Snowball Framework for Web Service Composition in SOA Applications

Mohamed Elkholy[1]
Computer Engineering Department
Pharos University in Alexandria
Alexandria, Egypt

Youcef Baghdadi[2]
Department of Computer Science
Sultan Qaboos University
Muscat, Oman

Marwa Marzouk[3]
Information Technology Department
Matroh University
Matroh, Egypt

***Abstract*—Service Oriented Architecture (SOA) has emerged as a promising architectural style that provides software applications with high level of flexibility and reusability. However, in several cases where legacy software components are wrapped to be used as web services the final solution does not completely satisfy the SOA aims of flexibility and reusability. The literature review and the industrial applications show that SOA lacks a formal definition and measurement for optimal granularity of web services. Indeed, wrapping several business functionalities as a coarse-grained web services lacks reusability and flexibility. On the other hand, a huge number of fine-grained web services results in a high coupling between services and large size messages transferred over the Internet. The main research question still concerns with "How to determine an optimal level of service granularity when wrapping business functionalities as web services?" This research proposes the Snowball framework as a promising approach to integrate and compose web services. The framework is made up three-step process. The process uses the rules in deciding the web services that have an optimal granularity that maintains the required performance. To demonstrate and evaluate the framework, we realized a car insurance application that was already implemented by a traditional approach. The results show the efficiency of snowball framework over other approaches.**

Keywords—Service oriented architecture (SOA); web service granularity; web service composition; software flexibility; snowball composition framework

I. Introduction

SOA allows software systems to be composed as a group of loosely coupled software components called services [1]. SOA aims to provide cost effective flexible solution to business organizations [2, 3]. However, SOA had not gained an extreme popularity until the emerging of web service technology in early 2000s [4]. Since that time, web service became the main trend to implement SOA systems [5]. Several organizations tend to wrap legacy software components in the form of web services to implement SOA-based applications [6]. Wrapping legacy software into web services reduces the cost of implementing new software systems. However, in several cases where legacy components are wrapped to be (re)used as web services, the final solution does not completely satisfy the SOA aims of flexibility and reusability. The reason behind that is the unsuitable level of service granularity. Service granularity has two different perspectives: business perspective and IT perspective. From a business perspective, service granularity is associated with the amount of business tasks fulfilled with that service. On the other hand from IT perspective, web service granularity is associated with size of data transferred from or towards the service as well as its code length [7].

Service granularity affects reusability, efficiency and performance of the services. Wrapping several business functionalities as a coarse-grained web services leads to a single use service [8]. Such service lacks reusability and flexibility since the separation of concerns and cohesion are missing. On the other hand, composing business tasks from large number of small fine-grained services leads to high coupling between services. Such situation leads to communication complexity and degraded performance. That is, an incorrect service granularity leads to bad performance, low reuse possibilities, inappropriate abstraction levels, and services without business value [9].

It is critical to balance between coarse-grained and fine-grained web services while mapping SOA design to individual web services [10]. Unfortunately, the literature lacks detailed studies about service granularity and its impact on reusability, flexibility, and performance [11].

Consequently, one of the main problems that faces developers while developing web services-based SOA is the difficulty to determine optimal service granularity, especially as there is no theoretical definition for service granularity in the literature.

The main research question still concerns with "How to determine an optimal level of service granularity when wrapping business functionalities as web services?"

This research proposes the Snowball as a promising approach to compose web services in SOA-based applications. Snowball is framework made up of a set of rules and a three-step process. The process uses the rules to check the right and optimal granularity of the services. It first decomposes a Business Process (BP) into smaller sub-processes that are further decomposed into business tasks, each of which is a set of activities. Next, it maps the tasks into individual fine-grained web services. Then it checks the fine-grained web services against the rules, in order to allow their integration. Finally, it optimizes the granularity.

Snowball aims at providing web services that have the optimal granularity while maintaining the required flexibility, reusability, and high performance in terms of low size of data transferred. It is meant to be used by organizations that want to offer its functionalities to users as web services, and can also be used by organizations to build up their own business applications.

To demonstrate and evaluate the framework, we realized a car insurance application that was already implemented by a traditional approach. The results show the efficiency of snowball framework other traditional approaches.

Moreover, the proposed framework has an advantage over other composition frameworks that generally use Business Process Execution Language (BPEL). It integrates and composes services functionalities before the implementation phase. Hence, the framework allows three different modes of services: wrapping legacy components, invocation from a service provider, or creation from scratch (coding). Therefore, Snowball eliminates the utilization of glue code languages such as BPEL, which leads to degraded performance and hard validation tests.

This paper is organized as follows: Section 2 presents related work. Section 3 develops the Snowball framework, i.e., rules and methodology. Section 4 presents the results of the empirical study. The conclusion section summarizes the contribution, its limitation, its impacts, and future work.

II. Related Work

One of the main challenges in web service applications concerns with the granularity of services. This section analyzes different popular methodologies for SOA applications, with respect to the granularity of web services. Several models tried to formalize different processes for an organization to adopt SOA [12]. However, fewer researches focused on services granularity and size of service messages.

SOMA is a popular SOA design framework, introduced by IBM, that models business functionality as coherent individual services. To implement new software for an organization, SOMA defines a domain decomposition approach to perform the design phase. The main idea is to decompose the business into logical coherent functional areas. Each area consists of related processes that are further split to smaller sub processes [13]. Each sub process is decomposed into a set of activities which are listed together to form service portfolio [14]. Each service's functionality in the service portfolio is assigned to a web service. SOMA has no restriction on service granularity or on the size of service messages, whether the service is from the legacy system or from external services. Hence, several SOMA designs that lead to large services that perform several individual functionalities, hence, missing the required flexibility [15]. Such situation leads to a set of non-reusable services neglecting SOA aims of software reusability [16]. On the other hand, SOMA application might be implemented with extremely high number of fine-grained services. Such implementation may lead to large size of data transfer while aggregating these services together [17].

Another popular model for SOA adoption is Service Oriented Architecture Maturity Model (SOAMM). SOAMM defines a model for monitoring different levels of development, implementation, and usage of SOA [18]. SOAMM defines a set of characteristics for organizational architecture that are essential for any organization to be able to implement web services-based SOA. SOAMM defines service selection and collaboration between services from the business point of view only [19]. However, SOAMM does not define rules for service granularity from IT perspective such as size of input/output messages.

Thomas Erl [20] defined Mainstream SOA Methodology (MSOAM) as a framework to design, implement, test, and deploy web services. MSOAM identifies seven activities during analyses and design phases. It starts by Ontology definition, then perform business model Alignment. Further it performs service oriented design to develop services that fulfill each process of business functionality. This framework has an advantage in defining dependencies between services. However, it does not define how these dependencies can affect service granularity. Thus several MSOAM applications suffer from coarse-grained web services lacking flexibility and reusability.

Business Process-driven Methods [21] is considered one of the most common strategy used to identify services in SOA. This method uses clustering algorithm to identify services from the business perspective. Business elements are divided into rules and requirements, and then a syntax analysis is applied to perform service selection for each BP [22]. Such method focuses on BPs, and gives less attention to data transfer. The main drawback of this method is the extremely fine-grained services that lead to large amount of communication overheads between services. Implementing web service application with large number of fine grained services increases the size of messages required for services communication [23, 24].

This figures out the problems associated with service granularity while implementing SOA by using web services. The literature lacks theoretical methods to define optimal service granularity. Unsuitable level of service granularity leads to significant drawbacks in flexibility, efficiency and performance of SOA based applications [25]. The proposed framework assists developers in deciding the optimal granularity of web services that maintains flexibility and high performance.

III. Proposed Snowball Framework

The proposed snowball framework provides a systematic approach to determine the optimal service granularity for web services-based SOA, in terms of performance and efficiency.

It defines a set of rules and a three-step process. The rules specify mapping business tasks to IT web services. The rules also define the conditions under which two services or more should be integrated together. The three processes define the actions taken to apply the rules to the business tasks step by step till getting a suitable level of service granularity. It aims at assuring an optimal service granularity that satisfies lower coupling and higher cohesion.

A. Service Granularity

The framework considers two different properties of service granularity: (1) the business functional granularity, representing the number of elementary business tasks fulfilled by the service, and (2) the data granularity, concerning with size of input/output data included in the service messaging.

From the business perspective, the fine-grained service is the service that performs an atomic task [23]. While from IT perspective, a fine-grained service is the service that has a limited size of data transfer. Thus a service could be fine-grained from a business perspective, and coarse-grained from a data granularity perspective. For instance, a service that displays a map performs a single business task but carries a huge size of data.

B. Snowball Rules to Optimize Service Granularity

The framework provides two sets of rules to optimally and efficiently integrate fine-grained services together, considering both functional granularity and data granularity. Dependencies between services are also a point of concern.

1) Rules to map business tasks into IT services: Mapping business tasks into IT services consists in assigning each single business task to an elementary web service, i.e., an elementary coherent fine-grained service.

a) Rule 1: If a legacy software component satisfies a single business task, then it is wrapped to act as a web service with only one single operation.

b) Rule 2: If the required functionility exists in public/private registries as a web service with one operation, then select it.

c) Rule 3: If the service is to be locally implemented (by coding), then the code includes only one single operation.

Applying these rules results in a high flexibility and reusability of mapped web services. However, increasing the number of individual web services in an application affects its complexity and performance in terms of response time and large size of network traffic [26, 27]. Therefore, there is a need to integrate and compose service into an optimal granularity by using the following rules.

2) Rules to integrate IT services: After assigning elementary business tasks to IT service, the output is a set of fine-grained service. The following rules are applied to these services to achive optimal granularity.

a) Rule 1: Two services Si and Sj are integrated together if:

- The business workflow requires execution of the two services sequentially.
- The input parameters for Si are the same as Sj or the output of Si is the required input for Sj.

b) Rule 2: Two services Si and Sj are integrated together if:

- Si and Sj are in the same business domain and are connected to the same database tables and.
- Si and Sj should be at the same branch of business workflow.

3) Factors that manages services integeration:

Factor 1: Si and Sj have sequential execution.

Factor 2: The output of Si is an input for Sj.

Factor 3: Si and Sj have the same I/O.

Factor 4: Si and Sj have connection to the same database.

Factor 5: Si and Sj have data dependencies, i.e., Sj cannot be executed until Si is completed as Sj has one (not all) of its inputs passed from outputs of Si.

C. Snowball Steps to Optimize Service Granularity

The Snowball process, shown in Fig. 1, consists of three main steps that should be completed to provide SOA applications with the required flexibility, reusability, and high performance of the services that compose them. Step 1 identifies business tasks, step 2 maps each business task into an IT service, whereas step 3 optimizes the service integration.

1) Service identification: Each BP is broken down into smaller sub-processes and then to single elementary tasks. An elementary task performs atomic coherent business functionality. Then all the elementary tasks are listed in a task table, as exemplified in Table I.

2) Mapping business tasks into IT services: This step uses the first aforementioned set of three rules to map business tasks into IT services. Mapping business tasks to a web service means selecting a web service that fulfills the business functionality of the task. Each atomic business task is mapped to an elementary web service that would be wrapped from the legacy systems or discovered over web service registries, or even implemented as a new web service. Different activities of mapping atomic business functionality to web services are shown in Fig. 2.

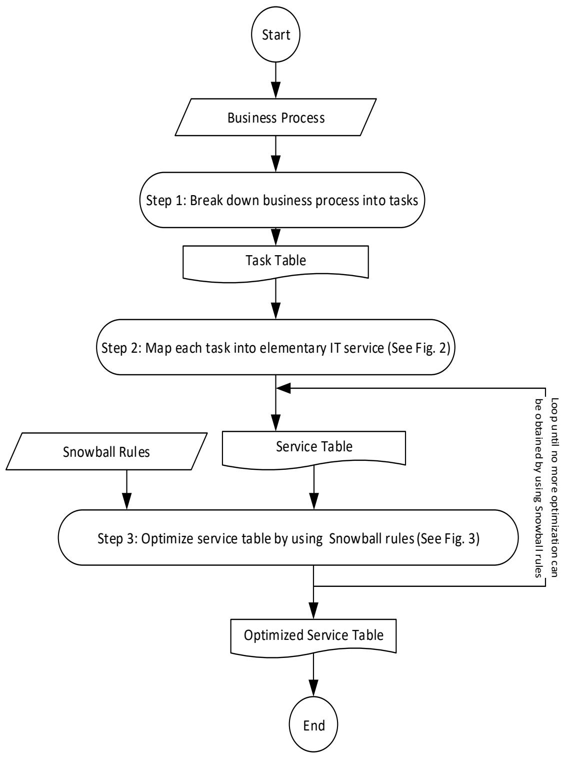

Start
Business Process
Step 1: Break down business process into tasks
Task Table
Step 2: Map each task into elementary IT service (See Fig. 2)
Snowball Rules
Service Table
Loop until no more optimization can be obtained by using Snowball rules
Step 3: Optimize service table by using Snowball rules (See Fig. 3)
Optimized Service Table
End

Fig. 1. Snowball Process.

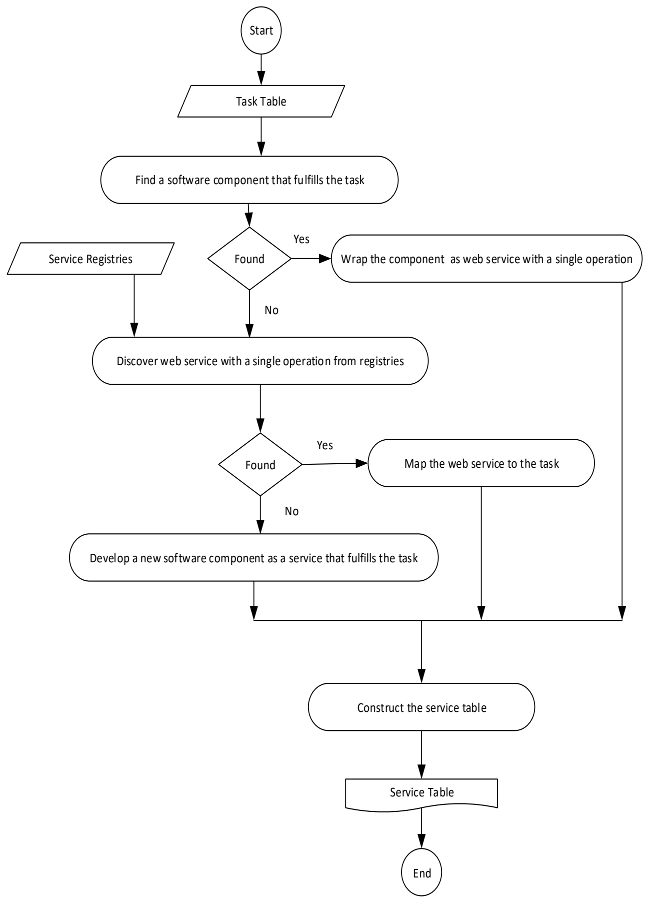

Start
Task Table
Find a software component that fulfills the task
Service Registries
Found
Yes
Wrap the component as web service with a single operation
No
Discover web service with a single operation from registries
Found
Yes
Map the web service to the task
No
Develop a new software component as a service that fulfills the task
Construct the service table
Service Table
End

Fig. 2. Details of Step 2 of the Snowball Process.

Up till this step the system is composed of fine-grained IT services that ensure a high level of cohesion. All the atomic services are then listed in a services table. The services table lists all the used services, including their business functionality, I/O parameters, connected database, and dependent services. The dependency between services is divided into control flow dependency and dataflow dependency. The I/O parameters and connected database are chosen as they have the most effect on services coupling. The dependency between services is further used to construct the criteria of integration between two services or more.

3) Optimization of the service table: The third step is responsible of optimizing the integration of different individual services together. Such integration avoids building applications from fine-grained web services that increases the interaction between the application and outer invoked services. This scenario leads to poor performance. The integration process would reduce the total size of I/O messaging of the client application to maintain high performance. Unlike traditional composition such as BPEL, the integration in the Snowball process consists of adding the business functionality of the first service to the functionality of second service. Thus, the integration of two services functionalities results in new service that performs the functionalities of both services. Snowball integrates services with each other in recursive rounds. In each round, a service is added to the existing one(s) to constitute a new integrated service. The newly integrated services will act as elementary services in the next round and so on till we get a service where no more integration process can be applied according to the rules. After each round the service table is updated to list the newly integrated services with their parameters. Hence, services integration process is repeated in recursive order by applying the second aforementioned set rules, in order to control service integration, as shown in Fig. 3.

In this step, Snowball framework applies the aforementioned second set of rules in order to achieve two main objectives for services integration regarding size of transferred data and database connections.

First objective: Minimize overall service interface messages.

This objective is achieved by applying rule1 regarding both the business workflow and the size of transferred data. If S1and S2 have the same input parameters, they are integrated together in a new service S3. During runtime S3 will be invoked with the input parameters of S1 and returns the output parameters of S2 rather than sending the same data twice from S1 and S2. The decrease in the sent and received data between the application and outer web services has a great effect on performance, especially for huge size of parameters.

Second objective: Minimize Database connections.

This objective regards the connection between web services and databases. Rule 2 may not be available for discovered services as the Web Service Describing Language (WSDL) file almost contains the input/output parameters without information about database connections. However, if the web services are wrapped from the legacy software asset or implemented as new services, information about connections between data bases and services are available.

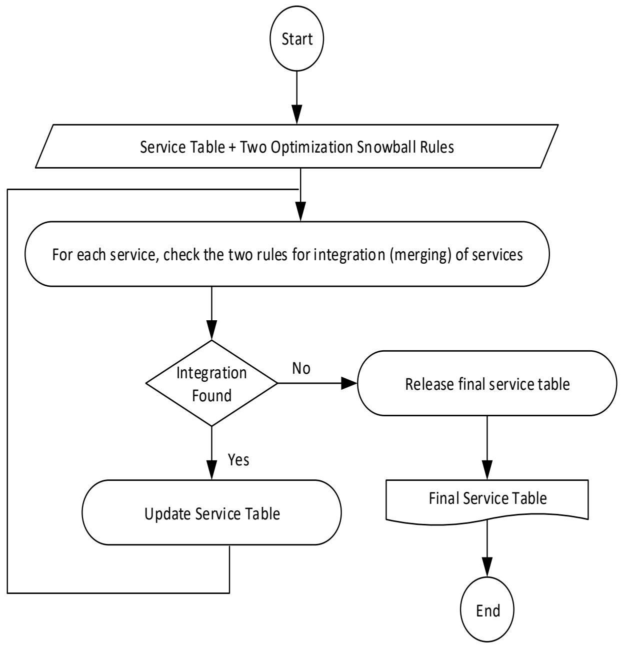

Start
Service Table + Two Optimization Snowball Rules
For each service, check the two rules for integration (merging) of services
Integration Found
No
Release final service table
Yes
Update Service Table
Final Service Table
End

Fig. 3. Step 3: The Optimization Process.

IV. Evaluation

To proof the significant enhancement of the proposed framework we focused on two main factors that affect SOA applications. The two factors are performance and efficiency. These two factors provide a clear measurement value that reflects how optimal a service granularity is. Performance is measured by calculating the response time between a service invocation and its reply. Efficiency is measured by calculating the total size of message used during the process of service request and reply. An insurance company was selected to evaluate the proposed work because insurance applications combine different functionalities from different business domains.

A. Case Description

The insurance company has many valuable software assets that support many of its running business processes. The company would like to offer the existing software functionalities and any newly created ones as web services. The focus is on the SOA-application that support car insurance claim BP, whereby a client request insurance reimbursement for his/her car. To prove the significant enhancement of the proposed framework the system functionalities are built using two approaches. In the first approach the system is designed using traditional SOA approach. In the Second approach, the required functionalities are built using Snowball framework. For the two approaches the response time and the total size of data transferred (message size) are calculated starting from receiving a user request till the claim process is completed.

B. Applying Traditional Method and Snowball Framework

For a traditional approach, each one of the listed in Table I one and representing different business tasks is mapped to an individual web service. Then a service portfolio is constructed without regarding the service granularity.

For snowball, the three-step processes mentioned above were executed as follows:

Step 1: The first step takes the car insurance claim BP as input and breaks it down to elementary tasks, which results in a task table, as shown in Table I.

Step 2: The second step takes the task table as input and maps each of the tasks to an elementary web service with single operation according to the first set of rules, as shown in Fig. 2.

Table II shows the resulting services. Each service describes the mapped task, the input and output parameters, the database to which it is connected, and its dependency to other services.

Step 3: The third step takes the service table as input and performs optimization according to the second set of rules, as shown in Fig. 3. The optimization is a recursive, where each iteration updates the service table according to the new integrated services. The process ends when there are no more services to be integrated. Every service in the service table is tested whether it can be integrated with another service according to the five factors defined in section 3.B.

Applying such factors to the service table presented in Table II results in the following integrations, as shown in Table III.

The three services S2, S3, and S4 require the same input: client ID, client name, and insurance document ID. Moreover, the three services are connected to the same databases: client database and the insurance document database. The three services also have control dependency as they all should be executed sequentially before S5 is invoked. Accordingly, the three services S2, S3, and S4 are aggregated into one service, named Sa. The three services S6, S7, and S8 can also be aggregated together, as they have client ID as an input parameter. The three services are also connected to the same database that is cars database. S6, S7, and S8 should be executed as perquisite condition for S9 and S11. Accordingly, the services S6, S7, and S8 are aggregated into one service, named Sb.

Table III shows the final services produced by Snowball. These are S1, S5, Sa, Sb, S9, S10, S11, and S12. For each integrated service, Table III describes the service functionality, the input and output parameters, the database to which it is connected, and its dependency to other services.

C. Experimental Results

The insurance application is built as a web services-based application by two different approaches: SOMA and Snowball.

- Traditional SOA application build up using 12 separated services.
- Snowball designed application that is built up using only 8 services after integrating the dependent services.

1) The experiment: The two applications were built using C# in .Net environment. The services were implemented on IBM server with processor Xeon E5, whereas, the service invocations were applied from a desktop with CPU core i7 and 16 M Byte memory. SOAP-UI was used as a testing tool to calculate the message size and the response time. The experiment was repeated ten (10) times and the average value was calculated. The response time and message size were calculated.

TABLE I. THE ELEMENTARY TASK OF THE BP

-	Receive a claim
-	Check insurer payment
-	Check whether the claim is in the insured period
-	Check whether the insurer have many claims (manipulator)
-	Take a decision for the claim
-	Get car year
-	Get car making company
-	Get car model
-	Get car price
-	Calculate estimated cost
-	Get new car cost
-	Get a decision for payment
-	Payment

TABLE II. RESULTING SERVICES SHOWING THE ELEMENTARY TASK OF THE BP

Service	***Functionality***	***Input parameters***	***Output parameters***	***Connected Database***	***Dependent services***
S1	Receive a claim	- Request from client			
S2	Check insurer payment	- -Client ID - -Client name - -Insurance document ID	- Boolean value	Client DB Insurance Document DB	S3, S4
S3	Check whether the claim is in the insured period	- -Client ID - -Client name - -Insurance document ID.	- Boolean value	Client DB Insurance Document DB	S2, S4
S4	Check whether the insurer has many claims (manipulator)	- -Client ID - -Client name - -Insurance document ID.	- Boolean value	Client DB Insurance Document DB	S2, S3
S5	Take a decision for the claim	- -Three Boolean values	- Boolean value		S2, S3, S4
S6	Get car year	- Car ID	- Car year	Cars DB	_
S7	Get car making company	- Car ID	- Making company	Cars DB	_
S8	Get car model	- Car ID	- Car model	Cars DB	_
S9	Get damaged component prices	- -Car year - -Car making company - -Car model	- Damaged component price	External DB	S6, S7, S8
S10	Calculate estimated maintenance cost	- -Damaged component price - -Maintenance cost	- Maintenance cost		S9
S11	Get new car cost	- -Car year - -Car making company - -Car model	- New car cost	-External DB	S6, S7, S8
S12	Get payment decision	- -Maintenance cost - -New car cost	- String		S10, S11

TABLE III. FINAL SERVICES COMPOSITION BY SNOWBALL FRAMEWORK

Service	***Functionality***	***Input parameters***	***Output parameters***	***Connected Database***	***Dependent Services***
S1	Receive a claim	Request from client			
Sa= integration of S2, S3, S4	Check insurer payment: check whether the claim is in the insured period and whether the insurer has many claims (manipulator)	- Client ID - Client name - Insurance document ID	- 3 Boolean values	- Client DB - Insurance Document DB	
S5	Take a decision for the claim	- 3 Boolean values	Boolean value		Sa
Sb= integration of S6, S7, S8	Get car year, car making company, and car model	- Car ID	Car year Making company Car model	- Cars DB	
S9	Get damaged component price	- Car year, - Car making company - Car model	Damaged component price	- External DB	Sa
S10	Calculate estimated maintenance cost	- Damaged component price - Maintenance cost	Maintenance cost		S9
S11	Get new car cost	- Car year - Car making company - Car model	New car cost	- External DB	Sb
S12	Get payment decision	- Maintenance cost - New car cost	String		S10, S11

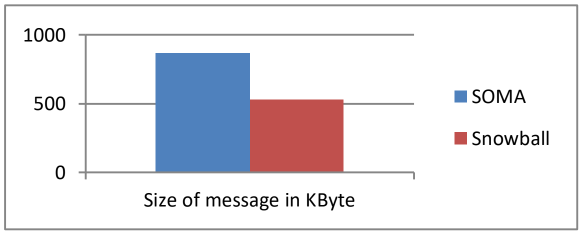
- Reply time: calculated using SOAP-UI

2) Results: The results listed in Table IV show a significant enhancement in the message size and respond time while maintaining the same flexibility and reusability features.

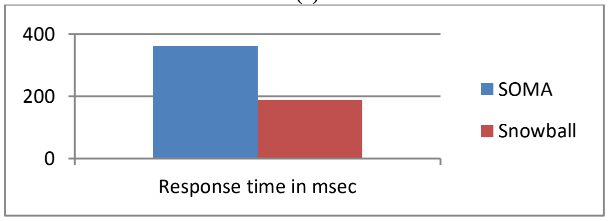
Fig. 4 shows the difference between traditional application and Snowball application across three metrics: message size (Fig. 4a), response time (Fig. 4b), and database connection (Fig. 4c).

TABLE IV. EXPERIMENTAL RESULTS

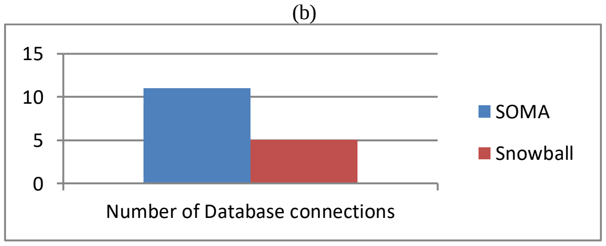
Approach	*Size of transferred message*	*Response time*	*Number of connected database*
Traditional SOMA	850 Kbyte	370 msec	11 databases
Snowball framework	510 Kbyte	190 msec	5 databases



(a)



(b)



(c)

Fig. 4. (a) Total Message Size in Kbyte, (b) Total Response Time in Msec, (c) Number of Connected Databases.

D. Discussion

The experimental results show a significant enhancement in two important parameters that affect the SOA applications. The first parameter is the performance that is measured by the response time of the invoked service. The average response time decreased by 48% from 362 in case of traditional applications, to be 188 msec using Snowball framework. The second parameter is the application efficiency that is measured by the size of transferred data. The transferred data significantly decreased by 39 % to be 530 KB in snowball application rather than 869 KB in case of traditional application. Such decrement in data size provides better network utilization specially when using wireless connections with narrow band width. Another significant enhancement was decreasing the number of connections between web services and database to only 5 databases rather than 11 in traditional applications.

Such results proof the significant enhancement of SOA applications while using snowball approach. Using snowball frame work defines an optimal service granularity that significantly decreases the application response time and its total transferred l data size.

It is worth mentioning that the experiment is meant to evaluate three criteria in a specific testing configuration: response time, size of the message, and number of DB connections. It also considers only one running BP to clarify our idea without adding more complexity in designing and implementing more BP.

Adding other metrics that measure a quality SOA-based application may result in a tradeoff.

V. CONCLUSION

The granularity of individual services that compose web service-based SOA is an important issue. Service granularity has significant impacts on the quality of the services regarding performance and efficiency. This work has tacked the issue of how to find an optimal service granularity. The Snowball framework was proposed to adjust web service granularity to maintain flexibility, performance, and efficiency of SOA systems. The framework is made up of two sets of rules and a three-step process.

The proposed framework was demonstrated and evaluated through the development of a web-services-based SOA application that supports the car insurance claim BP of an insurance company. The application was developed by traditional SOMA approach and Snowball framework. The experimental results show significant enhancement of Snowball over SOMA, in terms of response time, message size and DB connections. The snowball is limited to the optimization of the granularity from the perspective of performance of the services and the applications composed out of them.

The framework has practical and theoretical impacts. The developers of SOA-based applications can use it to optimize the granularity of their services to enforce their reuse and consequently the time to market. From, a theoretical perspective, the proposed work opens issues related to the optimization of the service granularly with respect to other quality criteria.

The future work will discuss the security enhancement offered by Snowball approach. Also our future work will analyze the problems associated with web service run time failure while using Snowball approach.

REFERENCES

[1] Mohsen, A. and Naeem, K., "A review and future directions of SOA-based software architecture modeling approaches for System of Systems," In Service Oriented Computing and Applications, Volume 12, Issue 3–4, pp 183–200.2018.

[2] Pulparambil, S., and Baghdadi, Y., "Service oriented architecture maturity models: A systematic literature review," Computer Standards & Interfaces, 61, 65-76. (2019).

[3] Papazoglou, M. P. and Georgakopoulos, A. D., "Service-Oriented Computing," In Communications of the ACM, vol. 46 (10), pp. 24-28. 2003.

[4] Erol, O., Mansouri, M., and Sauser, B., "A framework for enterpriseresilience using service oriented architecture approach," In: 20093rd annual IEEE systems conference. IEEE, pp 127–132.2009.

[5] Baghdadi, Y.,"Modelling business process with services: towards agile enterprises," International Journal of Business Information Systems, 15(4), 410-433. 2014.

[6] Baghdadi, Y. and Al-Bulushi, W. "A guidance process to modernize legacy applications for SOA," SOCA 9, 41–58 . https://doi.org/10.1007/s11761-013-0137-3.2015.

[7] Baccar, S. Baccar, S., Rouached, M., Verborgh, "Declarative Web service composition using proofs," In Service Oriented Computing and Applications Volume 12, Issue 3–4, pp 371–389. 2018.

[8] Immonen, A. and Pakkala, D., "A survey of methods and approaches for reliable dynamic service compositions," In Service Oriented Computing and Applications, Volume 8, Issue 2, pp 129–158.2014.

[9] Ayed Alwadain, Erwin Fielt, Axel Korthaus, Michael Rosemann,"Empirical insights into the development of a service-oriented enterprise architecture,"Data & Knowledge Engineering,Volume 105, Pages 39-52,ISSN 0169-023X.2016.

[10] Ding, Z. and Zhou, Z., "Race Test: harmful data race detection based on testing technologyin WS-BPEL," In Service Oriented Computing and Applications 13:141–154 doi.10.1007/s11761-019-00261-1.2019.

[11] Silic, M., Delac, G., and Srbljic, S., "Prediction of atomic web service reliability based on k-means clustering," In proceedings of the 9th Joint Meeting on Foundations of Software Engineering, pages 70-80, Saint Petersburg, Russia — August 18 - 26, ISBN: 978-1-4503-2237-9 doi>10.1145/24914112491424.2013.

[12] Baghdadi, Y., "SOA Maturity Models: Guidance to Realize SOA," International Journal of Computer and Communication Engineering 3 : 372-378, DOI:10.7763/IJCCE. 2014.V3.352.2014.

[13] A. Arsanjani, L. Zhang, M. Ellis, A. Allam and K. Channabasavaiah, "S3: A Service-Oriented Reference Architecture," in IT Professional, vol. 9, no. 3, pp. 10-17, doi: 10.1109/MITP.2007.53. May-June 2007.

[14] Osshiro M. et al. Márcio Osshiro, Elisa Y. Nakagawa, Débora M. B. Paiva, Geraldo Landre, Edilson Palma, Maria Istela Cagnin, "Cambuci: A Service-Oriented Reference Architecture for Software Asset Repositories," In: Latifi S. (eds) Information Technology - New Generations. Advances in Intelligent Systems and Computing, vol 558, Springer, Cham https://doi.org/10.1007/978-3-319-54978-1_74 ISBN 978-3-319-54978-1.2018.

[15] Laradi, N., Bernard, P. and Plaisent, M., "The Organizational Impacts of a Service Oriented Architecture," In Journal of Economic Development, Management, IT, Finance and Marketing, 10(1), 88-96, March 2018.

[16] X. Liu, Y. Ma, G. Huang, J. Zhao, H. Mei and Y. Liu, "Data-Driven Composition for Service-Oriented Situational Web Applications," in IEEE Transactions on Services Computing, vol. 8, no. 1, pp. 2-16, , doi: 10.1109/TSC.2014.2304729. Jan.-Feb. 2015.

[17] Camacho, J. A., Chamorro, C. D. and Caicedo, N. G., " Implementation by means of Web service with Service Orientation Architecture for a System Tele-operated," In International Seminar of Biomedical Engineering (SIB) IEEE: 10.1109/SIB.2018.8467754, 16-18 May, Bogota, Colombia. 2018.

[18] Candido, G. Colombo, A., Barata, J. and Jammes F, "Service-oriented infrastructure to support the deployment of evolvable production systems," In IEEE Trans. Ind. Informat., vol. 7, no. 4, pp. 759-767. 2011.

[19] Razavian, M., Lago, P. A., "Systematic Literature Review on SOA Migration, "In Journal of Software: Evolution and Process, 27(5), 337-372. https://doi.org/10.1002/ smr.1712.2015.

[20] Erl, T., "SOA Principles of Service Design (paperback)," The Book. Prentice Hall Press Upper Saddle River, NJ, USA ISBN:0134695518 9780134695518.2016.

[21] Choi, D.L. Nazareth, H.K. Jain, "The impact of SOA implementation on IT-business alignment: a system dynamics approach," In ACM Trans. Manag. Inf. Syst. 4, 3 (https://doi.org/10.1145/2445560.244556). 2013.

[22] Baryannis, G., Kritikos, K. and Plexousakis, D., "A specification-based QoS-aware design framework for service-based applications," In Service Oriented Computing and Applications, 11: 301.doi10.1007/s11761-017-0210-4.2017.

[23] Niknejad, N., Hussin, A.R.C., and Amiri, I.S., "Introduction of Service-Oriented Architecture (SOA) Adoption," In: The Impact of Service Oriented Architecture Adoption on Organizations. Springer Briefs in Electrical and Computer Engineering. Springer, Cham.2019.

[24] Guinard, D., Trifa, V., Karnouskos, S., Spiess, P. and Savio, "D. Interacting with the SOA-Based Internet of Things: Discovery, query selection, and on-Demand Provisioning of Web service," In ServComput IEEE Trans 3(3):223–235. 2010.

[25] Kumar, L. Kumar, S. R. and Sureka, A. (2017) Using source code metrics to predict change-prone web service: A case-study on ebay services. In 2017 IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE)Klagenfurt, Austria IEEE, DOI:10.1109/MALTESQUE.2017.7882009.

[26] Gazzarata, R, Vergari, F, Cinotti, T. S. and Giacomini, M., "A standardized SOA for clinical data interchange in a cardiac tele monitoring environment," In IEEE J. Biomed. Heal. Informatics, vol. 18, no. 6, pp. 1764-1774. 2014.

[27] H. M. Sneed, "Integrating legacy software into a service oriented architecture," Conference on Software Maintenance and Reengineering (CSMR'06), 2006, pp. 11 pp.-14, doi: 10.1109/CSMR.2006.28.