An Approach for Effort Estimation of Service Oriented Architecture (SOA) Projects

Esraa A. Farrag[1]*, Ramadan Moawad[2], Ibrahim F. Imam[1]

[1] Arab Academy for Science, Technology and Maritime Transport, Cairo, Egypt.
[2] Future University, Cairo, Egypt.

* Corresponding author. Email: Esraa_farag16@yahoo.com
Manuscript submitted September 14, 2015; accepted November 20, 2015.
doi: 10.17706/jsw.11.1.44-63

Abstract: In the last few decades SOA (Service Oriented Architecture) has become the new trend in the IT industry. Many organizations tend to migrate to SOA in order to cope with the rapidly changing business. Effort estimation of SOA projects has become a real challenge to project managers due to the limited literatures addressing this issue. The traditional effort estimation techniques do not fit SOA projects entirely, as SOA has unique characteristics were not addressed by the traditional cost estimation approaches. These unique SOA characteristics include: loose coupling, reusability, composability and discoverability. On the other hand, cost estimation approaches that were proposed to estimate SOA projects, are still immature and most of them are impractical. They cannot be used in real life projects, as they are more guidelines than actual practical cost estimation approaches. This paper proposes an effort estimation approach for SOA projects that has been applied to different variety of services. It considers SOA characteristics and the various cost factors for different types of services including available, migrated, new and composed services. This proposed approach provides effort estimation technique for each type of service. The proposed approach also gives effort distribution among project phases for easily resources allocation. This framework has been applied to real life projects in the IT industry as the SOA project is divided into its component services and each service is estimated solely based on its type. The services' efforts are then aggregated to calculate the project's overall effort. The estimated effort relative error in the case studies ranges from 3.66 % and 19.14%.

Key words: Software engineering, effort estimation, cost estimation, phased effort estimation, SOA projects, new service, migrated service, composed service.

1. Introduction

Recently, SOA (Service Oriented Architecture) has become the new trend in the IT industry. Many organizations tend to migrate to SOA in order to cope with the rapidly changing business [1]. These organizations are motivated by both business and technical benefits of SOA. The main SOA business benefits are higher productivity and better quality in less time [2]. However, the core benefits of SOA are mainly technical benefits [3] including separation of concerns , enhanced product quality and better coping with changing business. Despite these benefits, the main challenge with SOA projects is that project managers have no idea how to estimate the effort of SOA projects [4]. The traditional software effort estimation techniques don't fully fit SOA projects as SOA has unique characteristics [5]. These

characteristics include: loose coupling, reusability, composability and discoverability. SOA characteristics have a major impact on the cost that traditional cost estimation approaches can't address [6].Many SOA cost estimation approaches were proposed to solve this gap. However these approaches are more guidelines than actual estimation approaches.

Section 2 discusses the related work to our research with a brief overview on the existing cost estimation approaches and service classification. Our approach is presented in Section 3, that classifies the services into types and each type has its cost factors. The approach is applied to case studies that are detailed in Section 4. Section 5 concludes the paper.

2. Related Work

As this paper is mainly about the cost estimation for SOA projects, so this section is divided into two main subsections: effort estimation approaches and classification of services. The effort estimation approaches are subdivided into SOA specific cost estimation approaches and traditional cost estimation approaches.

2.1. Different Cost Estimation Approaches

In this subsection we will give a quick glance at the existing cost estimation approaches. Both traditional software cost estimation approaches and SOA specific cost estimation approaches are included in our scope.

2.1.1. SOA specific cost estimation approaches

SOA cost estimation approaches were proposed because the traditional software effort estimation approaches didn't address SOA characteristics. These unique characteristics include: loose coupling, reusability, composability and discoverability. The SOA cost estimation approaches include: Linthicum formula, Service Migration and Reuse Technique (SMART), AUS-SMAT framework and Divide and Conquer approach.

Linthicum Formula

This formula is one of the earliest approaches in SOA cost estimation [4]. The cost of SOA is calculated based on equation (1)

$$Cost\ of\ SOA = (Cost\ of\ Data\ Complexity + Cost\ of\ Service\ Complexity + Cost\ of\ Process\ Complexity + Enabling\ Technology\ Soluti \quad (1)$$

This formula considered the complexity nature of SOA, however the equation is more theoretical and cannot be considered as a real metric [5], [7]. Also it is hard to be applied to projects in the IT industry.

Service Migration and Reuse Technique Approach (SMART)

SMART [8] is a technique that tackles the legacy systems migration to SOA issue. It helps organizations to analyze their legacy systems in order to determine whether SOA migration is feasible.

The main disadvantage of SMART is that it considers only service migration and cannot be generalized to include all other types of services like new or composed services. Also, it is more guidelines than a practical cost estimation approach, which makes it hard to be applied in projects of the IT industry.

AUS-SMAT Framework

AUS-SMAT Framework has been developed by NICTA organization [9]. It aims to develop a framework for scope, cost and effort for SOA projects. In this framework the SOA project is decomposed into its constituent services. Each service is classified to its type either: service mining, service development, service integration or application development. Each service type has its own specific activities, templates, cost factors and cost functions. The overall cost of the project will be a summation of the cost of its constituent services.

This framework acknowledges that every service type has its own cost factors; however the framework is still being developed[7].

Divide and Conquer (D&C) Approach

The Divide and Conquer approach [5] was inspired from the divide and conquer algorithm which is used to solve complicated problems. This cost estimation approach not only takes the advantage of composability nature of services but also solves the complexity of services. The whole SOA project is broken down into its basic services; the cost of each service is estimated solely. The overall cost of the application is the summation of the cost of the component services in addition to the integration costs[9]. This approach presents a basic to our proposed approach as will be shown in the next section.

2.1.2. Traditional software cost estimation approaches

For the purpose of this paper, only the traditional approaches which have been adapted to estimate SOA projects are included in this subsection. These approaches are COCOMO II and Function Point.

COCOMO II Model

The Constructive Cost Model II [10] estimates cost of the software based on number of lines of code (LOC). However it is usually criticized as the LOC could be only obtained when the project is completed [11].On the other hand, COCOMOII is inadequate to estimate SOA projects as it doesn't consider service reusability nature of SOA[7] .

The traditional COCOMOII approach has been adapted to fit SOA projects. Tansey and Stroulia[12] have attempted to estimate the cost of SOA by applying both COCOMO II and real option theory to SOA projects. COCOMO II has been applied to service development and service migration, one the other hand real option theory has been applied to service composition. Tansey and Stroulia approach could estimate the cost of SOA project, except it assumes that the cost of developing composite service is the sum of the costs involved in developing its constituent services, however it ignores the additional costs incurred by specifying and testing that composition.

Function Point

Function point [13] estimates the cost of software based on its functional requirements. The traditional function point is based on counting the software functions. These functions include: number of inputs, number of user's outputs, the number of inquiries, number of files and the number of interfaces [13]. The traditional function measures the complexity of software based on 14 cost factors[14].

Unfortunately, traditional Function point doesn't support SOA perfectly as SOA doesn't completely meet traditional function point metrics[15]. Many calibrations to the traditional approach have been proposed in order to estimate the effort of SOA projects. Those calibration approaches are cosmic function point and SOA function point attempts.

Cosmic Function Point for SOA

Cosmic approach [16] was proposed in order to overcome traditional function point limitations when applied to SOA, such as inability to estimate non- monotonic applications and service boundary definition.

COSMIC also involves applying a set of models, principles, rules and processes to the Functional User Requirements (FUR) of a given piece of software. The result is a number represents the functional size of the service.

The main drawback of Cosmic is that it is not designed (yet) to estimate mathematically-intensive software such as: expert systems, simulation software, forecast software, Artificial Intelligence, etc.[17].On the other hand , cosmic presents wide set of guidelines for practical application of COSMIC measurement would still to test and experience[7] .

Calibrated Function Point for SOA

The traditional function point has been calibrated in order to estimate SOA projects[18]. The SOA project

is broken down into its component services .Estimating each service using Calibrated function point approach is achieved by making adjustments to traditional Function Point cost factors to empirically support SOA. The adjustments included eliminating unused function point cost drivers and adding SOA specific driver (service integration).

This approach considered the integration nature of service, however it didn't address the rest of the SOA characteristics .

From the different cost estimation techniques we can conclude that the traditional software effort estimation techniques do not fully fit SOA projects as SOA has unique characteristics [5].SOA characteristics have a major impact on the cost that traditional cost estimation approaches cannot address[6].Many SOA cost estimation approaches were proposed to solve this gap. However these approaches are more guidelines than practical estimation approaches. In the proposed approach a practical methodology is presented that considers the various SOA characteristics.

2.2. Classification of Services

Services are better estimated on their own by separation [16] .Each service type has its own cost factors. In order to estimate the SOA project, it is broken down into its component services. Each service is classified into its basic type. Each type of service has its own cost factors and its cost estimation approach. In this paper, the services are classified based on the service construction [19] into available, migrated, new or composed services.

Available: A service that already exists and will be used as is. Available services may be home-grown or 3rd party service[2].The available service has a zero development effort.

Migrated: A service which is generated through different strategies either wrapping, re-engineering or replacement[18] :

- *Wrapping:* Wrapping [19] is a black box migration strategy in which service interface is built to wrap the existing legacy system;
- *Re-engineering* :Re-engineering is the adjustment of the application to be in a new form to enable adding new functionality to the legacy system easily;
- *Replacement:* is removing the old system and replace it with the new service(s).

Each migration strategy has its own cost factors as detailed in[6].

New: A service built from scratch to satisfy the exact needs .Cost estimation of a new service is supposed to be a straight forward task. As traditional cost estimation approaches could be used .However cost factors of new service has to be also considered. Calibration of the cost factors enhanced the estimation process as in [15].

Composed: A composed service is a service composed of one or more of the above service types [19]. To estimate the cost of this service, the service is broken down into its basic component services. Estimate each service based on its type and all the efforts are summed to get the overall effort of the composed service [19].

3. The Proposed Cost Estimation Approach

In this paper the effort estimation approach is presented. In which the SOA project is broken down into its component services. Each service is classified into its basic type either available, migrated, new or composed. For each type there are cost factors and conditions which are considered. In this section, different estimation approaches are presented and application of these approaches will be presented in the next section.

In the following each type of service and its estimation approach is introduced:

3.1. Available Service

The available service is a service that already exists and ready to be reused. The development cost of the available service is zero; however the main cost of the available service is the integration and testing cost.

3.2. Migrated Service

The migrated service cost estimation approach has been proposed in our previous work[6]. Cost factors related to service migration have been identified in order to estimate the migrated service effort. These cost factors have been distributed among different project phases. Weight for each cost factor in each migration strategy has been assigned. The cost factors are weighted on scales from 1 to 3. 1 represents low effect on cost and 3 represents high impact on cost. This approach aims to estimate the cost of the migrated service early and accurately by knowing the effort of one of the early phases of the project (mainly requirement phase).These steps are shown in Fig. 1.

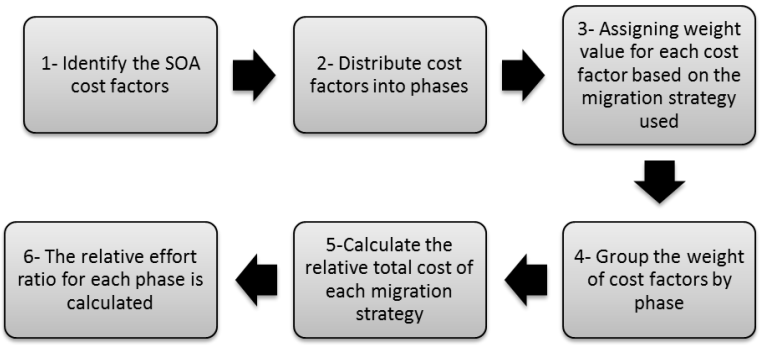

Fig. 1. Block Diagram shows the steps of phased effort for migrated services approach.

The approach is described in the following steps:

3.2.1. Cost factors identification

The cost factors related to service migration have been extracted from the existing literature [18]-[22].

3.2.2. Distribution of cost factors into SOA project phases

These factors have been distributed into the different SOA project phases as in [6]. The factors grouped by phase as follows

Requirements

In this phase the major function of the service is defined. The factors affecting this phase are: business agility cost of integration, business value and business risk;

Design

In this phase the target service is described in a sufficient way that skilled developers can develop the service in minimal effort. The factors involved are: need for original requirements, obsolete legacy system technology, experienced resources needed and need for source code;

Development

Development phase involves the code writing. The development effort is affected by: flexibility, code size, tools support and time required for migration;

Testing

In this phase, all test cases are run to validate and verify the service. SOA has many testing levels which are: functional testing, non-functional testing, integration testing and regression testing[22];

Integration and transition

In this phase, the services are integrated with the desired application .This phase has many factors which

involve: stable environment, maintainability post migration and solving existing problems in legacy systems.

3.2.3. Assigning relative Weights of Each Driver According to each Migration Strategy

Table 1. Factors Weight Distribution among Phases

	Wrapping	Reengineering	Replacement
Planning & Requirements			
Business agility	1	3	2
Integration with partners' cost	3	1	2
business value	1	2	3
Business risk	1	2	3
Planning & Requirements total weight	6	8	10
Planning & Requirements (%)	17.14	21.62	23.81
Design			
Need for Original requirements	1	3	1
Obsolete Legacy system technology	3	2	1
Experienced resources needed	1	2	3
Need for Source Code	3	3	1
Design total weight	8	10	6
Design (%)	22.86	27.03	14.29
Development			
Flexibility	1	2	3
Code size	1	2	3
Tools Support	3	1	1
Time required for migration	1	2	3
Development weight	6	7	10
Development (%)	17.14	18.92	23.81
Testing			
functional Testing	1	2	3
Non-Functional Testing	3	1	1
Integration Testing	3	2	3
Regression Testing	1	2	3
Testing weight	8	7	10
Testing (%)	22.86	18.92	23.81
Transition			
Stable Environment	1	2	3
Maintainability post migration	3	1	2
Solving existing problems in legacy systems(Maintenance)	3	2	1
Transition weight	7	5	6
Transition (%)	20.00	13.51	14.29
Relative Total Cost of Strategy	35	37	42

Each driver is assigned a relative weight according to each migration strategy. The weight scales from 1

to 3. The lowest effect is represented by 1, also weight 3 indicates highest effect, and weight 2 implies moderate effect. The weights are summed by phase and migration strategy. The cost drivers grouped by phase and their relative weight according to migration strategy are shown in[6]. All these are summed in Table 1 which will be our guide in the migrated service phased effort estimation.

3.3. New Service

This paper proposes two approaches to estimate the cost of a new service. The first approach is derived from traditional function point approach. It estimates the total effort of the service using calibrated function point approach . While the other approach estimates the effort of each phase of the service by knowing the effort of only one phase using the ratio concept. Both approaches are proposed and applied in this paper.

3.3.1. SOA calibrated function point estimation approach

The steps of this approach is more like the traditional function point cost estimation approach [13], [24] .The main adjustments have been made in the "General System Characteristic" step . In this step, 14 cost factors are weighted on scale from 0 to 5. As 0 represents no effect on cost and 5 represents maximum effect of this factor on cost. The cost factors in this paper are classified into 3 categories:

- Traditional function point considered factors
- Traditional function point ignored factors
- Calibrated function point considered factors

Each category with its constituent factors will be discussed in details.

Traditional function point considered factors

In this subsection we will discuss the traditional function point factors considered in our approach to estimate SOA projects. The description of each factor has been modified to fit SOA characteristics. For the purpose of this paper, the cost factors are weighted based on the complexity of implementation in SOA .The traditional function point factors considered in our research are: Data communications, Distributed data Processing, Performance, Heavily used Configuration, Service Complexity, SOA maturity, Reusability and Flexibility. These factors and their corresponding description in SOA are shown in Table 2.

It worth noting that, in the traditional FP, transaction rate is considered as a cost factor. While in this paper, transaction rate will be ignored as it will be already considered in the performance factor. Table 2 also shows the degree of influence of each factor.

Concerning the Service complexity factor, it will be weighted based on message model, service discovery, service pattern and security .Service complexity will be measured on scale from 1 to 5. If the service has 0 to 1 of the complexity factors, it is considered as a low complexity service, and takes weight 1 .If the service has 2 to 3 of the complexity factors, it is considered to a medium complexity service with weight 3

If the service has 4 to 5 of the complexity factors, it is a high complexity service and takes the weight 5.

Ignored traditional function point factors

The traditional function point factors ignored in SOA were useful in the traditional software. However these factor in SOA they are either unavailable or do not apply. All these ignored factors are given weight 0. These factors are: Transaction Rate, On-Line data entry, End-user efficiency, On-Line update, Operational ease and multiple sites. These ignored factors and their reason of exclusion is presented in Table 3.

Considered calibrated function point factor (service integration)

This subsection introduces the cost factors added by other adjusted function point SOA estimation [15] and will be considered in our approach. Calibrates FP approach in [18] added service integration as a cost factor in SOA projects.

Integration service indicates whether the application needs other services to be integrated to perform its functions. The service integration will be weighted on scale from 0 to 5. Increasing number of integrated

services will increase the weight of service integration.

Table 2. Traditional Function Point Considered Factors

	Factor	Traditional Function Point	SOA Function point	Degree of Influence
1	Data Communications	The degree to which the application Communicates directly with the processor.	SOA supports a number of communication protocols such as UDDI, XML, and SOAP.	The REST is less complex than SOAP[19] so SOAP is given weight 2 and Rest receives weight 1 More complicated protocols will take more weight.
2	Distributed data Processing	The degree to which the application transfers data among physical components of the application.	The services are distributed while service registry controls these services .	Normal service registry mechanism will take weight 1. Complicated service registry mechanism, we will use 2.
3	Performance	Performance is determined using the response time and throughput of the application.	Service performance is measured in terms of response time, throughput, availability, accessibility, successability and interoperability.	High performance services require extra effort [23] Low performance services will tale weight 1. High performance services will receive 5.
4	Heavily used Configuration	The degree to which computer resource restrictions influenced the development of the application	The hardware infrastructure complexity.	Low complexity infrastructure takes weight 1. Extremely complex infrastructure will take weight 5.

Calculate Total Degree of Influence and the Value Adjustment Factor

After the weights are assigned to each cost factor, the weights will be summed up as in equation 2 .The result is the total degree of influence TDI.

$$TDI = \sum GSC \tag{2}$$

Value Adjustment Factor could be calculated from equation 3.

$$VAF = (TDI \times 0.01) + 0.65 \tag{3}$$

Calculate the adjusted function point count

Adjusted Function point count is calculated based on equation 4.

$$\text{Adjusted FP Count} = \text{Unadjusted FP Count} \times \text{VAF} \quad (4)$$

Table 3. Ignored Function Point Cost Factors

	Factor	Description	Reason of Exclusion
1	Transaction rate	The degree of the application performance in the peak time.	This factor is highly related to performance in the traditional function point. As performance is considered, so transaction rate will be ignored.
2	Online data entry	The percentage of the data is entered or retrieved through interactive transactions.	Not Applicable in SOA
3	End-user efficiency	The degree of ease of use for the user of the application.	Not applicable in SOA.
4	Online update	The degree to which internal logical files are updated on-line.	Not applicable in SOA.
5	Operational ease	describes the start-up, backup and recovery procedures of the application	Out of SOA scope

Convert adjusted function point count to effort in man-hours using productivity factor

All the previous steps will lead to the function point count of each service. In this step we will determine the service effort based on the adjusted function point count. In order to convert the function point count to effort in man-hour, we will use the productivity factor.Productivity factor value vary according to the programming language used, the project nature, business domain, etc. In case if the organization has its historical project base counts, this will give an appropriate productivity factor[25].

If historical data doesn't exist, a historical data of the client could be used. If historical data doesn't exist at all, "market productivity" factors will be the only solution. They differ based on the programming language used.

For Java projects they use 12-14 hours per function points, to .Net they use 8-10 hours per function point. In our research we will use 8 hours per function point as the programming language used is .net and no historical data available. The estimated total effort could be calculated from equation 5.

$$\text{Estimated total effort} = \text{total adjusted function points} \times 8 \quad (5)$$

3.3.2. Phased effort estimation approach

This phased effort distribution approach provides a way to estimate the total cost of the service by knowing the cost of one of the early phases (in most cases requirements phase). It also helps project managers in the resource allocation process based on the effort of each phase. The detailed approach steps are as follows :

Identify the estimated effort distribution

The estimated effort distribution among phases is shown in Table 4 [26] .The table shows the phase and the corresponding estimated effort ratio in %.These ratios are reliable and has been applied in real projects in [26].

Get the actual effort of the requirement phase

The actual effort of the requirement phase will lead to estimate the other phase's efforts.

Calculate the estimated effort of the other phases

From Table 4 and the requirement phase it is possible to get the effort of the other phases using equation 6:

Table 4: Estimated Effort Distribution for Traditional New Software

Phase	% estimated effort
Requirements & Analysis	16
Design	15
Development	40
Testing	22
Integration	7
Total Effort	100

Estimated Effort of Phase = (phase estimated effort Percentage * requirement phase actual effort) /requirement phase estimated effort Percentage (6)

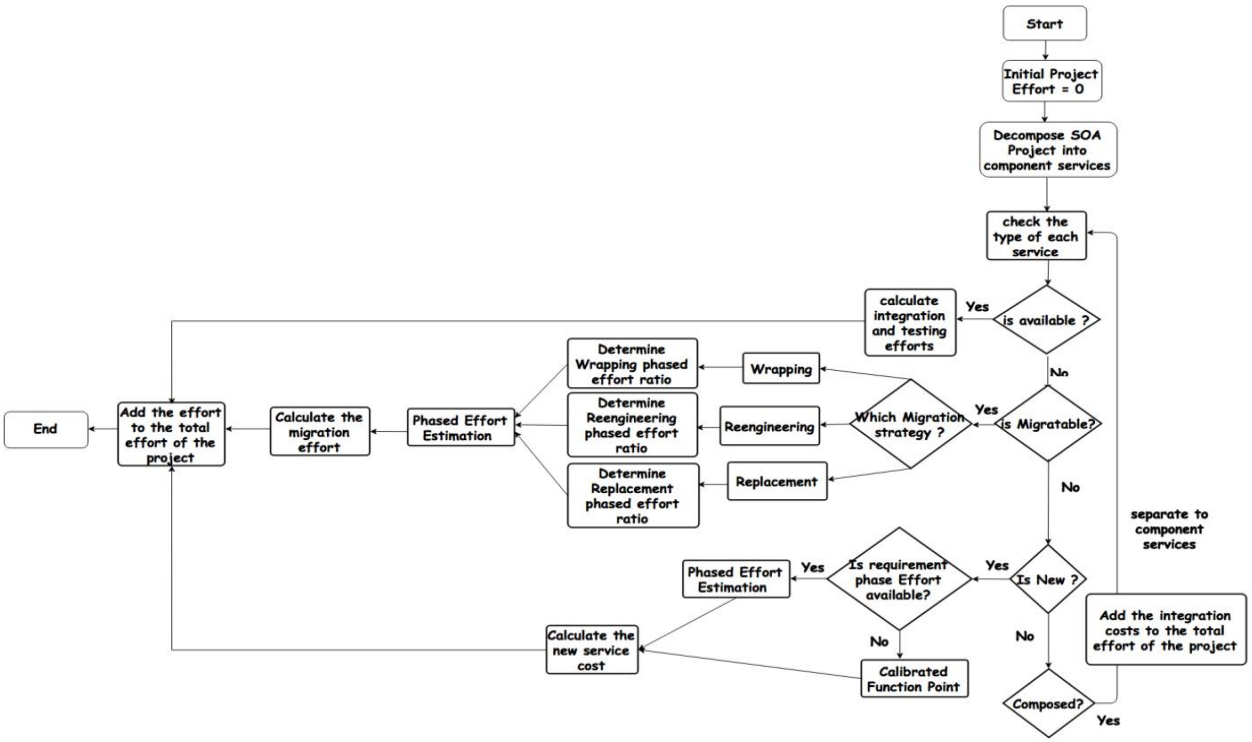

Fig. 2. Overall proposed approach.

3.4. Composed Service

Composed service can be estimated as in Fig. 2. The composed service is broken down to its constituent services and the cost of each of the component service is estimated solely based on its construction type available, migrated or new service. The efforts are aggregated and added to the cost of integrating these services into the composite service.

4. Experiment

In this section, the proposed cost estimation approach for different types of services is applied .The case studies data are for real projects implemented in organization working in the IT industry in Egypt for 15 years. The organizations domain is E-Government. The actual efforts of services are stored in the

organization's internal content management system as efforts of scattered different tasks. These efforts have been accumulated and grouped by phase to get the effort of each phase. The selected projects in the case studies had to contain services, at least one service. All the traditional software projects have been excluded. Unfortunately, there is no project that is composed entirely of services. As this project cost will be too high to be justified to the organization's upper management. Instead we included services used in traditional software projects. These services are either: composed, migrated or new .The available service type is excluded from our research since its development effort is zero. The integration effort is included in each service as a separate phase. Only completed projects at the time of this research have been included in the study. The actual effort includes all the efforts from start till the end of the project. Only projects with documented technical details are included in the study. Such Technical details include project circumstances, cost factors, project size, team size, project duration and technology used. Unfortunately there is lack of the historical projects as there is undocumented data. Projects with lost or incomplete data have been excluded from the study. Table 5 shows the details of the case studies with the projects domain, and the name of each project and the constituent services in the sample. The effort per each phase is required in the study. The total effort of each phase is calculated by summing up all the efforts of the constituent activities. The effort has been measured in Man-hour. The study includes two projects Alpha and Beta .Each project is built as a traditional software, however it includes services . The services included in the study as will be shown in the next subsections.

Table 5. Case Studies Description

Project name	Services	Domain	Total project duration	Project size	Team size (Person)	Project technology
Project Alpha	Customer Name AutoComplete Service	E-Government / financial	11 months	M	3	Asp.Net , Sql Server 2005 , WCF
	Change Password Service				6	
	IntegrationWithCustomer Service				7	
	Client "X" Integration Service				9	
	CalculateTotals Service				5	SilverLight , Sql Server 2005 , WCF
Project Beta	Invoice Service	E-Government / Telecommunications	6 months	M	7	Asp.Net , Sql Server 2005

4.1. Project Alpha

It is a project which is implemented in Egyptian post offices to allow major companies (customers) to insert deposits from post offices. Those deposits are integrated in the customer's internal system. The backend in post offices is website using Asp.net, SQL server database. The project was built as traditional software however; the most reusable parts in the website are either developed as services from scratch or migrated into services. The services are either made using WCF or normal web services. Application of the

proposed approach to the project's services will be detailed next in this section.

Table 6. Auto Complete Unadjusted Function Point

Module	*Function Description*	*Type*	*RET/FTR*	*DET*	*Function Complexity Rating*	*Function Points*
AutoComplete Name	select list of customers based on the first letters typed	ILF	1	2	Low	7
AutoComplete Name	takes the first names of the customer	EI	1	2	Low	3
AutoComplete Name	return list of names	EO	1	2	Low	4
Total Unadjusted Function Point Count						***14***

Table 7. AutoComplete Cost Factors and their Weights

General System Characteristics		Consideration (Y/N)	Degree of Influence	NOTES
1	Data communications	Yes	2	SOAP
2	Distributed data processing	No	0	
3	Performance	Yes	2	moderate performance
4	Heavily used configuration	No	0	
5	Transaction rate	No	0	
6	On-line data entry	No	0	
7	End-user efficiency	No	0	
8	On-line update	No	0	
9	Service complexity	Yes	1	Synchronous , Syntax , Orchestration , Low complexity
10	Reusability	Yes	0	not reusable
11	Installation ease (SOA maturity)	Yes	3	was one of the early SOA projects
12	Operational ease	No	0	
13	Multiple sites	No	0	
14	Flexibility	Yes	2	moderate flexibility
15	Service integration	Yes	1	integrated with only one website
Total Degree of Influence TDI			**11**	

4.1.1. Customer name autocomplete service

Service Description

This service takes the first letters of the customer's name and returns a list of customers whose name starts with these letters.

This service has been developed from scratch and has been modified to enhance its performance. The effort of building the service from scratch and the service migration will be both included in the study.

New Service Estimation: Calibrated Function Point Effort Estimation

The calibrated function point approach estimates the effort based on the functions of the service. The functions of the service and its unadjusted function point are shown in Table 6. The total unadjusted function point is 14.

The cost factors are weighted based on the requirement documentation. The ignored factors are given weight 0. These factors are shown in Table 7 which shows that the TDI = 11.

From equation 3, VAF = (1×0.01) + 0.65 = 0.76.

From equation 4, Adjusted FP count = 10.64

From equation 5, estimated total effort = 85.12 Man-Hour.

The actual effort is 91 Man-Hour.

The relative error is 6.46 % which calculated from equation 6 and shown in Table 8.

$$\text{Relative Error} = (\text{Estimated Effort} - \text{Actual effort})/\text{Actual Effort} \times 100 \quad (6)$$

Table 8. Auto Complete New Service Relative Error

Estimated Total Effort (Man-Hour)	Actual Effort(Man-Hour)	Relative Error
85.12	91	6.46 %

New Service Estimation: Phased Effort Estimation

Phased effort ratios are shown in Table 4. From these ratios the estimated effort for each phase is calculated through equation 7. Note that the requirement phase estimated effort equals the actual effort as the requirement phase effort is not estimated. Table 9 shows the total effort relative error = 17.58%.

Table 9. Autocomplete Service New Service Phased Estimation Results

Phase	% Estimated effort	Estimated effort (Man-Hour)	Actual effort (Man-Hour)	Relative error
Requirements & Analysis	16	12	12	0 (unestimated)
Design	15	11.25	19	-40.79%
Development	40	30	34	-11.76%
Testing	22	16.5	16	3.13%
Integration	7	5.25	10	-47.50%
Total Effort	**100**	**75**	**91**	**17.58%**

Table 10. Migrated AutoComplete Service Estimated and Actual Effort

Phase	Estimated Effort ratios	Actual Efforts (Man-Hour)	Estimated Effort (Man-Hour)	Relative error
Requirements	21.62%	8	8	0 (unestimated)
Design	27.03%	12	10	16.67%
Development	18.92%	8.5	7	17.65%
Testing	18.92%	9	7	22.22%
Implementation	13.51%	7.5	5	33.33%
Total	**100%**	**45**	**37**	**17.78%**

Migrated Service: Phased Effort Estimation

As mentioned earlier this service has been developed from scratch and has been also migrated. In the migration step, many enhancements have been undertaken. These modifications include: performance enhancements by returning only top 10 customers not all the customers and invoked by minimum 3 letters not just one letter. The migration strategy used has been re-engineering.

The estimated effort ratios presented in Table 1 has been used to calculate the estimated effort .Comparison between actual and estimated effort and the error % is shown in Table 10.

4.1.2. Change password service

Service Description

This service enables the user to change his password based on specific password policy. The traditional password policy has been modified to increase the security level. The service has been developed originally using WCF technology. In the migration step, the password policy has been modified and performance enhancement has been undertaken. But unfortunately the data of the new service is not complete. Only the complete data of the migrated service is available. The migration strategy used in this service is re-engineering.

Migrated service Effort Estimation

The migration estimated effort, actual efforts and the relative effort are shown in Table 11 .

Table 11. Change Password Migrated Service Results

Phase	Estimated Effort %	Estimated Effort	Actual Efforts (Man-Hour)	Relative Error
Requirements	21.62%	19	19	0 (unestimated)
Design	27.03%	23.75	25	4.98%
Development	18.92%	16.63	17	2.19%
Testing	18.92%	16.63	19	12.49%
Implementation & Integration	13.51%	11.87	15	20.85%
Total	**100%**	**87.88**	**95**	**7.49%**

4.1.3. Integration with customer service

Service Description

This service main functionality is to integrate with the customer. It has to send customers their data of interest in the required format. The service data are then inserted in the customer's temporary database. The customer invokes this web service, to get the data. The service has been built using WCF technology .The service main functionality is to select the data from SQL server 2005 database. This service was also developed from scratch however the data of the new service is not complete. The modifications of the functionality included adding new database fields, changing in the format of the data and enhancing the performance. The migration strategy used is re-engineering. The migrated service data is complete so in our experiment, only the migrated service will be included.

Migrated Service Effort Estimation

The results of the migrated service are shown in Table 12.

Table 12. Integration with Customer Service Migrated Effort Estimation

Phase	Estimated Effort %	Estimated Effort (Man-Hour)	Actual Efforts (Man-Hour)	Relative Error
Requirements	21.62%	22	22	0 Unstimated
Design	27.03%	27.51	32	14.05%
Development	18.92%	19.25	25	22.99%
Testing	18.92%	19.25	23	16.29%
Implementation & Integration	13.51%	13.75	16	14.08%
Total	**100%**	**101.76**	**118**	**13.76%**

4.1.4. Client "X" integration service

Service Description

This service main functionality is to transfer the data from a temporary database to the customer's internal system, which in this case was SAP system. The integration is achieved via web service. This service has been built to satisfy a specific customer's needs.

New Service Estimation: Calibrated Function Point Effort Estimation

As this service has been built from scratch to satisfy special customer's needs, the calibrated function point results are shown in Table 13. As shown the relative error is 14.59 %.

Table 13 : Client "X" Integration New Service: Calibrated Function Point Effort Estimation Results

Estimated total effort(Man-Hour)	Actual effort (Man-Hour)	Relative error
216.08	253	14.59%

New Service Estimation: Phased Effort Estimation

The phased effort estimation results are shown in Table 14 . The relative error of the total service is 3.66%

Table 14 :Client "X" Integration New Service: Phased Effort Estimation Results

Phase	% Estimated effort	Estimated effort (Man-Hour)	Actual effort (Man-Hour)	Relative error
Requirements & Analysis	16%	39	39	0 (unestimated)
Design	15%	36.5625	42	12.95%
Development	40%	97.5	89	9.55%
Testing	22%	53.625	57	5.92%
Integration	7%	17.0625	26	34.38%
Total	100%	243.75	253	3.66%

4.1.5. Calculate totals service

Service Description

This service has been developed from scratch. This service main function is to make some calculation (summation, average, count) of fields in transactions table in the Alpha project. These totals are to be displayed in a web pages using Silverlight. The technology used is Silverlight, WCF services, SQL server 2005.

New Service Estimation: Calibrated Function Point Effort Estimation

Table 15 shows the estimated effort, the actual effort and the relative error.

Table 15. Totals Service Calibrated Function Point Estimates and Relative Error

estimated Total Effort (Man-Hour)	Actual Effort (Man-Hour)	Relative Error
192.56	224	14.04%

New service Estimation: Phased Effort Estimation

Table 16 shows the totals service phased effort with total error relative error 19.08%.

4.2. Project Beta

This project was built in order to provide the facility to the citizens to pay their bills from the post offices. The system is integrated with the biller's internal system in inquiry of the bill and paying the bill. The backend in post offices is website using Asp.net, SQL server 2005 database. The project is built as

traditional software however; the most reusable parts in the website are either developed from scratch or migrated into web services. The services are either made using WCF or normal web services. Unfortunately there is lack in documentation in this project and incomplete data about the effort spent in this project. However, only one service has complete data which is included in our study.

Table 16 . Totals Service Phased Effort Ratio Results

Phase	% estimated effort	Estimated Effort (Man-Hour)	Actual Effort (Man-Hour)	% error
Requirements & Analysis	16%	29.00	34	0(un estimated)
Design	15%	27.19	42	35.27%
Development	40%	72.50	81	10.49%
Testing	22%	39.88	45	11.39%
Integration	7%	12.69	22	42.33%
Total	100%	181.25	224	19.08%

4.2.1. Invoice service

Service Description

This service has been built from scratch. Its main functionality is to inquiry about the bill status either paid or not, and ability to pay the invoice.

New Service Estimation: Calibrated Function Point Effort Estimation

Relative error shown in Table 17

Table 17. Relative Error Invoice Service

Estimated total effort	Actual effort	%Error
265.68	223	19.14%

New service Estimation: Phased Effort Estimation

The actual and estimated phased efforts are shown in Table 18. The total service relative error is 7.51.

Table 18. Invoice Service, New Service Phased Effort estimation Results

Phase	% estimated effort	Estimated Effort (Man-Hour)	Actual Effort (Man-Hour)	Relative Error
Requirements & Analysis	16%	33.00	33	0 (unestimated)
Design	15%	30.94	41	24.54%
Development	40%	82.50	87	5.17%
Testing	22%	45.38	39	16.35%
Integration	7%	14.44	23	37.23%
Total	100%	206.25	223	7.51%

Table 19. Accumulation of the Results of the Case Study

Project Name	Service name	Type of Service	Estimation Approach Used	Estimated effort (Man-Hour)	Actual Effort (Man-Hour)	Relative Error

Project Alpha	Customer Name AutoComplete	New	Calibrated Function Point	85.12	91	6.46%
			Phased Effort Estimation	75	91	17.58%
		Migrated	Phased Effort Estimation	37	45	17.78%
	Change Password	Migrated	Phased Effort Estimation	87.88	95	7.49%
	IntegrationWithCustomer	Migrated	Phased Effort Estimation	101.76	118	13.76%
	Client "X" Integration	New	Calibrated Function Point	216.08	253	14.59%
			Phased Effort Estimation	243.75	253	3.66%
	CalculateTotals	New	Calibrated Function Point	192.56	224	14.04%
			Phased Effort Estimation	181.25	224	19.08%
Project Beta	Invoice Service	New	Calibrated Function Point	265.68	223	19.14%
			Phased Effort Estimation	206.25	223	7.51%

5. Conclusion

This paper proposes an SOA effort estimation approach which is based on classifying the services into its basic type and estimate the effort considering the cost factors related to this service type. The SOA project is broken down into its component services. Each service is classified into either: available, migrated, new or composed. The available service development effort is zero, as the main effort exists in the integration and testing. Migrated service effort estimation is based on the migration strategy either wrapping, re-engineering or replacement. The cost factors are weighted and distributed into phases. Phased effort distribution enables the decision makers to choose which migration strategy could be used. The phased effort ratio which has been applied and the relative error ranged from 7.49% to 17.78%. However it gives more accurate results while estimating each phase solely.

While the new service has been estimated using two approaches calibrated function point and phased effort ratio estimation. The calibrated function point considers the SOA characteristics ignored by the traditional function point. The calibrated function point relative error ranged from 3.66% to 19.14%. The phased effort ratio approach relative error ranged from 3.66% to 19.08%. The overall results are shown in Table 19. The overall phased effort results are better than the calibrated function point .However the calibrated function point estimates the effort in the early stages of the project. While phased effort estimation estimates the next phases by knowing the effort of the requirement phase or at least by using expert opinion method.

5.1. Limitations of the study (Threats to Validity)

There are some factors that may affect the validity of the results:

- The sample projects were from the same organization and the same business domain.
- The projects had the same technology (.Net , SQL server)
- The data available are from 2 projects only
- Only migration strategy available data is re-engineering. There were replacement case studies however the data is either incomplete or unavailable. Also the wrapping strategy data is available but it wasn't implemented using SOA, it was implemented using traditional software.
- Lack of projects documentation and history has prevented further analysis for the case studies.

5.2. Future work

We believe that there is a significant room for improvement by applying this approach to several domains with data collected worldwide. In order to get phased effort estimation for each domain, as well as detailed estimation of composed services.

References

[1] Abrams, C., & Schulte, R. W. (2008). Service-oriented architecture overview and guide to SOA research. *Gartner Research*.

[2] Bajwa, I. S., Kazmi, R., Mumtaz, S., Choudhary, M. A., & Naweed, M. S. (2008). SOA and BPM partnership: a paradigm for dynamic and flexible process and IT management. *World Academy of Science, Engineering and Technology*, *45(4)*, 16-22.

[3] Krafzig, D., Banke, K., & Slama, D. (2005). Enterprise SOA: Service-oriented architecture best practices. Prentice Hall Professional.

[4] Linthicum, D. (2006). How Much Will Your SOA Cost? Here are Some Guidelines. Retrieved, from http://soa. sys-con. com/node/318452

[5] Erl, T. (2005). Service-oriented architecture: concepts, technology, and design. *Pearson Education India*.

[6] Farrag, E. A., & Moawad, R. (2014). Phased effort estimation of legacy systems migration to service oriented architecture. *International Journal of Computer and Information Technology*, *3(3)*.

[7] Seth, A., Agrawal, H., & Singla, A. R. (2014). Techniques for evaluating service oriented systems: A Comparative Study. *Journal of Industrial and Intelligent Information*, *2(2)*.

[8] Lewis, G., Morris, E., & Smith, D. (2005). Service-oriented migration and reuse technique (smart). *Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice* (pp. 222-229).

[9] Li, Z., & Keung, J. (2010, June). Software cost estimation framework for service-oriented architecture systems using divide-and-conquer approach. *Proceedings of the2010 Fifth IEEE International Symposium on Service Oriented System Engineering* (pp. 47-54).

[10] Merlo–Schett, N. (2002). COCOMO (Constructive cost model). *Seminar on Software Cost Estimation WS* .

[11] Sunkle, S., & Kulkarni, V. (2012). *Cost Estimation for Model-Driven Engineering*. Springer Berlin Heidelberg.

[12] Tansey, B., & Stroulia, E. (2007). Valuating software service development: Integrating COCOMO II and real options theory. *Proceedings of the First International Workshop on The Economics of Software and Computation*.

[13] Timp, A. (2005). IFPUG function point counting practices manual, release 4.2. *International Function Point Users Group*.

[14] Grupe, F. H., & Clevenger, D. F. (1991). Using function point analysis as a software development tool. *Journal of Systems Management*, *42(12)*.

[15] Mahmood, K., Ilahi, M. M., Ahmad, B., & Ahmad, S. (2012). Empirical analysis of function points in service oriented architecture (SOA) applications. *Industrial Engineering Letters*, *2(1)*, 6-12.

[16] Santillo, L. (2007). Seizing and sizing SOA applications with COSMIC function points. *Proceedings of SMEF*.

[17] Rivas, G. D., & Quintana, M. Estimation in service oriented architecture.

[18] Mahmood, K., Ilahi, M. M., Ahmad, S., & Ahmad, B. (2011). Integration efforts estimation in service oriented architecture (SOA) applications. *Information and Knowledge Management*.

[19] Li, Z., & O'Brien, L. (2011). A qualitative approach to effort judgment for web service composition

based SOA implementations. *Proceedings of the 2011 IEEE International Conference on Advanced Information Networking and Applications* (pp. 586-593).

[20] Almonaies, A. A., Cordy, J. R., & Dean, T. R. (2010). Legacy system evolution towards service-oriented architecture. *International Workshop on SOA Migration and Evolution*.

[21] Khadka, R., Saeidi, A., Jansen, S., & Hage, J. (2013). A structured legacy to SOA migration process and its evaluation in practice. *Proceedings of the2013 IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems*.

[22] Canfora, G., & Di Penta, M. (2006). SOA testing technologies further reading web services-interoperability.

[23] Youssef, H. (2012). Method for the scoping and sizing of service oriented systems.

[24] Alexander, A. (2004). How to determine your application size using function points.

[25] Games, Y. M. P. (2012). Functional size, effort and cost of SOA projects with function points.

[26] Yang, Y., He, M., Li, M., Wang, Q., & Boehm, B. (2008). Phase distribution of software development effort. *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 61-69).

Esraa A. Farrag was born in 1986, in Alexandria, Egypt. She has graduated from Faculty of Science, Alexandria University in 2007. She has graduated from ITI (information technology institute) in 2008. She started working in the IT industry as a software developer in July 2008 till current. Her research interests are mainly service-oriented architecture and software engineering.

Farrag was the ideal student in Faculty of Science in Alexandria University in 2007. A lso she took the title of the Ideal Volunteer from Resala Charity Organization in May 2013. Esraa is currently working as a development team leader in ITSC in Cairo, Egypt, and preparing her master degree in arab academy for science and technology plus her non-governmental activity in resala charity organization.

Ramadan Moawad obtained his B.Sc from Military Technical College in electric engineering and M.Sc degree from Military Technical College in computer engineering. He obtained his PhD from ENSAE College, France in software engineering. He taught several courses in CS and CE in several institutions including the American University in Cairo, the Military Technical College, Cairo Uinversity and the Arab Academy for Science, Technology and Maritime Transport. He joined Future University in 2011 and currently working as the vice-dean of FCIT. He published over 50 papers in different journals and conferences locally and internationally. His research interest is software engineering and software quality assurance. He has refereed several papers in IEEE Transactions in Software Engineering journal and the international journal of software Engineering (IJSE).

Ibrahim F. Imam received his B.Sc. in mathematics and statistics in 1986 and a graduate diploma in computer science and information in 1989 from Cairo University. He received his M.Sc. in computer science in 1992 and his Ph.D. in information technology and engineering in 1995 from George Mason University.

Dr. Imam edited three international books and several international journals. He chaired three international conferences and workshops. He authored co-authored over 60 papers in refereed journals, conference proceedings, and workshop proceedings. His

research focused in the fields of artificial intelligence, data mining, text mining (Arabic & English), pattern recognition, and machine learning. Dr. Imam is a steering committee member of the international journal of artificial intelligence and machine learning. He served as program committee member for many international conferences.

Dr. Imam's research in arabic text mining is well recognized. He stared along with his students a novel approach for text mining (A presentation is available up on request). He and his students developed a database containing more than 15 tables, each covers over 12 million diacritic Arabic words.

Dr. Imam's research in image processing and pattern recognition stared in 1995. He introduced a break through approach for recognizing faces in 1996. The approach is published in the top ranked conference and journal all over the world.