A Model Based Approach on Multi-Agent System and Genetic Algorithm to Improve the Process Management in Service Oriented Architecture

Behnaz Nahvi[1], Jafar Habibi[2]
[1]*Islamic Azad University, Science and research Branch, Tehran, Iran.*
[2]*Sharif University of Technology, Department of Computer Engineering, Tehran, Iran.*
jhabibi@sharif.edu

***Abstract*—Service oriented architecture is based on the provision of services. To enhance the performance of the systems by providing a better combination of services, it is necessary to extract more information compared to the one in the service registry. In this regard, the accomplished works have been focusing on the basic concepts of service-oriented architecture. The service composition is based on the information in service registry, provided by the service provider. Further, centralized combination with insufficient information does not meet the system performance requirements. This solution helps to facilitate resource distribution and reduces tasks of the central unit. In this paper, efforts have been made to use the agents in the proposed model to enhance the processes of a system. In the proposed model, agents have been used in order to extract and manage the essential information in service registry table. This information forms the basis of monitoring and selection of services. Agents are used for selecting and making efficient composite services. Finally, different communications and configuration mechanisms are implemented using multi-agent systems that can perform service-oriented architecture. Moreover, genetic algorithm is used to enhance architectural processes. In the proposed model, the genetic algorithm and multi-agent system has enhanced productivity of the system and its important quality attributes. The system runs in the international conference of computer society of Iran (ICCSI). Implementation of this model in the real environment and its comparison with its implementation on the prototype could be helpful to justify better efficiency and accuracy for future applications.**

***Index Terms*—Multi Agent System; Service Oriented; Process Management; Genetic Algorithm.**

I. Introduction

Service-oriented architecture provides a flexible and dynamic platform for the implementation of distributed systems. Considering the centralized management and control of systems reduces the performance of a system, taking the advantage of distributed methods can have significant positive effects onto its performance [1, 2]. Therefore, a model of multi-agent systems with service-oriented architecture has been proposed. The ability to manage processes, for distributed and decentralized can be added to the previous models. In this domain, managing processes within the structure is one of the existing challenges, which can significantly increase costs, decrease performance and productivity of the environment.

In service-oriented architecture, services cannot be inter-operated by themselves. Without automating services, negotiations, consistencies, arrangements, compositions, rules, and intense inspection of data and protocols, service-oriented architecture cannot do anything [3]. The concepts of service-oriented architecture have not been able to cover all these issues on its own without using other methods. Additionally, methods that have been used to overcome the existing service interaction problems using semantics have not been able to resolve it completely. Thus, this paper proposed a model of autonomous agents according to the rules of service-oriented architecture that can perform these functions and interact with other agents.

Process management can be done under the process of optimization [4]:It creates the relevant subject to service composition [5, 6]. From this perspective, various works have been reported [7-11], and they are still being done in order to resolve the problem and optimize the process management: Each work has its own specific features. Most of the work performed in the field of service composition focused on the problem of identification [12, 13], choosing services. There has been less attention to the fact that composed services are divided into two reusable and disposable groups. It is not necessary to assess and store the composed disposable services after execution. Storing the sequence of composed services, used for several times can substantially increase the performance of the system.

Related works performed in this domain recognize that designers tend to save the sequence during the design time in the system registry table. However there is a possibility that a large number of composed services re not to be known at the start of the usage. The main problem is the lack of some criteria to distinguish these two composed services from each other. Thus, in the proposed model, an agent that is responsible for the assessment and storage-composed service is generated at the run time.

On the other hand, the ever-increasing growth in the number of services [14-16], which provide the same function that have the solutions to validate, verify, describe, register and monitor the services based on their quality parameters is inevitable. Using a combination of agents and intelligence methods, such as genetic algorithms to solve these problems is a strategy that we have chosen.

In order to achieve high performance, a lot of methods have ignored the concepts of service-orientation, which results in the existing of new problems. Hence, protecting the service-oriented concepts has always been among one of the challenges that we are currently facing.

The use of multi-agent systems is one of the methods that allow the distribution of management and control in distributed systems [1, 7, 17-25]. Agent-based processing is an approach which develops complex systems by independent agents and eases their design [15, 26]. The attributes of agents make them flexible and robust entities. The flexibility is the result of agents' ability in response to environmental changes. Following the goals and work improvement leads to agents' robustness. Because of such quality, making use of agents in complex, dynamic, and open systems and error prone environments will result in improving the performance of systems.

The proposed method includes ten groups of agents. Necessary data is extracted by the agents and stored in a meta-data repository. It is updated and managed in service registry table and will be used during the runtime. Genetic algorithm is used to choose and prioritize the services. Scenario-based methods are used for evaluation; hence, in addition to feasibility study, the satisfaction level of quality attributes is studied.

The paper is organized as follows: in the second section, previous works are reviewed, and the concepts used in proposed framework are described in the third section. In the fourth section, a system that gives reservations plane tickets and hotel for persons is used to evaluate the proposed model, and the last section is devoted to the conclusions and future work.

II. RELATED WORK

Service composition is divided into two general groups: one of them is the time based composition, which is divided into static composition in design time and dynamic composition in runtime, and the other one is composition with human intervention, which is performed manually or automatically [27]. In the proposed model, services are composed dynamically at runtime leading to composite services. Automated compositions are usually based on a semantic approach in which both the data processing and the generation of the service composition scheme are performed using semantic approaches. In these methods, well-defined information models are usually used and are converted into a processable machine; ontologies are used to provide descriptions of functional and non-functional attributes of services in order to automatically perform discovery, selection, and composition of services [28, 29]. A new algorithm based on binary tree is proposed [30] for service composition which selects services, according to user's preferences and services quality issues, among composite services that are responsive to user's needs.

Maintaining history of service composition is one of the most important issues that should be considered in the proposed model. If the history is not recorded after each usage, the composition time will decrease the performance of the system in similar usages. Extended service-oriented architecture has been provided [31] in which it focuses on service composition for software development, especially on the data related to the composition and makes use of the dependency-aware service management and composition to extend services. Existing relations in the process model consist of controlling the systems flow, input and output. Service dependency seems to be considered from service composition management point of view in [33] log file, which is the available in service oriented architecture audit files have been used to determine the relations and dependency among services.

In [32], Service dependency graph, which is dynamically generated drawing is shown. Then an intelligent discovery algorithm has been used to find the minimum composition, which can meet the user's requirements.

In [33], a framework is presented to make composite web services. The environment implanted in the service-oriented architecture is a dynamic environment, in which the services are regularly changed, added, and updated. Thus, the web service composition at the initial stage should be performed based on the updated data. In this framework, semantic data are used to compose the services. Composition process in this framework is started through modeling service composition flow structure and limitations in graphical modeling interface. The graphical flow, limitations, and service patterns are translated and saved in composite web service language.

In [34], it has been tried to implement the service recognition framework using three groups of agents and three groups of data. Service recognition agents are:

i) User agent: start to discover and save the result in service directory.
ii) Service agent: it registers or deletes the services.
iii) Directory agent: it is responsible for replying to user agent by searching in the directory.

Intelligent service discovery can be upgraded by increasing the amount of provided data. Here are some of the mentioned data in [35] which lead to system upgrade:

i) User profile: the user identity is registered and authorizations are determined.
ii) Terminal profile: the terminal static data is saved.
iii) Context: it is the dynamic data of user and terminal like user status, place of residence, etc.

In [36], interactions between services are studied. Processes in the service-oriented architecture are done due to different purposes and one of them is processes which are registered and discovered to compose the services.

A lot of researches in the domain of services registry and discovery have presented different methods to deploy UDDI. In [37], services have been registered based on domain and in this way it has been tried to improve the usage of repository. One of the issues discussed in this research is not maintaining the relations between services and in order to solve this problem, the characteristics scheme of the service have been introduced and registry process has been done based on this scheme. In this way, the assortment of services using this scheme has been possible. In [7], the solution to services discovery in a dynamic way has been discussed. Further, the possibility of services replacement has been brought up.

Services re-registry at certain times show the services availability, which is a disadvantage to this method and increases the system overhead. Each service shows its availability in certain intervals. Instead of that, the agents could be used and in this way the processes, which are done to achieve this purpose are improved.

In the service-oriented architecture, agents have been used for different purposes agents and different roles. The composition agent is responsible for composing and executive coordination in a composed service and executive management agent is responsible for global controlling of other agents tasks. Management agent is responsible for global control of the composition of other components, which are called composer agent. The presented method to compose web services is based on the composition of the logical programs. In [19], a framework based on agents has been used to implement and deploy the service oriented architecture.

As shown in Figure 1, FUSION (Flexible and User Services Oriented Multi-agent Architecture) [38] is a modular multi-agent architecture, which services and application programs are controlled and managed by consulting BDI agents. They follow THOMAS architecture principles [39]. In FUSION, four fundamental blocks, which ease the architecture abilities are:

Application program: It represents all programs, which can be used to exploit the features of the system.

Agent platform: It is responsible for integration of a group of agents, which each has special features and attitude. Agents in this architecture are served as the controller and manager of all application programs and services.

Service: It is the system’s ability to process and obtain data.

Communication protocol: It allows the services and application programs to communicate directly with agent’s platform. This protocol is completely open and independent of any programming.

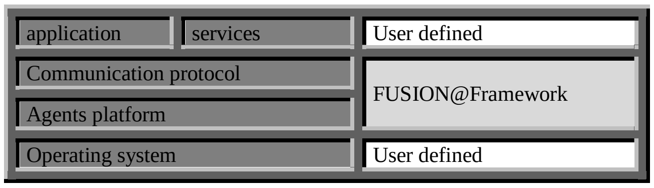

Figure 1: FUSION framework [38]

III. Proposed Framework

In our proposed model, we try to improve the performance by adopting optimization methods, as it was reported in [40] that the genetic algorithm is used to improve the architecture performance.

The objective of the proposed model is to achieve optimized processes of service oriented architecture. Positive and negative effects of the provided model in the fundamental concepts of service-oriented architecture are shown in Table 1. Advisory BDI agents are capable of collaboration and proposing a solution in the dynamic environment and they can encounter a real problem, even when there are restrictions on the issue and they have access to limited resources. For this reason, it is used in the proposed model.

Communication protocol allows services and applications to communicate directly with the agent's platform. The protocol is completely open and independent of any programming language. This protocol is based on SOAP standards to record all messages between the platforms and the services and applications. All external communications follow the same protocol. The relationship between agents in the platform is done by ACL which is the agents' communication language based on FIPA. We used agent platform for direct communication with agents; hence, the communication protocols are not required to be used in all cases. Nonetheless, we will require it to communicate with all the services.

Table 1
Impact of use of the proposed model in fundamental principle of service oriented architecture

Fundamental principle of service oriented architecture	Positive	Negative
Standardized Service Contract	The agent-based systems also will use this contract.	-
Loose Coupling	Factors use independent of location of services and coding in the whole system.	-
Service Abstraction	-	Part of the information that is available for agents beyond information that is available in the standard service contract
Service Reusability	-	Lower abstraction may limit reusability
Service Autonomy	-	-
Service Statelessness	-	The only case that will be kept last performances is response time and quality and number of run
Service Discoverability	dynamics of agents optimal Discoverability	-
Service Compos ability	Use of agents will optimum the combination	-

As shown in Figure 6, the presented model used ten categories, which include:

User interface agent: It is placed in layer of consumer and organizes a piece of the information needed in the model. In the layering of the reference of service-oriented architecture [18], information about the provided services is stored in layer of consumer by a building block (Figure 2). User interface agent is responsible for connecting agents, system and the applications. By benefiting from this agent, there is no need to use communication protocol and FIPA, ACL. The system also has a request for analysis sent to security agent, which is then given to the manager agent.

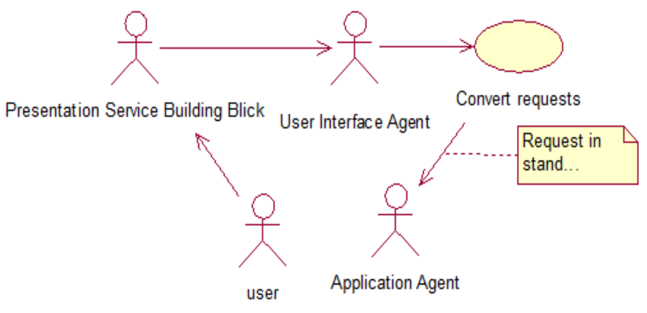

Figure 2: Use case diagram of a user interface agent to manage registration and service discovery

Applications agent: It is an agent that is responsible for communication between platform and application and manages incoming requests and service responds to the application. Application agent is always in standby mode.
Directory agent: It manages a list of services used by the system. Applications agent can utilize services that are listed in the platform. The Manager agent and the Directory agent commands discover the service in the system. Using the algorithm, agent directory select the best and most appropriate service or group of services (
Figure 3). Service can be added, removed or dynamically changed in this list. The efficiency of service, ES (average time to respond to the request), run number, RN and the Quality factor (QF) are information that is constantly changing. Quality factor is a very important information. First, for all services, QF is set at 1. When the service cannot work, (fail or cannot be found, or when performance is reduced compared to his previous performances) QF is reduced. The value of QF increases when a job is correctly done and the basis for selection of the service is correct.

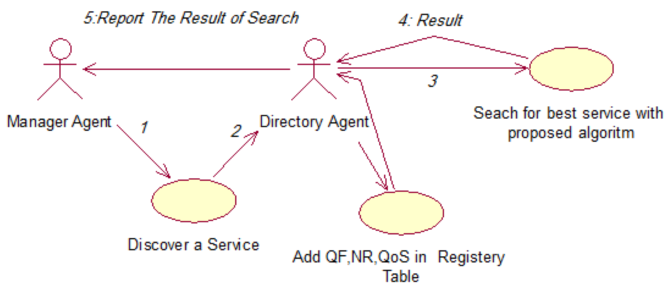
5:Report The Result of Search
4: Result
3
Manager Agent
1
2
Directory Agent
Seach for best service with proposed algoritm
Discover a Service
Add QF,NR,QoS in Registery Table

Figure 3: Use case diagram of the directory agent, the model proposed

Communication agent: It is responsible for the communication between applications, services, users and all components of the architecture. This agent is also responsible for the control of messages. Agent communications send a ping message according to a certain order for all the services. If a service does not respond to this message, the director says to find the same service and the agent directory says to reduce the amount of quality factor (QF).
Security agent: All messages are sent to the security agent and the agent's duty is to assess messages in terms of syntactic and semantic.
Administrator agent: It is responsible to monitor agent's function in the system and periodically confirms registered agent's status by sending ping messages. Administrator agent kills this agent in cases when it does not receive any response, hence it produces a new case of agent. In this case, there is an ever-present agent in the system.

If a message is a false information, the security agent informs the relevant agent (the applications agent and services agent) that the message is not delivered. Further, this message is sent to the directory agent to change the value of QF if needed.

With regard to the quality of service and user preferences, the administrator agent decides which agent should be called. User can explicitly request a service or allows a directory agent to decide which service is the best for the application. Directory agent also checks whether the service is working correctly or not, as shown in

Figure 4.

For any requests received, the communication agent sends the ping message according to a particular order to all services. When the service is not responding, the administrator agent tells to find a similar service and informs directory agent to change the amount of QF.

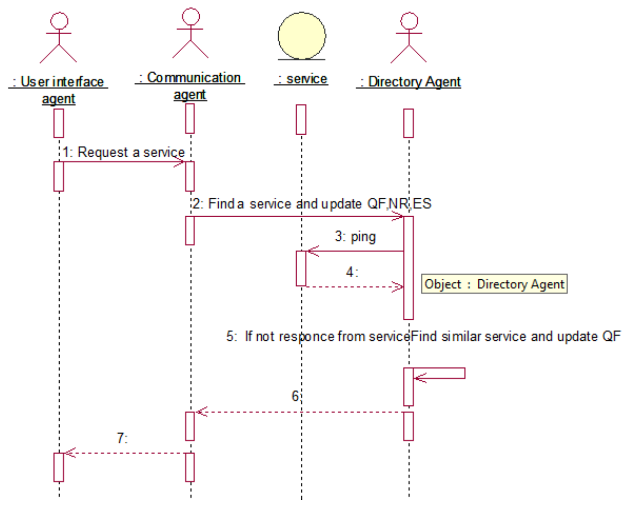

Figure 4: Sequence diagram when user allows a Directory agent to decide which service is best for the application

Service agent: It is responsible for all communication between the server and the platform (Figure 5).

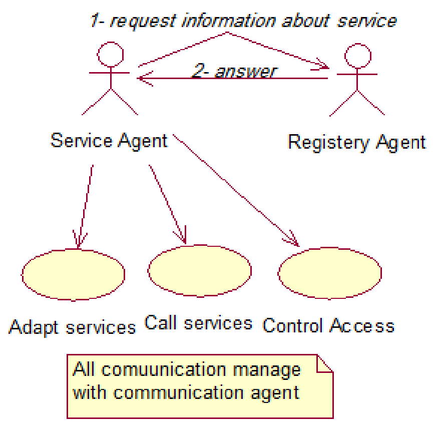
1- request information about service
2- answer
Service Agent
Registery Agent
Adapt services
Call services
Control Access
All comuunication manage with communication agent

Figure 5: Service agent function, the model proposed

Manager agent: It is the reasoning unit in the administrator agent that detects that this service is available as atomic or between a combinations of the registered services in the directory or not available in the directory. It also detects whether there is a service for a responsible agent and which service must be invoked in the absence of a desired application to compound agent.

Secure agent checks all messages in terms of semantically and syntactically. It also periodically checks the locations and operating modes and monitors whether they are unemployed (pending), corrupted or vice versa.

Composition agent: If the desired service is not found in the directory, this agent is responsible for the selection and composition of atomic services to build the desired service. In short, the performance of these agent include:

Divergence in flow: The current composition of services into several sub-streams, in which each of them should be implemented by the service agent. The duty of composition agent is to refer it to the manager agent that sends the service to the responsible agent.

Performance agent: This agent is responsible for the control and regulation of service composition.

Collecting the results: The agent component is responsible for gathering the results of the combined service and returns the results to the manager agent.

Grouping agent: To speed up the search and selection services, the same services are placed in groups. In the case of error or the need to restore a particular service, it refers to a group that can be easily detected in the corresponding service. Grouping agent puts eth same services in the same category. This classification can be done according to various criteria. For example, a consistent style for each of the subsystems should be considered as a parameter for classification. The diagram of the proposed model is shown in Figure 8.

Manager agent chooses a mechanism that consists of a set of methods that allows the architecture to achieve the most appropriate service at the time of requests. The mechanism for selecting the best service to answer the request when the work starts and when the new request is received is by considering the following parameters:

- The amount of quality factor (QF)
- User preferences
- Estimated delivery time
- The defined value of the fitness function

The first two parameters (*QF and User preferences*) are preset and they are not required to be calculated by the manager agent. The runtime must be computed by the method. Using predictive techniques, such as neural networks are one of the ways that the value can be obtained.

Using optimization methods in selecting the service helps the whole process to be performed with better speed and quality. One method that can be used to choose and give priority to services positioned in common groups is the genetic algorithm.

Chromosomes are made up of six genes services, such as service name, address service, the efficiency, performance and quality of service of Chromosomes genes.

The Fitness function was calculated according to Equation (1) and a value in the last genes was assigned as the basis for the priority selection.

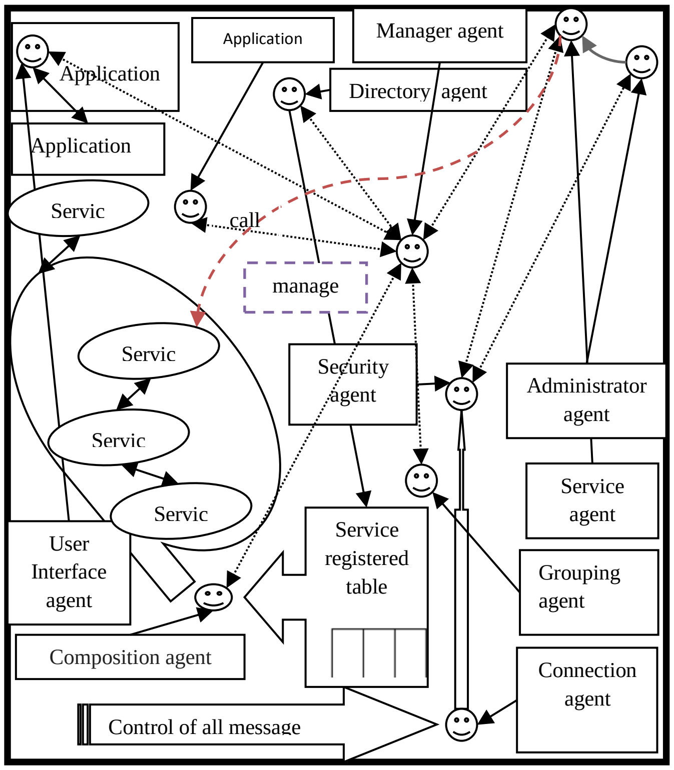

Figure 6: Proposed Model

$$\text{Fitness function} = \left(\sum \text{Num of run} + 1\right) * (\text{QF}) * \text{Response time}^{-n} \quad (1)$$

$0.0001 \ll QF \ll 1$
$0.1 \ll Response\ time \ll \text{max of response time}$

Max of response time set by manager agent
$1 \ll n \ll 5$
$0 \ll Num\ of\ run \ll \text{end of service lifecycle}$

For the evaluation of the model, each service is stored in the *QF*. The combined services that is the value of *QF* is calculated by Equation (2).

$$QF_{CS} = \frac{\sum_{i=1}^{n} QF_i}{n} \quad (2)$$

QF is the extent to which the initial value = 1 is given for all services. After each inappropriate execution or lack of response, it is reduced to the (0.001) amount. *n* is the number of services that is presented in a combined service.

Num of run is a symbol of satisfaction rate of performing the desired service, in which zero is the value at the beginning of pilot phase. Efficiency is defined as the time spent answering the response: Lesser responses are more suitable for us (For this, fitness function is given in the denominator).

An example made from the chromosomes is shown in

Figure 7.

Fitness function	performance	QF	number of run	address	name

Figure 7: Example of chromosome

In the optimum optimization and building of composite web service, genetic algorithm was applied through MATLAB programming language (Table 2). Each chromosome entails as many tasks as genes, where the equivalent service carries out the task. The framework was implemented through a real-life database, where the basic information concerning web services are available in conference's site.

Table 2
Elements of GA

Population size	Initial population	Selection function	Crossover function 0	Mutation function			Crossover
50	20	Roulette	0.8	Gaussian	Scale = 1	Shrink = 1	Scattered

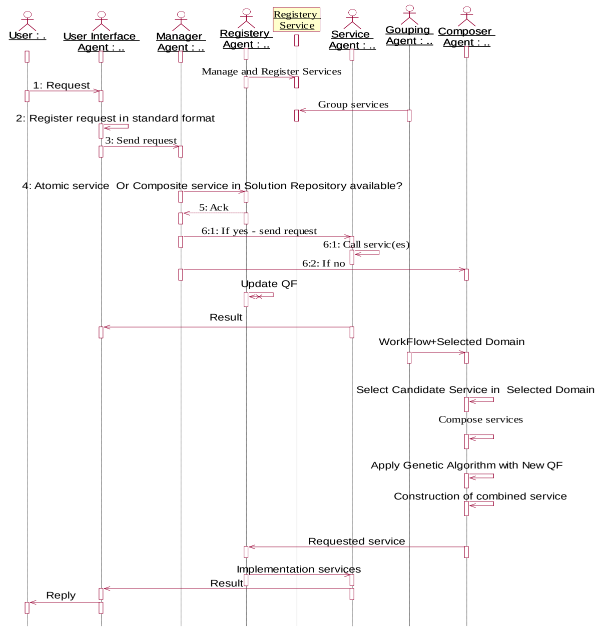
User : .
User Interface Agent : ..
Manager Agent : ..
Registery Agent : ..
Registery Service
Service Agent : ..
Gouping Agent : ..
Composer Agent : ..
Manage and Register Services
1: Request
Group services
2: Register request in standard format
3: Send request
4: Atomic service Or Composite service in Solution Repository available?
5: Ack
6:1: If yes - send request
6:1: Call servic(es)
6:2: If no
Update QF
Result
WorkFlow+Selected Domain
Select Candidate Service in Selected Domain
Compose services
Apply Genetic Algorithm with New QF
Construction of combined service
Requested service
Implementation services
Result
Reply

Figure 8: State diagram of the proposed model

IV. Evaluation

To evaluate the proposed model, we consider a system that gives reservations plane tickets and hotel for persons whose paper has been accepted at the conference. This system was implemented in a distributed manner that will choose the best suggested among the different services. For example, a person living in Iran and his article accepted in the United States is required to get the best suggested in terms of schedule and cost to participate at the conference.

The directory agents, after receiving information about each flight of the service provider airline, provide the required specifications and put in the service in the registration table.

With acceptance articles, the stimulus source starts and the user requests a stimulus by enter the system to receive a service that facilitates his attendance in a conference. User interface agent received this request from the application of conference and sends to a security agent. After reviewing the request and ensure the security and integrity of syntactic and lexical by security agent, it refers to the administrator agent. The Administrator agent searches between services in the directory table. Table 3 shows the existing data in the table directory.

Table 3
Available service at the service registration table.

Name of Service	Source	Destination	Cost	Flying duration (hours)	Stop duration to next flight
S1	Iran	Netherlands	300	4	1
S2	Iran	Germany	420	5.5	3
S3	Dubai	America	2850	20	2
S4	Netherlands	America	1260	14	1
S5	Germany	America	1100	12	3
S6	Iran	Dubai	130	2	2

The absence of desirable service means that the flight from the United States to Iran refers to the combination of the request agent. The combination agent search between the existing services declared the proposed options to the administrator agent. With the confirmation of the combined service, administrator agent registered the composite service in the directory and dictated the desired method directory agent. The directory agent was responsible for implementing this approach in one of existing services and selected one of them. At this stage, the genetic algorithm started and declared the best result to the agent administrator.

Manager agent then informed the user interface agent, and the possible results were declared to ensure the best flight for the user.

In the proposed framework, the chromosomes are formed simultaneously with the agent duties. The initial population of chromosomes is composed of 50 chromosomes and the initial values of required data were randomly initialized according to expert opinion. As mentioned, the efficiency is the required time to system accountability to certain number of events. This time is considered from the moment that the user log into the application and request the flight information to attend the conference to the time of issuance the flight ticket. The time value that lowers the efficiency of the system would be better.

At the beginning of the pilot phase, random values between zero and six hundred seconds to each service will be given. The number of runs represents the efficiency of the service; hence, the more runs indicate that the service has more capability to respond to user's needs.

Prior to using the system, value of service quality for all services is set at 1. In each time, when it is implemented unsuccessfully, the value is calculated according to the following equation:

$$\begin{gathered} Qf(t+1) = QF(t) - 0.001 * \alpha \\ 0 < QF \ll 10 \\ 0 \ll \alpha < 10 \end{gathered} \tag{3}$$

Directory agent is responsible for calculating and updating this value. If the value of *QF* is less than or equal to zero, the combined service will be deleted.

Table 4 represents the initial values that are randomly assigned. The results of applying the values of application after 1000 iterations are shown in Table 5.

The results in the proposed model show that the performance increased after the training phase. QF is the result of evaluating the proposed model at runtime. After every unsuccessful execution, its value is reduced according to the Equation (3). The values are shown in Table 2. In this case, the application performance, the system response time plus 0.01 multiplied by cost are considered. The ultimate goal of the system is to reduce the amount of performance for the consumer that increases user's satisfaction.

Table 4
Information of application used in the proposed model

Fitness function	Quality Factor	The number of run	Performance	Combined service name
0.0400	1	0	270	S1
0.0400	1	0	50	S2
0.0146	1	0	137	S3
0.0036	1	0	548	S4
0.0220	1	0	91	S5
0.0040	1	0	495	S6
0.0062	1	0	323	S7
0.0033	1	0	598	S8
0.0426	1	0	47	S9
0.0075	1	0	266	S10

When the number of running services is higher, it suggests that the service is more effective and win the best service in the selection process when it is executed. The selected service also has the lowest response time in the same services. The updates of these values by agents in the proposed model showed that intelligence has increased and the selection prevents poor services. Combined services were made in the experimental phase and the results show that it has a much better performance than the previous combined services.

Table 5
Information of application used in the proposed model after 1000 iteration with α =2

Fitness function	Quality Factor	The number of run	Performance	Combined service name
0.493	0.904	11	15	S19
0.744	0.992	14	20	S40
0.859	0.992	12	15	S31
0.99	0.99	18	19	S10
1.148	0.984	20	18	S12
1.344	0.96	41	30	S2
1.558	0.984	18	22	S42
1.66	0.996	9	6	S16
1.9023	0.98	32	17	S41
2.754	0.958	45	16	S7

V. CONCLUSIONS AND FUTURE WORK

Processes management of service-oriented architecture to invoke the service and use them in combination with the services is one of the important issues in service-oriented architecture. Much research has been done in this field.

In this paper, a method was proposed, that is using multi-agent systems, which can help to distribute resources and reduce tasks of the central unit in service-oriented architecture. It also helps to implement the functions of service-oriented architecture as distributed. The proposed model was evaluated based on scenario based compromise evaluation method. The main scenarios of this problem and the balance parameters were measured according to expert opinion. Finally, a selection of services was optimized by the genetic algorithm that results in the implementation of the application sample showed that there is an overall increased productivity of the system.

A practical usage of the proposed model in a practical environment was investigated and obtained results are reported. The results indicated that more time is needed to evaluate a system that runs in the international conference of computer society of Iran (ICCSI). Implementation of this model in real environment, and its comparison with its implementation on the prototype could be helpful to justify better efficiency and accuracy for future applications.

ACKNOWLEDGMENT

This article was kindly supported by Islamic Azad University, Science and research Branch, Tehran, Iran.

REFERENCES

[1] S. Pan and Q. Mao, "Case Study on Web Service Composition Based on Multi-Agent System," *Journal of Software*, vol. 8, pp. 900-907, 2013.
[2] D. I. Tapia, J. Bajo, and J. M. Corchado, "Distributing functionalities in a SOA-based multi-agent architecture," in *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)*, 2009, pp. 20-29.
[3] T. Erl, "Service-oriented architecture (SOA): concepts, technology, and design," 2005.

[4] J. H. Behnaz Nahvi "Adoption of Runtime Quality of Service in Genetic Algorithm on Memory-equipped Service-Oriented Architecture," *Advanced Science Letters*, 2016.

[5] P. Bartalos, "Effective automatic dynamic semantic web service composition," *Inf. Sci. and Technol. Bulletin ACM Slovakia,* vol. 3, pp. 61-72, 2011.

[6] M. Korotkiy and J. Top, "Blackboard-style Service Composition with Onto⇔ SOA," in *International Conference WWW/Internet, Portugal*, 2007.

[7] O. Hioual and Z. Boufaida, "An Agent Based Architecture (Usingplanning) For Dynamic And Semantic Web Services Composition In An Ebxml Context," *International Journal of Database Management Systems (IJDMS),* vol. 3, pp. 111-131, 2011.

[8] M. Alrifai, T. Risse, and W. Nejdl, "A hybrid approach for efficient Web service composition with end-to-end QoS constraints," *ACM Transactions on the Web (TWEB),* vol. 6, p. 7, 2012.

[9] E. Pejman, Y. Rastegari, P. M. Esfahani, and A. Salajegheh, "Web service composition methods: A survey," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2012.

[10] J. Hoffmann and I. Weber, "Web Service Composition," in *Encyclopedia of Social Network Analysis and Mining*, ed: Springer, 2014, pp. 2389-2399.

[11] D. Mallayya, B. Ramachandran, and S. Viswanathan, "An Automatic Web Service Composition Framework Using QoS-Based Web Service Ranking Algorithm," *The Scientific World Journal,* vol. 2015, 2015.

[12] A. Kazemi, A. Rostampour, P. Jamshidi, E. Nazemi, F. Shams, and A. N. Azizkandi, "A genetic algorithm based approach to service identification," in *Services (SERVICES), 2011 IEEE World Congress on*, 2011, pp. 339-346.

[13] A. V. Dastjerdi and R. Buyya, "A taxonomy of qos management and service selection methodologies for cloud computing," *Cloud Computing: Methodology, Systems, and Applications,* pp. 109-131, 2011.

[14] H. Jiang, X. Yang, K. Yin, S. Zhang, and J. A. Cristoforo, "Multi-path QoS-aware web service composition using variable length chromosome genetic algorithm," *Information Technology Journal,* vol. 10, pp. 113-119, 2011.

[15] G. Amirthayogam, M. Rathinraj, A. Gayathri, K. V. Reddy, T. Vijayalakshmi, M. Hemalatha, *et al.*, "Web service discovery with qos-an agent-based approach," *Int. J. Futuristic Sci. Eng. Technol*, vol. 1, pp. 1-5, 2013.

[16] S. Usmani, N. Azeem, and A. Samreen, "Dynamic service composition in SOA and QoS related issues," *International Journal of Computer Technology and Applications,* vol. 2, pp. 1315-1321, 2011.

[17] Z. Balfagih and M. F. B. Hassan, "Agent based monitoring framework for SOA applications quality," in *Information Technology (ITSim), 2010 International Symposium in*, 2010, pp. 1124-1129.

[18] T. Konnerth, "An Agent-Based Approach to Service-Oriented Architectures," Universitätsbibliothek der Technischen Universität Berlin, 2012.

[19] A. Butoi, G. A. Morar, and A. Ilea, "Agent-Based Framework for Implementing and Deploying of SOA," *Journal of Mobile, Embedded and Distributed Systems,* vol. 4, pp. 107-113, 2012.

[20] F. Brazier, V. Dignum, M. N. Huhns, C. Derksen, F. Dignum, T. Lessner, *et al.*, "Agent-based organisational governance of services," *Multiagent and Grid Systems,* vol. 8, pp. 3-18, 2012.

[21] S. Kumar, "Agent-Based Semantic Web Service Selection and Composition," in *Agent-Based Semantic Web Service Composition*, ed: Springer, 2012, pp. 15-25.

[22] P. Novak, P. Kadera, P. Vrba, and R. Sindelar, "Architecture of a multi-agent system for SCADA level in smart distributed environments," in *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, 2013, pp. 1-8.

[23] P. Papadopoulos, H. Tianfield, D. Moffat, and P. Barrie, "Decentralized multi-agent service composition," *Multiagent and Grid Systems*, vol. 9, pp. 45-100, 2013.

[24] J. A. García Coria, J. A. Castellanos-Garzón, and J. M. Corchado, "Intelligent business processes composition based on multi-agent systems," *Expert Systems with Applications,* vol. 41, pp. 1189-1205, 2014.

[25] A. Latrache, E. H. Nfaoui, and J. Boumhidi, "Multi agent based incident management system according to ITIL," in *Intelligent Systems and Computer Vision (ISCV), 2015*, 2015, pp. 1-7.

[26] L. Pang, R. Y. Zhong, and G. Q. Huang, "Agent-Based Service-Oriented Architecture for Heterogeneous Data Sources Management in Ubiquitous Enterprise," in *Advances in Sustainable and Competitive Manufacturing Systems*, ed: Springer, 2013, pp. 367-378.

[27] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Information Sciences*, vol. 280, pp. 218-238, 2014.

[28] Q.-m. YU, W. Lan, and D.-m. HUANG, "Fishery web service composition method based on ontology," *Journal of Integrative Agriculture,* vol. 11, pp. 792-799, 2012.

[29] X. Zhou, G. Xu, and L. Liu, "An approach for ontology construction based on relational database," *International Journal of Research and Reviews in Artificial Intelligence,* vol. 1, pp. 16-19, 2011.

[30] H. Tang, F. Zhong, and C. Yang, "A tree-based method of web service composition," in *Pervasive Computing and Applications, 2008. ICPCA 2008. Third International Conference on*, 2008, pp. 204-209.

[31] J. Zhou, D. Pakkala, J. Perala, E. Niemela, J. Riekki, and M. Ylianttila, "Dependency-aware service oriented architecture and service composition," in *Web Services, 2007. ICWS 2007. IEEE International Conference on*, 2007, pp. 1146-1149.

[32] A. M. Omer and A. Schill, "A Framework for Dependency Based Automatic Service Composition," in *Business Process Management Workshops*, 2009, pp. 535-541.

[33] E. Karakoc and P. Senkul, "Composing semantic Web services under constraints," *Expert Systems with Applications,* vol. 36, pp. 11021-11029, 2009.

[34] V. Suraci, T. Inzerilli, and S. Mignanti, "Design and Implementation of a Service Discovery Architecture in Pervasive Systems," *IST Mobile Wireless Summit, Dresden, Germany,* 2005.

[35] B. Benatallah, H. Reza, H. R. M. Nezhad, F. Casati, F. Toumani, and J. Ponge, "Service mosaic: A model-driven framework for web services life-cycle management," *Internet Computing, IEEE,* vol. 10, pp. 55-63, 2006.

[36] J. Liu, N. Gu, Y. Zong, Z. Ding, S. Zhang, and Q. Zhang, "Service registration and discovery in a domain-oriented UDDI registry," in *Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on*, 2005, pp. 276-283.

[37] H. Samset and R. Bræk, "Dynamic service discovery using active lookup and registration," in *Services-Part I, 2008. IEEE Congress on*, 2008, pp. 545-552.

[38] C. I. Pinzón, J. F. De Paz, D. I. Tapia, J. Bajo, and J. M. Corchado, "Improving the security level of the FUSION@ multi-agent architecture," *Expert Systems with Applications,* vol. 39, pp. 7536-7545, 2012.

[39] E. Argente, V. Botti, C. Carrascosa, A. Giret, V. Julian, and M. Rebollo, "An abstract architecture for virtual organizations: The THOMAS approach," *Knowledge and Information Systems,* vol. 29, pp. 379-403, 2011.

[40] R. Kazman, P. Clements, and L. Bass, "Software architecture in Practice," *Addison Wesley Abril,* vol. 11, 2003.