Improving Service-Oriented Architecture Processes in Process of Automatic Services Composition Using Memory and QF, QWV Factor

Behnaz Nahvi
Department of Computer Engineering
Science and Research Branch
Islamic Azad University
Tehran, Iran

Jafar Habibi
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran

***Abstract*—The application of service-orientated architecture in organizations for implementation of complicated workflows in electronic way using composite web services has become widespread. Thus, challenging research issues have also been raised in this regard. One of these issues is constructing composite web services by workflows. These workflows are composed of existing web services. Selections of a web service for each of workflow activities and fulfilling users' conditions is still regarded as a major challenge. In fact, selection of a web service out of many such web services with identical function is a critical task which generally depends on composite evaluation tool of QoS. Previously proposed approaches do not consider exchange restrictions on the composition process and internal processes of architecture and previous experiences, and they ignore the fact that value of many of QoSs depends on the time of implementation. Selection of web services only based on QoS does not bring about optimal composite web service. Thus, till now, no solution has been proposed that performs composition process automatically or semi-automatically in optimal manner**

Objective: identification of existing concerns on composition of services and then designing a framework to provide a solution which consider all concerns and finally performing tests in order to examine and evaluate proposed framework

Method: in the proposed framework, elements affecting management of service-oriented architecture processes are organized according to a logical procedure. This framework identifies processes of this style of architecture based on requirements in service-oriented architecture processes management and according to qualitative features in this area. In the proposed framework, in addition to using existing data in the problem area, existing structure and patterns in the area of software architecture are also utilize, and management processes in service-orientated architecture are improved based on propriety of available requirements. QWV are qualitative weighted dynamic features which indicate priority of users, and QF is quality factor of service at the time of implementation which is weighted in the framework. These factors are used for constructing composite web service. Multifactor computing is known as a natural computational system for automating the interaction between services. The factors in multi-agent systems can be used as the main reliable mechanism for the control which usually use data exchange for accelerating their evaluations. For identification of all concerns in the solution space, many aspects should be examined. To this end, classes of agents are defined which investigate these aspects in the form of four components using repository data.

Results: proposed framework was simulated by Arena software and results showed this framework can be useful in automatic generation of needed services and meet all concerns at the same time. Results support that using agents in the model increased speed of accountability and satisfaction of users as well as system efficiency.

***Keywords*—Service-orien ted architecture; process management; multi-agent systems**

I. Introduction

Service-orientated architecture provides a collection of designing principles for operationalization and implementation of automatic commercial process in heterogeneous environments[1]. Service-oriented architecture includes management of applications, services, processes, firmware, infrastructures and software tools in line with business objectives. Service-oriented architecture process management is classified into service-oriented architecture lifecycle management, service management, service change management, service composition management and service interaction management, service registries and exploration management. Service-oriented architecture management is actually a multi-faceted task which covers IT management and Business Service Management. There are several ways for implementing SOA, but the most famous and most popular way are run by Web Service technology[2] that depends on infrastructure of World Wide Web and use of open XML standards such as SOAP, WSDL. Another benefit of using web services is their interoperability between different heterogeneous organizations or units [3].

Over recent years, the number of web services with identical performance and different quality has increased and is still growing [4] which leads to increased complexity of composition[5]. Cost and efforts for constructing composite services in manual way is certainly higher than composition cost as automatically[6]. The other reason which increases cost of manual effort compared to automatic efforts is that usually demands are issued continuously or previous demands are changed. Thus, there is need for techniques which automatically compose services and reduce cost and effort to respond to needs of user[7]. Composition of services based on QoS is one of the issues raised in service-oriented architecture.

Most of the available methods consider QoS as static. It means that fixed values are considered for QoS values in all user demands over the time. While value of these features depends on the time and have different values depending idea of the user on different applications.

Generally, the lifecycle of a composite web service includes three stages: 1. Design, 2. Implementation and supervision, 3. Re-engineering [8, 9]. In the first stage, obvious features of composite web service are identified. In the second stage, at the time of design and supervision, addressing and implementation of composite web services and finding solutions and errors which emerge in this stage is done. Finally, in re-engineering stage, features of composite services are modified which depends on the data obtained from second stage. The solutions for performing this process are also provided as dynamic (like[10]) which are capable of review in composite web service immediately after second stage (dynamic binding). In this paper, a model is proposed which considers three stages of lifecycle of composite service construction. First stage is described in more details in this paper, and subsequent stages are extended in the future works.

In the proposed model, three stages are organized by agents in four components and depository. Ten classes of agents are embedded in components and perform their tasks using depositories. In the stage of design, user interface agent provides the demand for application agent following taking user demands and transforming them to standard format. The application agent turns user's need to tasks and workflow of composite web service is generated. Second stage, that is, discovery of suitable candidate services for each activity, is done by registry factor. For accelerating this operation in the proposed model, services are grouped in sub-domains considering their functions by grouping agent in design stage. Registry agent searches for relative service for each task in sub-domains. If corresponding service is not found, it searches for previous composite services in the system memory, i.e. solution depository. In case of not finding suitable service, demand is referred to the composer agent service composer should generate workflow manually or semi-automatically. This workflow is logically as an ordered set of activities that each activity is performed by an atomic service. Following selection of suitable candidate services for taking part in composition operation, third stage is performed. That is, selection of the best service out of candidate services for taking part in composition. Registry agent does selection under supervision of manager agent using genetic algorithm considering dynamic values of QoS and QF. QoSs are regularly updated by evaluator agent. QWV is qualitative features vector considered by the user, which indicates significance of each feature in the view of user. Quality of service in web services (QoS) includes some non-operating characteristics such as cost performance, runtime, the availability, performance and security success rate[11]. QWV is weighted value of QoS which is dynamic and is weighted over the time according to priorities of user by operating user interface and provides optimization. Value related to service quality factor (QF) denotes quality of service implementation and availability of service. It is stored in model memory and updated by registry agent in Meta data of system. Following selection of suitable service out of candidate services, last stage of design is finished. Second phase of lifecycle, i.e. implementation and supervision, is followed by service agent, management agent, and security agent, and processes related to third cycler of composite web service lifecycle is performed in SME component. In our model, agents are put in components and they perform respective operations using depositories which are the same as model memory. The purpose is generating composite service which has non-functional features optimal for user. As explained in previous paragraph, designing composite service is a time-consuming and complex process, and if it can be accomplished automatically, certainly less costs and efforts are needed compared to manual manner[6].

Service-oriented architecture lifecycle management, service management, service changes management, service composition management, service interaction management, and service registries and exploration management is done by agents in the model. Multi-agent system performs a combination of managerial affairs in service-oriented architecture. Data needed by the system are extracted by agents and embedded in the Meta data depository. These data are updated and managed in registry service table, and they would be used at the time of implementation.

Overall goal of previous works is finding administrative workflow which can generate services in a composite web service to reach requested functionality. The other point is the way of selecting services out of candidate services which has been done for predefined workflow to reach non-functionality tasks as QoS. In the previous works, agents have not be used comprehensive in the whole architecture processes [12, 13]. They used multi-agent system as intelligent control layer for managing affairs. Establishment of interaction among services [14] and manipulating handling changes [15] are among tasks which led to emergence of agents in architecture composition.

In summary, the proposed framework deals with classification of service-oriented architecture processes. In service composition part, using quality factor definition it was practically shown that agents can make service-oriented architecture activities considerably more effective and efficient by applying management and control over model components and storage and updating model knowledge in depositories. Eeach part in the future model will be provided for optimization of these processes in model components and depositories. Works done for this paper are explained in detail in the following:

1) Firstly efforts for automatic composition of services were reviewed and analysed and existing concerns and gaps were identified.

2) Existing processes in service-oriented architecture were identified and they were structured in the proposed model.

3) Agents were used for improving processes in the architecture.

4) Internal memory was added to the system by adding depositories to the framework.

Priorities of this framework over previous methods include as follows:

1) Quality factor (QF) indicates framework's evaluation of provided services which is more reliable criterion for selection of optimal services beside QoS which is provided by service generator dynamically known as QWV.

2) Weighing quality factor (QF) by agents in the framework allows evaluation of services in implementation stage.

3) QWV is weighed by user interface factor and includes user priorities using updated values of QoS, and this value is used by composer agent for constructing composite service.

4) Composer agent identifies the best existing compositions using compliance function, in which use QoS, user priorities, and framework evaluation of the service is also considered. It is performed by genetic algorithm. The best composition is referred to the registry agent. Using QF and QWV factors makes genetic algorithm more effective compared to similar works in selection of the best composite services.

5) For filling the existing gaps in composition issue, this paper provides a framework which considers all issues in architecture. Using agents beside depositories reduces necessity for relationship between framework and user and it allows automatic generation of composite web service to meet user needs and covers all existing concerns in composition issues.

It is known that composition is a multipurpose problem. Thus, in the proposed model, it is attempted to consider several problems simultaneously in the composition. A new approach is provided for multipurpose problem and currently a comprehensive introduction and experimental approach for it is discussed. In automatic multipurpose composition there are some concerns. Proposed framework considers composition with all aspects. For composition, a fitness function is used and QF and QWV are embedded in it. Agents are used for implementing GA in order to update variables. Features of the proposed model is different from most of traditional methods and can be used for other multipurpose problems with similar features. As in the proposed framework, specific philosophy and view is used in classification of architecture management processes, needed issues are described in explanation of the model.

In the following, 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. Standard of service definitions in the framework and ranking services are also described. In the fourth section, for evaluation of the framework, curriculum of second semester 2015 – 2016 in Islamic Azad University of Karaj is implemented in the framework and the last section is devoted to conclusions and future work.

II. Review Of Literature

In this section, previous studies and works are reviewed. Many works have been conducted on service changes management area which [16] can be an example. In[3] using intelligent solution for finding optimal solution is suggested. In Table 1 part of works on service composition area is shown. Service composition issue is divided into two main fields including service selection and service composition[17]. In Figure 1, service selection methods are classified according to[18]. In composition of services, each selected service has specific administrative capability and it is unique. These services should be implemented in the form of a composite service and provide needed capabilities. Workflow implicitly makes this composite service. For composition of works, both semantic links and key terms comparison can be used. Semantic links will be used in the future works.

TABLE I. A Comparison Table for Current Works

Author	approach	QoS_Aware	Semantic Based	Co-agent	Using optimization algorithms
Mohammed, et al. [19]	Heuristic algorithm	Yes	No	No	Yes
. Wang, et al. [20]	ant colony optimization algorithm	Yes	No	No	Yes
X. Wang, et al. [21]	bee colony approach	Yes	No	No	Yes
T. Zhang [22]	particle swarm optimization algorithm	Yes	No	No	Yes
V. S. Jeure et al. [23]	particle swarm optimization algorithm	Yes	No	No	Yes
X. Zhao, et al. [24]	Immune algorithm	Yes	No	No	Yes
M. Fahad, et al.[25]	Semantic-based BusinessProcess Execution Engine	Yes	Yes	No	No

III. Concept of Proposed Framework

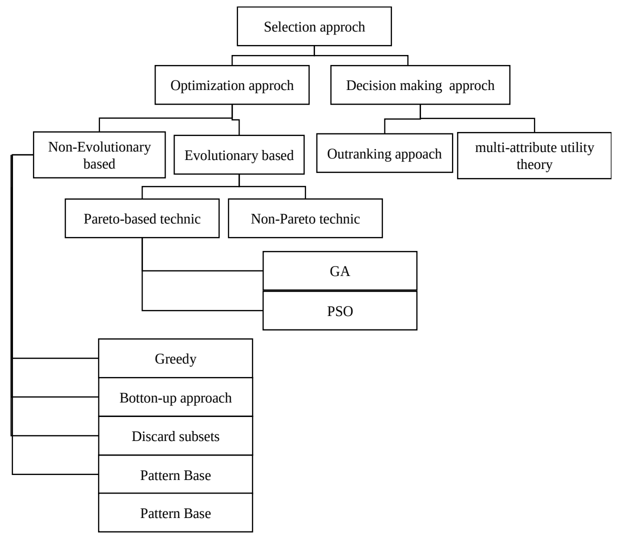

Fig. 1. Web service selection approaches [18]

In this section, assumptions, symbols and icons used in the model are described. The proposed model is composed of components and depositories. Each component is a computing entity which is an interconnected collection of the system functionality requirements. Components communicate through one or more ports to the environment. Ten categories of agents help components so that system requirements are removed. The overall structure of the proposed framework has been shown in Figure 2. Business Process Management (BPM) is take from stakeholders by user interface agent and it is stored in this component under Request For Change (RFC) standard form. Requirements in this framework are stated under title of RFC so that it can be integrated to ITIL framework in the future. Problem definition in the proposed framework is defined by RFC which is an ordered triple: RFC (T, QWV, C)

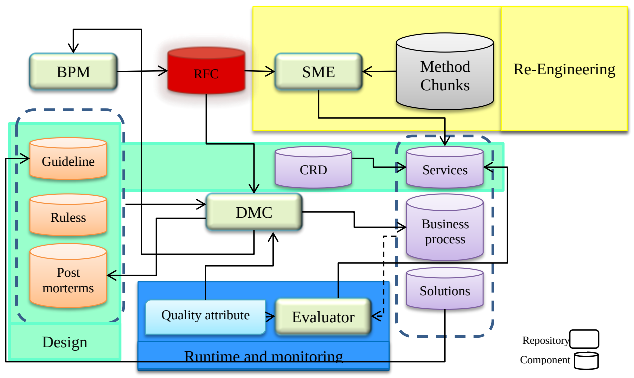

Fig. 2. The structure of the proposed framework

- T specifies a set of works composed of $\sum_{i=1}^{n}$ task ti where n denotes total number of works. Application agent is responsible for developing workflow in BPM component. Service workflows are implemented by modeling languages (BPEL)[19]. Application agent produces and manages problem statement, that is, RFC (Figure 3).

- QWV is weighted vector of QoS which specifies user priorities on qualitative features of service (QoS) that is composed of: QWV = <qw_1, qw_2, qw_3, …, qw_m>

m denotes number of qualitative features which are important for system in the framework and are obtained by manager agent at the beginning of the work based on the results taken by application agent. User priorities are reflected in this weighted vector.

- Constraints (C) determine constraints specified by the user.

- In terms of emission of changes in service depository in the framework, our design is taken from active depository style. Some services have been specified in depository which are informed of specific events shared in the depository. These services inform changes in service depository to DMC component.

Situational Method Engineering (SME) component reengineers in lifecycle of composite web service using method chunks. Decision-Making Center (DMC) component has axial stats in proposed framework. In this component, different concepts are put together and general goals of framework are realized. In fact, it can be stated DMC component has decision making status in the framework.

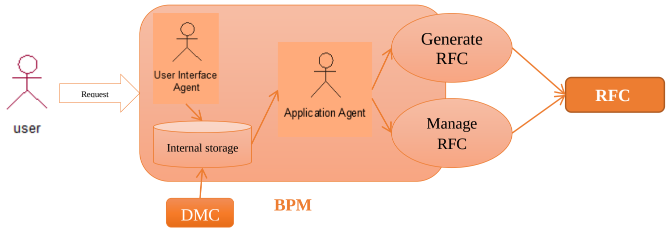

Fig. 3. BPM components structure

Manager agent contains a series of principles, rules, guidelines, tools and previous experiences (located in depositories) which decision making is done using them. Overall this component is shown in Fig 4. Manager agent in this component makes decision considering priorities of QWV and QoS stakeholders for retrieval of agents[26]. Solution repository component plays key role in realization of approach based on the pattern adopted in framework. Significant point in proposed framework is that it has learning capability and solution repository is raised as one of the memories of this intelligent framework. In other words, this component of architecture can be extended by new solutions. In addition, possibility for correction or changing existing solutions is also available. In relation with the way of storing these solutions, registry agent with the help of grouping agent, models solutions in the form of modeling language BPMN. BPMN can be transformed into BPEL standard. Thus, maximum compatibility and standardization in the framework is prepared and it allows that business processes are easily developed with extending solutions.

In the proposed framework, each service is connected to a standardizer known as common rules database (CRD). This unit is a basis of existing rules and norms, related to application case, which is embedded in the proposed framework in order to chat and negotiate with CRD connected to different services. Proposed framework is shaped based on reference layering of service-oriented architecture[27, 28]. Multi-agent system is used for improving service-oriented architecture processes (Figure 5). In evaluator component, registry agent scores services, business processes and solutions considering their qualitative features, and turns QoS value from static to dynamic state. Here two working areas are distinguished. First area supervises measurement of qualitative features. Second area scores these cases given their qualitative features. In fact, in the first area, measurement and identification of qualitative features is dealt and on the other hand, multivariate decision making is faced in the second area. In relation with measurement of qualitative features, expertness is the most important point (Figure 6). In other words, considering complexity and qualitative of these criteria, human agent or expertness role is bolded. The methods which have addressed this issue have used scenario-based techniques and in addition they have also utilized controlling and screening tools. In the future works, a fuzzy system will be used in this section for measuring qualitative features.

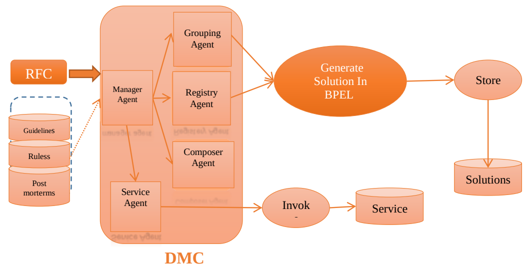

Fig. 4. DMC components structure

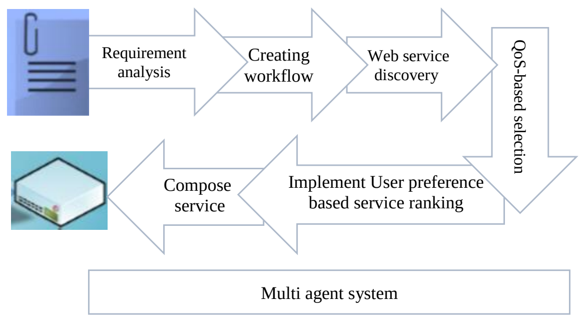

Fig. 5. DMC components structure

In relation with scoring services, business processes and solutions, there are different approaches, which somehow are based on scoring by experts In the proposed framework, scoring is delegated to registry agent, which performs scoring by combination of results obtained from solutions depository and QF and results taken from BPM component. The value related to QF, QWV is also updated in this component.

A. *Defining Service in Framework*

In order to assure comprehensibility of service by the machine and automatic relationship without human intervention, each service is defined in the framework by tuples.

Tuple :(ID,I,O,P,E,NF)

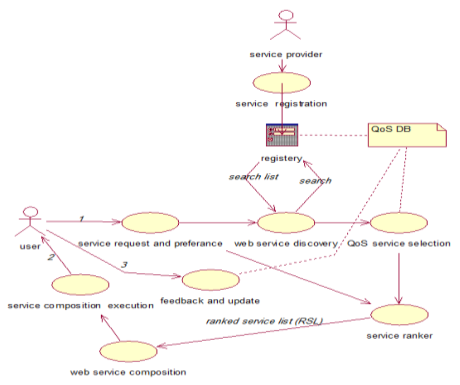

Fig. 6. Scoring the values of QoS, by experts

Values of this senary tuple are described in the following:

ID: its value is unique and specifies identity and name of service.

I: it specifies service input. It determines requirements needed by the service at the retrieval time for successful invoking the service.

O: It specifies service output when the service is invoked successfully.

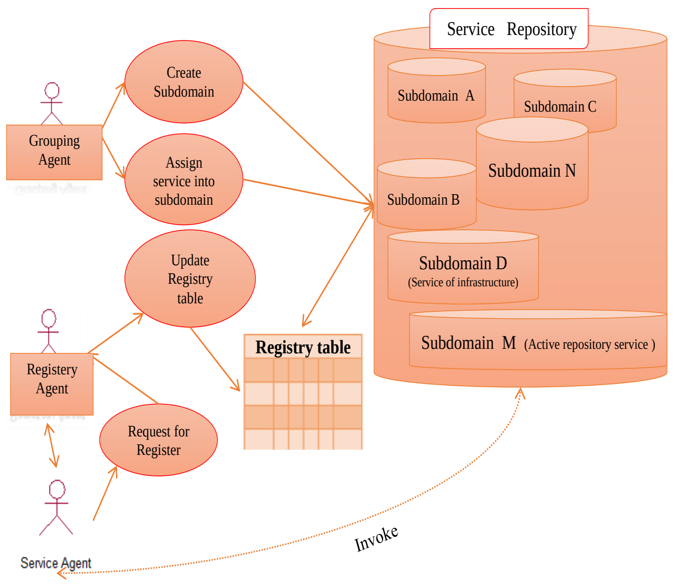

Fig. 7. Repository Service in the proposed framework

P: pre- requirements; a collection of conditions which should be prepared prior to service retrieval so that service is invoked successfully.

E: Result; a set of conditions that must exist following successful invoking the service.

Data conversion services which are responsible for the task of converting between different technologies are among these services which has been shown in Fig 7.

NF: Specifies non-functional features of service which include QoS and QF, and compliance function will be defined for it considering each problem.

B. *Ranking Services*

The core of automatic composition of services is based on algorithm for ranking QoS of services according to user demand and service composition algorithm (Algorithm 1). Values related to QoS of each service is weighted at the time of service recording by service generator in registry table. Agents in the framework are responsible for supervision over these features and if necessary, these values are updated by service agent. QWV value is recorded in features depository following imposing weights by application interface.

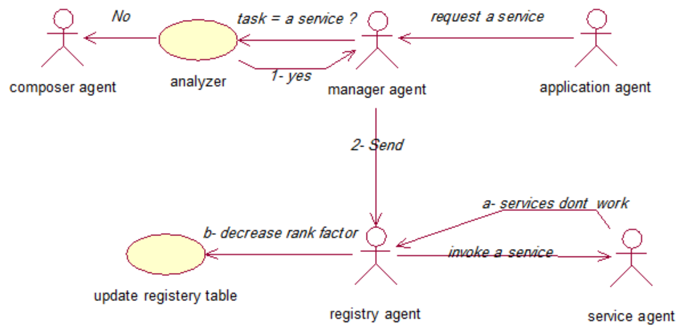

Fig. 8. Request a service use case diagram

Algorithm 1 : User Preference Based web service ranking algorithm (RFC (T,QWV,C), WS)

```
// RFC: User Request
// $S_i$: web service
// SDSi: sub domain of Si
// $t_i$: Task involved in user request
// $WSL_i$: list of web services in the registry table
// $SL_i$: search list
// $FL_i$: filtered list
// QFRL: QF ranked List
// QWV: QoS weighted vector
// ⟨QWV⟩=⟨$qw_1$, $qw_2$⟩
// qw 1: Cost Weight
// qw 2: Response Time Weight
// CSLi: composed Services List ( Register  in solution repository)
Begin
(1)For each task $t_i$ in RFC
    (2)Discover ($WSL_i$, $SDS_i$)
        (3)For each $S_i$ in $SL_i$
        Do
        (4)If ( Availability == true) & (QF>0)
            (5)SL.add()
        (6)Endif
        (7)EndFor
    (8)QoS based Service Selection (SL)
    (9)Compute QoS Rank(FL)
    (10)Final Rank based Sorting (QFRL,QWV)
    (11) Apply Genetic Algorithms
(12)End For
(12)Return CSL
End
```

QF value is weighted by the framework. Initial weighing for this factor is set as 1 in all services. When the service cannot work, fails, or it cannot be found, value of this factor is reduced according to Formula (1), or when performance of it is reduced compared to previous runs, this value is decreased. As shown in Fig 8, registry agent is responsible for updating this value in each service. QF is increased when the assigned works are properly done, and it becomes a basis for service selection. α is wegithed by the framework considering the problem($0 < QF \leq 1$) , ($-10 \leq \alpha < 10$) . If non implementation of the service has considerable impact on the system, this value is considered as large, otherwise, smaller values are considered for it.

$$QF(t+1) = QF(t) - 0.01 * \alpha \qquad (1)$$

I, P value is weighted in BPM component by user interface agent given features of the service demanded by the user, and it is stored in RFC depository. DMC component is responsible

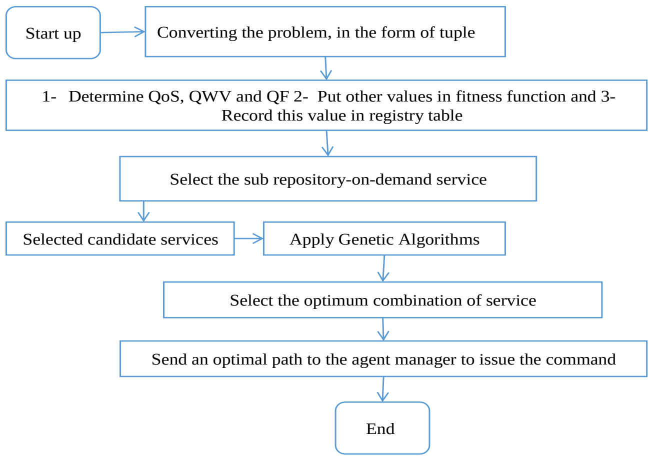
Start up
Converting the problem, in the form of tuple
1- Determine QoS, QWV and QF 2- Put other values in fitness function and 3- Record this value in registry table
Select the sub repository-on-demand service
Selected candidate services
Apply Genetic Algorithms
Select the optimum combination of service
Send an optimal path to the agent manager to issue the command
End

Fig. 9. Flowchart of combined service in the proposed framework

for making decision on continuing the work. Process of the works is shown in summary in

ure 9.

A composite web service (CWS) is a set of services, where features of each service are specified by senary tuple. Generator of composite service should assign a tuple to this set which contains features of composite service.

In tuples I, O, P, E service performance is shown. Using tree structure described in[29], CWS performance in its corresponding tuple is shown. Non-functional features of this composite service are weighted in NFi which includes two parts; QF which takes a default value and QoS, composite web service, which is calculated by composer agent. Composition of web services is done through four ways[30] which is given in Figure 10. In composite structure (sequential (a), cycle (b), parallel (c), and branch structure (d)) works are run sequentially and, each work is run several times in loop structure. In parallel structure, all work can be done at a time and after completing all tasks, parallel structure of the next work can be started. In branch structure, if at least one of the things is done, the next work outside of the structure can be done. Given that the composite web service is composed of these four structures, QoS of composite web service can be obtained using calculations corresponding to the structure by the formula (2) [24]. Web service composition algorithm based on QoS shown in Algorithm 2.

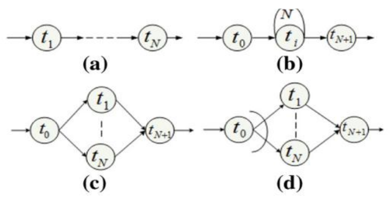

Fig. 10. basic composite models of composite service

Algorithm 2: QoS aware web service composition algorithm (UR(,QWV,),RSLi)

```
Begin

(1)Rank Services
        (1.1) Compute  QWV for each task $t_i$ in T
        (1.2)Save the $RSL_i$ For each task  $t_i$ in T
(2)Store each $RSL_i$ in task tables
(3)Compute Service Composition (SC) table
        (3.1)Generate all possible Composition plans by
        taking Cartesian product of all the Task tables
        obtained in Step (2)
        (3.2)Save the Composition plans (CP) in Service
Composition Table
(4)Calculate QoS Aggregated value for each CP in Service
Composition and save in
Composition Plan List (CPL)
(5)Constraint Analyzer
        (5.1)Perform Constraint Analyzer (SC, C) for each
CP in CPL
        (5.2)Save composite services that satisfy constraints
        in Filtered Composition Plan  List (FCP)
(6)Pareto Optimal based Selection
        (6.1)Perform Pareto Selection (FCP)
        (6.2)Save Composition Plans at eriltering in Pareto
        Optimal based Selected List (POSL)
(7)Compute Aggregated QoS Rank for each CP in POSL
        (7.1)Evaluate all the  Rank for each CP in POSL
        (7.2)Save the CP with  Rank in POSL
(8)Calculate Final rank (POSL,QWV)
        (8.1)Compute Final rank for all CP in POSL
        (8.2)Sort and save the Composition Plan in Ranked
        Composition Plan List (RCPL) Based on Final Rank
(9)Execute all the Composition Plan in RCPL
(10)Get feedback and up date Rep()
End
```

$$Time = \begin{cases} \sum_{i=1}^{N} T_i\,(a) \\ \sum_{i=1}^{N} T_i\,(b) \\ \max T_i\,(c) \\ \min T_i\,(d) \end{cases} \quad Cost = \begin{cases} \sum_{i=1}^{N} C_i\,(a) \\ \sum_{i=1}^{N} C_i\,(b) \\ \sum_{i=1}^{N} C_i\,(c) \\ \min C_i\,(d) \end{cases} \qquad (2)$$

IV. Application Case: Curriculum

A curriculum scenario is a sample of the system composed of service combination. For implementing curriculum in a semester, selection of instructors, their working days, selection of needed courses, and class selection are among services, combination of which is one of the issues which yet there is no automatic comprehensive solution that QoS features are applied in service selection. User demand includes a set of

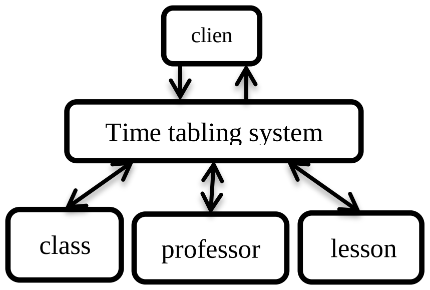

Fig. 11. time tabling scenario

works like class reservation, instructor reservation, and course selection. Atomic services (instructor, class, and course) should be selected in such a way that the best combination is obtained. Users can enter type of service and constraints and specific conditions and local constrains like QoS (QWV) setting in the form of demand as RFC standard. QoS weights are used in ranking services and play effective role in the process of constructing optimal composite web service in fitness function for genetic algorithm process. When user demand is entered into the system (RFC), firstly sub-domain of services is selected and then respective candidate services are chosen for doing works. These sub-domains are done by grouping agent considering expertise of instructors and their field and faculties, location of classes, and courses for the semester.

For combination generation, raking services is done using values of QWV and QF. User idea of QWV and system evaluation of QF services play key role in selection of optimal combination. Following ranking, genetic algorithm selects the best existing combination with imposing fitness function (formula 2), and manager agent and service agent and compositor agent are responsible for running this combination. Following retrieval of respective services and running process, QF values are updated by registry agent, and this memory will be contribute in subsequent runs in ranking services. Recording experience of each run increases system efficiency in subsequent runs.

$$fitness\ Function = \frac{\sum_{i=1}^{k} QF_i * QWVi}{k} \qquad (2)$$

User demand is firstly turned to triple by application interface agent. In this ordered triple (T, QWV, C) RFC, works include selection of respective course in specified time with selection of instructor and class. In QWV, weigh vector of qualitative features of user priorities are specified in multi-criteria decision, which is implemented by optimization, and conditions and constraints of user are stated in C form, which can choose out of 10 available conditions. Services in each university are prepared in a table in Excel environment along with qualitative features as manual by the education responsible person, part of which is shown in Table 2. Information related to QoS is constantly evaluated and updated by evaluator which is education CEO, and they are accessible in features depository. Combination program is updated in solution depository at every run periodically, and it is removed if necessary. Service providers register services available in service registry and their qualitative features are updated dynamically in the framework.

Framework evaluation environment is composed of three virtual machines. Service exploration and service ranking as well as construction of composite web services is done in a system with following characteristics: 3 GB RAM, Windows XP3, 2.13 GHz, CPU 3 Intel Core i. Clients, which are made by application interface in technical, literature and veterinary faculties and are responsible for service registry construction, have following characteristics: RAM 2 GB and Windows 7 ,2GHz, Duo CPU, 2Intel Core. Information related service providers and information related to service quality and actually service quality depository and solution depository and service depository are embedded in virtual machine with following characteristics: Windows XP, 3GBRAM and 3GHz, 2.13 CPU, Intel Core i.

In optimization and optimal composite web service construction part, MATLAB programming language is used for applying genetic algorithm. For each service in this application case, compliance between instructor expertise and respective course and the costs parameters are also especially considered in addition to other qualitative features of services, which are used in calculation of final compliance function. One of the parameters considered in this case, is the time needed for performing curriculum by system.

TABLE II. EXAMPLES OF SERVICES AVAILABLE INCLUDING CLASSES, LESSONS AND MASTER

Amount hour/	Specialty Code	master code	master	row
25000	12	91253	Dr.Khalilian	1
32000	8	45871	Dr.Nikravan	2
28000	11	52489	Dr.Pishvayi	3
15000	9	54219	Dr.shemirani	4
30000	11	36542	Dr.salajeghe	5

hour/ Amount	Class location	Class code	row
5000	2	154	1
4000	3	147	2
5000	2	123	3
3000	1	187	4
6000	3	132	5
2500	1	23	6
5500	5	352	7

Specialty Code	Amount hour/	Lesson code	row
12	38000	302	1
8 , 11	58000	305	2
8	45000	132	3
9	87000	845	4
11	25000	251	5
8	30000	265	6
12	4000	245	7

TABLE III. GENETIC ALGORITHM PARAMETERS

number of population	max of iteration	percent of crossover	percent of mutation
100	1000	0.01	0.99

Genetic algorithm parameters are set according to following table and obtained results are used at every run in simulation environment.

Finally, the proposed framework is simulated using Arena software, Version 13.50 on Islamic Azad University system, Karaj Branch with following characteristics: 3GB RAM 2.13 GHz (CPU Intel Corei3). Overall form of simulated framework is shown in Fig 12. This simulation is evaluated in five steps. In the first step, framework responds to 27 demands of users. In the next step, 45 user demands and after in last step 60, 75, 90 user demands are evaluate. Evaluation showed that with increasing user demands and updating internal system values, framework memory and optimizations by genetic algorithm considerably help reducing system response time and costs including computational costs and service use cost.

V. CONCLUSION

Proposed framework is provided with considering challenges existing in management of service-oriented architecture processes. Service-oriented architecture processes management is classified into service-oriented architecture lifecycle management, service management, service change management, service composition management and service interaction management, service records and exploration management. In this framework, service-oriented architecture processes are regularly structured and management of each part is done considering its performance.

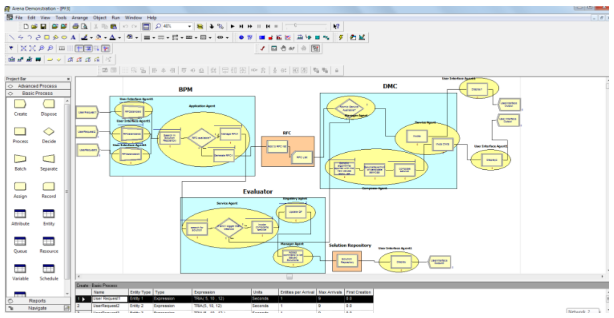

Fig. 12. Simulated framework in arena

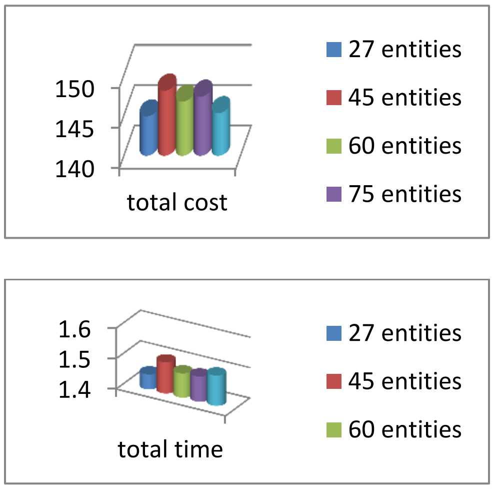

Fig. 13. Two-parameter simulation results show that with increasing time and cost of implementation of the framework increases system performance

As mentioned, focused processes have problems in scaling and run and strength. In focused processing, the processor should have the whole problem knowledge and can manage integrated knowledge and utilize it. Management and processing such knowledge needs high computational power and it is beyond capacity of a single focused system.

Thus, in this framework, multi-agent system is used for distributing the processing to improve service-oriented architecture processes. Depositories of this framework as its internal memory increase system computational power in consecutive runs and improves responding speed. For evaluating proposed framework, service combination is used. Curriculum system of Islamic Azad University, Karaj Branch, was used as application case by the proposed framework and it was tested. Results show that using this framework increase system speed and efficiency after pilot period in stabilization period. The more user demands are met by more works and more services are needed in combination, efficiency of proposed framework is shown more. Cost and time of responding and system availability and efficiency are among parameters which were evaluated. In future works, it is attempted to use BPM component in fuzzy system and then framework will be evaluated by standard databases like WS-Challenge database. Using semantic will be also among future works.

REFERENCES

[1] T. Erl, "Service-oriented architecture (SOA): concepts, technology, and design," 2005.

[2] J. El Hadad, M. Manouvrier, and M. Rukoz, "TQoS: Transactional and QoS-aware selection algorithm for automatic Web service composition," Services Computing, IEEE Transactions on, vol. 3, pp. 73-85, 2010.

[3] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, "Heuristics for qos-aware web service composition," in Web Services, 2006. ICWS'06. International Conference on, 2006, pp. 72-82.

[4] E. Goncalves da Silva, L. Ferreira Pires, and M. van Sinderen, "Supporting dynamic service composition at runtime based on end-user requirements," 2009.

[5] M. Rambold, H. Kasinger, F. Lautenbacher, and B. Bauer, "Towards autonomic service discovery a survey and comparison," in Services Computing, 2009. SCC'09. IEEE International Conference on, 2009, pp. 192-201.

[6] Z. Li and L. O'Brien, "Towards effort estimation for web service compositions using classification matrix," International Journal on Advances in Internet Technology Volume 3, Number 3 & 4, 2010, 2010.

[7] 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.

[8] W. Gaaloul, S. Bhiri, and M. Rouached, "Event-based design and runtime verification of composite service transactional behavior," Services Computing, IEEE Transactions on, vol. 3, pp. 32-45, 2010.

[9] F. Gao, E. Curry, M. I. Ali, S. Bhiri, and A. Mileo, "Qos-aware complex event service composition and optimization using genetic algorithms," in Service-Oriented Computing, ed: Springer, 2014, pp. 386-393.

[10] V. Cardellini, E. Casalicchio, V. Grassi, F. Lo Presti, and R. Mirandola, "Qos-driven runtime adaptation of service oriented architectures," in Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, 2009, pp. 131-140.

[11] C. Jatoth, G. Gangadharan, and R. Buyya, "Computational Intelligence based QoS-aware Web Service Composition: A Systematic Literature Review."

[12] M. Vallée, F. Ramparany, and L. Vercouter, "A multi-agent system for dynamic service composition in ambient intelligence environments," in The 3rd International Conference on Pervasive Computing (PERVASIVE 2005), 2005, pp. 165-171.

[13] Z. Huang, J. Zhang, and Q. Cong, "Agent-Based Service-Oriented Dynamic Integration ERP Architecture," in The Second International Symposium on Networking and Network Security (ISNNS 2010), 2010, p. 136.

[14] L. G. Nardin, A. A. Brandão, and J. S. Sichman, "Experiments on semantic interoperability of agent reputation models using the SOARI architecture," Engineering Applications of Artificial Intelligence, vol. 24, pp. 1461-1471, 2011.

[15] J. Li, D. Ma, L. Li, and H. Zhu, "AADSS: Agent-based adaptive dynamic semantic web service selection," in Next Generation Web Services Practices, 2008. NWESP'08. 4th International Conference on, 2008, pp. 83-89.

[16] H. K. Dam and A. Ghose, "Supporting change propagation in the maintenance and evolution of service-oriented architectures," in Software Engineering Conference (APSEC), 2010 17th Asia Pacific, 2010, pp. 156-165.

[17] Y.-Y. Fanjiang, Y. Syu, C.-H. Wu, J.-Y. Kuo, and S.-P. Ma, "Genetic algorithm for QoS-aware dynamic web services composition," in Machine Learning and Cybernetics (ICMLC), 2010 International Conference on, 2010, pp. 3246-3251.

[18] 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.

[19] M. Mohammed, M. A. Chikh, and H. Fethallah, "QoS-aware web service selection based on harmony search," in ISKO-Maghreb: Concepts and Tools for knowledge Management (ISKO-Maghreb), 2014 4th International Symposium, 2014, pp. 1-6.

[20] D. Wang, H. Huang, and C. Xie, "A Novel Adaptive Web Service Selection Algorithm Based on Ant Colony Optimization for Dynamic Web Service Composition," in Algorithms and Architectures for Parallel Processing, ed: Springer, 2014, pp. 391-399.

[21] X. Wang, Z. Wang, and X. Xu, "An Improved Artificial Bee Colony Approach to QoS-Aware Service Selection," in Web Services (ICWS), 2013 IEEE 20th International Conference on, 2013, pp. 395-402.

[22] T. Zhang, "QoS-aware Web Service Selection based on Particle Swarm Optimization," Journal of Networks, vol. 9, pp. 565-570, 2014.

[23] V. S. Jeure and Y. Kulkarni, "Approaches for Web Service Selection," 2014.

[24] X. Zhao, Z. Wen, and X. Li, "QoS-aware web service selection with negative selection algorithm," Knowledge and Information Systems, vol. 40, pp. 349-373, 2014.

[25] M. Fahad, N. Moalla, and Y. Ourzout, "Dynamic Execution of a Business Process via Web Service Selection and Orchestration," Procedia Computer Science, vol. 51, pp. 1655-1664, 2015.

[26] 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.

[27] T. O. Group, "SOA Reference Architecture," 2011.

[28] L.-J. Zhang and J. Zhang, "SOA reference architecture," Web services research for emerging applications: Discoveries and trends, Information Science Reference, pp. 1-15, 2010.

[29] Y.-Y. FanJiang and Y. Syu, "Semantic-based automatic service composition with functional and non-functional requirements in design time: A genetic algorithm approach," Information and Software Technology, vol. 56, pp. 352-373, 2014.

[30] W. Wang, Q. Sun, X. Zhao, and F. Yang, "An improved particle swarm optimization algorithm for QoS-aware web service selection in service oriented communication," International Journal of Computational Intelligence Systems, vol. 3, pp. 18-30, 2010.