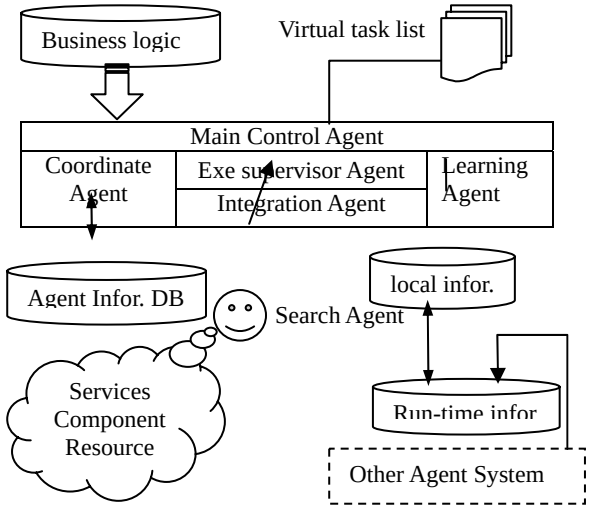

Fig. 3 Agents Society of Run-time Assembly Model

Secondly, the mobile Multi-Agent coalition is composed of some other kinds of agents including: services search agent, execution supervisor agent, learning agent, integration agent, coordinate agent and safeguard agent, which make distributed services component composite, integrate, scheduling dynamically according to business logic. All of them function and cooperate with each other to overcome the shortcoming of traditional e-business system and create a real-time virtual cyber enterprise.

3.2 Component Dynamic Assembly Model

The Dynamic Assembly Model covers the whole process of the service-oriented components discovery, composition, negotiation, and orchestration dynamically and seamlessly, which makes it possible when fulfilling some complex commercial tasks by interacting with distributed component resources. Nowadays, many distributed service-oriented applications designed to execute on Grid Computing Environment, require the simultaneous co-allocation of multiple service-oriented component resources in order to meet high business performance frequently. The approach via agent coalition mechanism for services allocation and management architecture is very helpful to the co-allocation resource efficiently. With the help of Mobile Agents, new component clusters are generated and coupled to run at a higher level. So, the main focus is to design more adaptive component interfaces to enable the assembly and scheduling process more flexible and context-aware. Agent is the best solution.

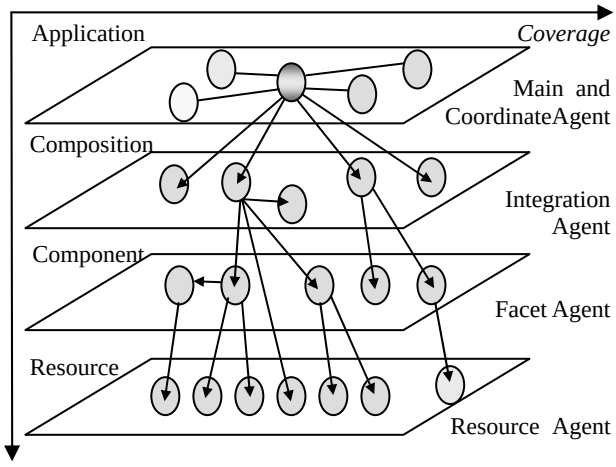

Fig. 4 Component Dynamic Assembly Model

The Facet Agent based on XML encapsulates the detail information of interface and was supported by the component assembly history information warehouse. It has self-descriptions and can communicate with other component agents to composite into a new style according to the e-Business requirements and the characteristic information retrieved from history data. Main Control Agent can migrate to other node to supervise the compiling and execution of local assembled component according to the Virtual Task List optimized by dynamic scheduling Algorithm. After each successful running, some key information about the component will be inserted to the knowledge base.

3.3 Run-time Agent Gene Design

A soft gene for agent is a hereditary unit that determines some particular characteristics in a component. After adopting certain soft genes, the software component can make certain practice activities, update its own behavior rules, etc. the main agent gene is the soul of the system in the run-time assembly process. It is define as XML, whose items can be easily updated by operation classes. We give the main definitions here:

```
<? xml Version="1.0" encoding = "utf-8">
  <Main Agent url=" " KnowledgeBase=" " …>
      <Agent id=0001 type="ExeAgent">
        <Infor>It is a Exe Supervisor Agent!</Infor>
        <IP>12.121.10.156</IP>
        <KnowledgBase>url </KnowledgeBase>
        <DataPool>virtual execution lists</DataPool>
      </Agent>
      <Agent id=0002 type="SearchAgent">
        <Infor>It is a Search Agent!</Infor>
        <IP>12.121.10.100</IP>
        <KnowledgBase>url </KnowledgeBase>
        <DataPool>search requirements</DataPool>
      </Agent>
      <Agent id=0001 type="CoordinateAgent">
        <Infor>It is a Coordinate Agent!</Infor>
        <IP>12.121.10.101</IP>
        <KnowledgBase>url </KnowledgeBase>
        <DataPool>coordinate relations</DataPool>
      </Agent>
      <Agent id=0001 type="CompositeAgent">
        <Infor>It is a Composite Agent!</Infor>
        <IP>12.121.10.1<IP>
        <KnowledgBase>url </KnowledgeBase>
        <DataPool>component id</DataPool>
      </Agent>
      ……
  </Main Agent >

Gene MainAgent
{    Attribute:
        double   Skillness,Intelligence,LeaderAbility;
        string   IPaddress, State, Task ;
        struct   *VirtualExeList, *MessageQueue ;
        Agent    *xAgent[N] ;
     Method:
       InitiateAllAgent( );
       UpdateGene(XML DOM)
       SendControlMessage(Agent *Target);
       CheckGoal(*Compnent);//whether meet the goal
       Scheduling(struct *VirtualExeList);
       MigrateToDestin(string IPadress);
       ……
       Destroy( );
}

Gene ExeAgent
{   Attribute:
        long int TaskNum;
        struct   *VirtualExeList, *MessageQueue ;
        …..
   Method:
       OptimizeExeList(*VirtualExeList);
       UpdateGene(XML DOM)
       …..
}

Gene CompositeAgent
{   Attribute:
        long int ComponentNum;
        struct   *Compo, *MessageQueue ;
        …..
   Method:
       ComponentInsert(*Compo, TagetAddress);
       ComponenDelete(*Compo, SourceAddress);
       SearchRequest(string Requirement)
       UpdateGene(XML DOM)
       …..
}……
```

With the help of Agent Gene revised and optimized adaptively, components can be easy understood, assembly according to the commercial computing requirements compared with that of before.

4. SCHEDULING ALGORITHM

4.1 Agent-based Concurrent Algorithm

Agent-based concurrent scheduling algorithm processes requests for service between agents and the executions of the tasks list. We mainly use the Enhanced Coffman-Graham Algorithm to perform concurrent scheduling to reduce the time of execution. Applicable components of systems are organized logically according to the optimized virtual execution task list.
:

Algorithm

Tasks List: $T=\{T_1, T_2,\ldots, T_n\}$ $a(T_i)$: T_i *Tag*

T_i *successor aggregate :* $S(T_i)=\{T_k | T_i \Theta T_k\}$

$N_{EC}(T_i)$: $Max\{time(T_i+S(T_i))$

Step1: Include tasks without successor and tag as {R};

Step2: for i=1 to n do

{ *a. Calculate* $N_{EC}(T_j)$ *of all Tj in {R};*

b. take out the Task with $Min(N_{EC}(T_j))$*;*

c .if $Min(N_{EC}(T_j))>1$*, Random select a* T_R*;*

d. $a(T_R)=i$*;*

e. Add the task Which has no Tag but its successors are all tagged into {R} in T_R *preceding aggregate.*

}

Step3: Construct $L'=(u_n,u_{n-1},\ldots,u_2,u_1)$*, to ensure* $a(u_i)=i$ $(i=1,2,\ldots,n)$*;*

Step4: Eliminate virtual tasks in L' to construct L;

Step5: To $(T \Theta L)$*, use common Graham algorithm to scheduling tasks in L;*

Reference: common Graham algorithm[7]

4.2 Mismatch Component Assembly algorithm

The semantic matching of service-oriented component is always hindered by mismatches when assembly. Formalized definition of matched component is:

a) Match(O, O')=$m_{spec}(O_{spec}, O'_{spec})\wedge m_{sig}(O_{sig}, O_{sig}')$

If O and O' signature and specification match, then they are matched components.

b)R_j:the e-Business Required Components vector space

C: the exist component in service environment

$$M(C, R_j)= \frac{\sum_{i=1}^{N} r_{i,j} \times c}{\sqrt{\sum_{i=1}^{N} r^2_{i,j}} \times \sqrt{\sum_{i=1}^{N} c^2_{j}}}$$

If $M(C, R_j)<allow_{min}$, then they are match.

When the Search Agent find unmatched component, it can solve in the follow ways by coordinate agent:

- Send a message to Main Agent to adjust to suit the component if the execution can meet the goal
- Search another component to composition as a matched one
- Use Interface Agent to mask unnecessary function and parameters
- Function as middle ware to cover the discrepancy
- Duplicate and revise the original code to produce a new suitable component according to knowledge base.
- Specialized the component some control parameters to get the sub-function of a component
- Use learning mechanism to enhance the component function to match the requirement
- Translate the component I/O data format

4.3 Run-time Agent Self-learning Algorithm

Collaborative and Shareable Learning System is used to improve the performance of Mobile Agent Coalition on service-oriented components dynamic scheduling. It's the integration of Reinforcement Learning, Bidding, Genetic and other Algorithms.

We use shared knowledge and commercial logical by refining each agent's particular knowledge to improve the assembly of components and scheduling of task. During the run-time, the agents can dynamically alter the learning contents and presentation over time. Specifically, the agents can migrate and participate in a number of important tasks such as automatic composition of component according to the commercial logic, analysis and delivery services to users and adaptation to new patterns and services. This self-learning algorithm system is hierarchical in terms of its distribution to different agents. Agent Coalition can use learning mechanism to upgrade other algorithms includes component evolution and aberrance algorithm, virtual task list optimization algorithm, and commercial logical pattern auto-builder algorithm.

Every type of Agents can have its own and common knowledge base, locally or remotely, to build a cooperative, co-evolutionary mechanism in which agents can learn by using a team-based reward system. System will obtain the best team for achieving the task in e-business computing environment which requires coordination to succeed. These agents mainly use the auction mechanism to negotiate their roles dynamically. Agents bid individually according to their perceptions. The system chooses the best combination of bids for the team; the chosen bids may not be optimal for each individual, but the system learns to bid as a team and develops the best team-based strategy. We develop our learning algorithm in a distribute, simulated version of the real environment intelligence

5. CONCLUSION AND FUTURE WORK

Services-oriented Component based on Multi-Agent is the revolution to the traditional computing technologies, which make the information infrastructure be more quickly adapt to changing business priorities. Here we propose the framework and main part algorithms' brief view, which is an effective way to achieve our goals. But, some algorithms is far more satisfied, and still need to give deeply research for better solutions, especially for more matured Self-learning, effective assembly and efficient scheduling Algorithms.

ACKNOWLEDGEMENT

This work has been supported by the National Natrual and Science Fundation of China granted in this year.

REFERENCES

[1] S.McIlraith, T.C.Son, "Semantic Web Services". *IEEE Intelligent Systems, Special Issue on the Semantic Web*, 16(2): pp46-53,March/April 2001

[2] Laurent, "Component-oriented software technology". Niestrasz, "Object-oriented Software Composition", Prentice Hall International , pp3-28, 1995

[3] R.H.Reussner. "The use of parameterized contracts for architecting systems with software components". In W.Weck, J.Bosch, and C.Szyperski, editors, *Proceedings of the Sixth International Workshop on Component-Oriented Programming*, June 2001

[4] "Business Process Modeling Language", Accessed June 2004 from *http://www.bpmi.org*

[5] Van der Aalst WMP. "Petri Net Based Workflow Management Software". *Proceedings of the NFS Workshop on Workflow and Process Automation in information System*. Athens, Geogia, May 1996.

[6] Lawrence Wilkes, "Web Services Roadmap for On Demand Business", *CBDI Journal.*

[7] Daniel, Jon Feldman, "Parallel Processor Scheduling with Delay Constraints", in *Proc. Of the SAINT Symposium*, Jan.2001

[8] Luck, "Multi-Agent Roadmap". *www.AgentLink.org*