Global Journal of Computer Science and Technology
Volume 11 Issue 15 Version 1.0 September 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

A Framework for Costing Service-Oriented Architecture (SOA) Projects Using Work Breakdown Structure (WBS) Approach

By Yusuf Lateef Oladimeji, Olusegun Folorunso, Akinwale Adio Taofeek, Adejumobi, A. I

University of Agriculture, Abeokuta, Ogun State, Nigeria

Abstract - With end users demanding faster response time and management demanding lower costs and more flexibility, Service Oriented Architecture (SOA) projects are becoming more complex and brittle. Proper costing and identification of feasible benefits of SOA projects are quickly becoming a significant influence in the mainstream of all industries. SOA is intended to improve software interoperability by exposing dynamic applications as services. Current SOA quality metrics pay little attention to service complexity as an important key design feature that impacts other internal SOA quality attributes. Due to this complexity of SOA, cost and effort estimation for SOA-based software development is more difficult than that of traditional software development. Unfortunately, there is little or no effort about cost and effort estimation for SOA-based software. Traditional software cost estimation approaches are inadequate to address the complex service-oriented systems. Although numerous sources expound on the technical advantages of SOA as well as listing praises for their intuitive and qualitative benefits, until now no one has provided a reliable and quantifiable result from SOA implementations currently in production. This paper proposes a novel framework based on Work Breakdown Structure (WBS) approach for cost estimation of SOA-based software by dealing separately with service parts. The WBS framework can help organizations simplify and regulate SOA implementation cost estimation by explicit identification of SOA-specific tasks in the WBS. Furthermore, both cost estimation modelling and software sizing work can be satisfied respectively by switching the corresponding metrics within this framework. We provide an example case study to demonstrate proposed metrics and we also investigate the benefit of SOA to its adopters.

Keywords : *Service-Oriented Architecture (SOA), Software Cost Estimation, Work Breakdown Structure (WBS), Framework, Return on Investment (ROI).*

GJCST Classification : *D.2.9, D.2.8*

A Framework for Costing Service-Oriented Architecture (SOA) Projects Using Work Breakdown Structure (WBS) Approach

Yusuf, Lateef Oladimeji[α], Olusegun Folorunso[Ω], Akinwale, Adio Taofeek[β], Adejumobi, A. I[ψ]

Abstract - With end users demanding faster response time and management demanding lower costs and more flexibility, Service Oriented Architecture (SOA) projects are becoming more complex and brittle. Proper costing and identification of feasible benefits of SOA projects are quickly becoming a significant influence in the mainstream of all industries. SOA is intended to improve software interoperability by exposing dynamic applications as services. Current SOA quality metrics pay little attention to service complexity as an important key design feature that impacts other internal SOA quality attributes. Due to this complexity of SOA, cost and effort estimation for SOA-based software development is more difficult than that of traditional software development. Unfortunately, there is little or no effort about cost and effort estimation for SOA-based software. Traditional software cost estimation approaches are inadequate to address the complex service-oriented systems. Although numerous sources expound on the technical advantages of SOA as well as listing praises for their intuitive and qualitative benefits, until now no one has provided a reliable and quantifiable result from SOA implementations currently in production. This paper proposes a novel framework based on Work Breakdown Structure (WBS) approach for cost estimation of SOA-based software by dealing separately with service parts. The WBS framework can help organizations simplify and regulate SOA implementation cost estimation by explicit identification of SOA-specific tasks in the WBS. Furthermore, both cost estimation modelling and software sizing work can be satisfied respectively by switching the corresponding metrics within this framework. We provide an example case study to demonstrate proposed metrics and we also investigate the benefit of SOA to its adopters.

Keywords : Service-Oriented Architecture (SOA), Software Cost Estimation, Work Breakdown Structure (WBS), Framework, Return on Investment (ROI).

Author [α] : Department of Computer Science University of Agriculture, Abeokuta, Ogun State, Nigeria GSM Phone no : +234 803 403 7098; E-mail : truevisionconsulting@yahoo.com
Author [Ω] : Department of Computer Science University of Agriculture, Abeokuta, Ogun State, Nigeria GSM Phone no : +234 803 564 0707; E-mail : folorunsolusegun@yahoo.com
Author [β] : Department of Computer Science University of Agriculture Abeokuta, Ogun State, Nigeria GSM Phone no : +234 806 286 7612; E-mail : atakinwale@yahoo.com
Author[ψ] : Department of Electrical and Electronics Engineering University of Agriculture Abeokuta, Ogun State, Nigeria GSM Phone no: +234 703 321 5455; E-mail : engradejumobi@yahoo.com

I. Introduction

A well-developed understanding of the Return on Investment (ROI) for SOA has been a complex undertaking [1]. This is due in part to the deficiency in comprehensive historical data on which to base any such model. For the most part, SOA often exist as pilot projects than as full-blown production systems, and even those rare production-quality systems that do exist are too new for use in understanding critical issues to the ROI equations, such as reusability and redeployment. Most organizations that want to build an SOA don't have a clue on how to approach the cost estimation process. In many cases, they grossly underestimate the cost of their SOA, hoping the management won't notice, this is done to get approval and reveal the higher costs later after investment may have been made and too late to go back. This is not a good management practice. The other problem militating against building a comprehensive cost model for SOA is the need to separate the service cost that results in SOA from any well-designed application and the specific or incremental cost that obtains from a well designed SOA application built on services architecture. Software cost estimation for Service-Oriented Architecture (SOA) development confronts more challenges than for traditional software development. One of the main reasons is the architectural difference in SOA compared to traditional software development. Josuttis [2] has pointed out that distributed processing would be inevitably more complicated than non-distributed processing, and any form of loose coupling will increase complexity. Meanwhile, the more complexity involved in a system, the more difficulty the designers or engineers have to understand the implementation process and thus the system itself [3]. In other words, people have to devote more effort to accurate manipulations when performing more complicated tasks. In practice, building a true heterogeneous SOA for a wide range of operating environments may take years of development time if the company does not have sufficient SOA experience and expertise [4]. It is difficult to foresee and justify the cost and effort of developing an SOA application before the project starts. The problem of SOA cost estimation has not been addressed adequately in the existing literature.

The current cost estimation approaches for traditional software development are inadequate for complex service-oriented software. For example, COCOMO II cannot arrive at global cost approximation for the entire SOA application development, and expert judgment may easily fall into traps of uncertainty or bias because of the complexity of the SOA. This paper proposes a novel framework by employing a Work Breakdown Structure (WBS) approach in an attempt to deal with cost estimation problem for SOA based software development. Within this WBS framework, services are classified into three primitive types and one combined type according to different development processes. Cost estimation for developing primitive services can be handled as sub-problems that are small and independent enough to be solved. For combined services, the division procedure will emerge recursively until all the resulting separated services are primitive. The cost and effort of service integration is then calculated gradually following the reverse division sequence. The application of the WBS cost estimation framework is demonstrated using a case study. The result shows that the proposed framework can simplify and regulate the complicated development cost estimation for SOA-based applications. The business goals and objectives of SOA are to increase agility and reduce costs while the technical goals and objectives are to increase usability, improve maintainability and reduce redundancy [5]. SOA hold out the promise for a brave new world of applications development, deployment, and reuse that many proponents believe will usher in unprecedented levels of Return on Investment (ROI) for a domain that has long suffered from cost overruns and excessive, often unjustified expenditures. The ability to lower the cost of integration while improving the leveragability of key software and business process assets are only a few of the reasons why the ROI of service-oriented architectures and composite applications is thought to herald a new economic reality for IT and business development. Ease of use and lower training costs, lower cost of deployment, faster time to market, improved business requirement matching, and better multi-channel deployment are among the myriad reasons the technologies are so eagerly awaited by business and IT managers alike.

II. Related Work

a) SOA Services

SOA is a collection of services with well-defined interfaces and a shared communications model. A service is a coarse-grained, discoverable, and self contained software entity that interacts with applications and other services through a loosely coupled, often asynchronous, message-based communication model [6]. A system or application is designed and implemented to make use of these services. This developed capability may itself provide services within the overall SOA. The underlying idea of SOA is that it would be cheaper and faster to build or modify applications by composing them out of limited-purpose components that can communicate with each other because the components strictly adhere to interface rules [7]. The advent of the Internet and World Wide Web (WWW) introduced a new wave of research on collaborative product development environment [8][9][10][11][12]. Yusuf et al., [13] Observed that the Internet is no longer a simple network of computers but a network of potential services in which the functional views of services need to be clearly defined during the design of an Internet-based distributed engineering system. The most common form of SOA is that of Web services in which all of the following apply: service interfaces are described using Web Services Description Language (WSSL), payload is transmitted using Simple Object Access Protocol (SOAP) over Hypertext Transfer Protocol (HTTP), and Universal Description, Discovery and Integration (UDDI) is optionally used as the directory service [14]. However, WSSL, SOAP, and HTTP are not the only foundation on which an SOA can be built. Other technologies such as CORBA and IBM's Web sphere can be used as part of the messaging backbone of an SOA.

b) Work Breakdown Structure

A Work Breakdown Structure (WBS) is a hierarchical decomposition (tree structure) of the work required to accomplish a goal. It is developed by starting with the end objective and successively re-dividing it into manageable components in terms of size, duration, and responsibility [15]. However, it is often done as a modification of an existing WBS for a similar project. It is an essential starting input to both estimation and to scheduling. In essence, it provides the chart of accounts for a project. To know what something cost, it needs to exist as a task in the WBS. In large projects, the approach is quite complex and can be as much as five or six levels deep. Usually, items at the same level of hierarchy are in the order they are executed, although this is not required. Traditionally, definition of the WBS is left to vendors with the integrated master schedule and price proposal based upon it included as part of the RFP response. More often than not, the organization of the WBS in software development follows the traditional "waterfall" method of system development. The primary constraint is that the WBS fulfils the requirements of the statement of work [16]. Since the development of the software within services is about the same as traditional development, we suggest Breakdown of SOA into Services from a WBS perspective.

c) COCOMO II

COCOMO II (Constructive Cost Model) [17] is one of the best-known and best-documented algorithmic models, which allows organizations to estimate cost, effort, and schedule when planning new software development activities. Tansey and Stroulia [18] have attempted to use COCOMO II to estimate the cost of creating and migrating services. They reported that COCOMO II should be extended to accommodate new characteristics of SOA based development. COCOMO II is generally inadequate to accommodate the cost estimation needs for SOA-based software development. When considering the declarative composition specifications, a fundamentally different development process may be adopted in SOA-based software. Based on the Internet technologies, SOA-based software can be realized as a composition of loosely coupled services with well-defined interfaces and consistent communication protocols. These services hide technical details, and are not restricted to any specific technology. In other words, the service implementation is programming language and platform independent. Therefore, an SOA-based application could comprise the combination of all possible development strategies and development processes. Consequently, although the COCOMO II model has a large number of coefficients such as effort multipliers and scale factors, it is difficult to directly justify the cost estimation for SOA-based software development. On the other hand, considering the difference between component orientation and service orientation [19], the COCOMO II model by itself is inadequate to estimate effort required when reusing service-oriented resources. COCOMO II considers two types of reused components, namely black-box components and white-box components. Black-box components can be reused without knowing the detailed code or making any change to it, while white-box components have to be modified with new code or integrated with other reused components before it can be reused. Similarly, within the SOA framework, there are black-box services that can be adopted directly, and white-box services that should be ported from legacy systems. Nevertheless, taking black-box reuse for instance, the difference between code-level and service-level reuse is significant. Whether a code-level component is suitable or not for reuse should be understood and revealed by using reverse engineering or reengineering [20] according to the real situation. Comparatively, the contractually reusable and loosely coupled service can be reused directly through service discovery techniques, for example semantic annotation and quality of service.

d) Function Point Analysis and Software Sizing

Size prediction for the constructed deliverables has been identified as one of the key elements in any software project estimation. SLOC (Source Line of Code) and Function Point are the two predominant sizing measures. Function Point measures software system size through quantifying the amount of functionality provided to the user in terms of the number of inputs, outputs, inquires, and files. In practice, Function Point can be used continuously throughout the entire software development life cycle, which provides the essential value of what the software is and what it does with data from the user's viewpoint. Santillo attempts to use the Function Point method to measure software size in an SOA environment [21]. After comparing the effect of adopting the first and second generation methods, that is the International Function Point User's Group (IFPUG) and Common Software Measurement International Consortium (COSMIC) respectively, Santillo identifies several critical issues. The prominent one is that SOA is functionally different from traditional software architectures, because the "function" of a service should represent a real-world self-contained business activity [2]. More issues appear when applying IFPUG to software system size measurement. For example, the effort of wrapping legacy code and data to work as services cannot be assigned to any functional size. Measuring with the COSMIC approach, on the contrary, is supposed to satisfy the typical sizing aspects of SOA-based software. However, there is a lack of guidelines for practical application of COSMIC measurement in SOA context. In addition to the application of Function Points, Liu et al. [22] use Service Points to measure the size of SOA-based software. The software size estimation is based on the sum of the sizes of each service.

$$Size = \sum_{i=1}^{n} (P_i \times P)$$

Where P_i is an infrastructure factor with empirical value that is related to the supporting infrastructure, technology and governance processes; P represents a single specific service's estimated size that varies with different service types, including existing service, service built from existing resources, and service built from scratch. This approach implies that the size of a service-oriented application depends significantly on the service type. However, the calculation of P for various services is not discussed in detail.

e) SMART and SMAT-AUS Framework

The "Service-Oriented Migration and Reuse Technique" (SMART) was developed to assist organizations in analyzing legacy capabilities for use as services in an SOA. SMART was derived from the Options Analysis for Reengineering (OAR) method developed at the SEI that was successfully used to support analysis of reuse potential for legacy components [23]. SMART gathers a wide range of information about legacy components, the target SOA, and potential services to produce a service migration strategy as its primary product. However, SMART also produces other outputs that are useful to an organization whether or not it decides on migration.

Information-gathering activities are directed by the Service Migration Interview Guide (SMIG). The SMIG contains questions that directly address the gap between the existing and target architecture, design, and code, as well as questions concerning issues that must be addressed in service migration efforts. Use of the SMIG assures broad and consistent coverage of the factors that influence the cost, effort, and risk involved in migration to services. Unlike SMART, SMAT-AUS [24] is a framework that is developed to determine the scope and estimate cost and effort for SOA projects. This framework reveals not only technical dimension but also social, cultural, and organizational dimensions of SOA implementation. When applying the SMAT-AUS framework to SOA-based software development, Service Mining, Service Development, Service Integration and SOA Application Development are classified as separate SOA project types. For each SOA project type, a set of methods, templates and cost models and functions are used to support the cost and effort estimation work for each project time which are then used to generate the overall cost of an SOA project (a combination of one or more of the project types). Except for the SMART (Software Engineering Institute's Service Migration and Reuse Technique) method [25] that can be adopted for service mining cost estimation, currently there are no other metrics suitable for the different projects beneath the SMAT-AUS framework. Instead, some abstract cost-estimation-discussions related to aforementioned project types can be found through a literature review. Umar and Zordan [26] warn that both gradual and sudden migration would be expensive and risky so that costs and benefits must be carefully weighed. Bosworth [27] gives a full consideration about complexity and cost when developing Web services. Liu et al. [22] directly suggest that traditional methods can be used to estimate the cost of building services from scratch. Since utilizing solutions based on interoperable services is part of service-oriented integration (SOI) and results in an SOI structure, Erl [28] gives a bottom line of effort and cost estimation for cross-application integration: "The cost and effort of cross-application integration is significantly lowered when applications being integrated are SOA-compliant." A generic SOA application could be sophisticated. But this can be handled in SMAT-AUS by breaking the problem into more manageable pieces (i.e. a combination of project types) however specifying how all of these pieces are estimated and the procedure required for practical estimation of software development cost for SOA-based systems is still being developed.

f) ROI of SOA Based on Traditional Component Reuse

Barry Boehm provided two useful formulas when estimating the cost of software systems reuse. One formula is from the provider's point of view, while the other is from consumer's [29]; Provider-focused formula:

Relative Cost of Writing for Reuse (RCWR) = Cost of Developing Reusable Asset / Cost of Developing Single-User Asset

Consumer's formula : Relative Cost of Reuse (RCR) = Cost of Reuse Asset / Cost of Develop Asset from Scratch

Poulin Jeffery [30] examined large-scale SOA service providers to estimate the value ranges for these formulas in practice. His data shows that RCWR ranges between 1.15 and 2.0 with median of 1.2, while RCR ranges between 0.15 and 0.80 with a median of 0.50. In other words, Paulin work suggests that creating reusable software component for a broad audience takes more resources (15% to 100% more) than creating a less generic point solution. The 20% of the total cost of development directed towards reuse, a factor Poulin calls Relative Cost of Reuse (RCR) would represent an impressive number in the pre-object-oriented development world, but in the world of service oriented architectures and component application; it is believed that 80% is a more accurate figure. This ability to reuse the majority of the software development by an organization is one of the key attributes of SOA development, and while the number will vary greatly from one development organization to another, it is our believe that the early adopters will see this or an even greater degree of reuse simply because initial SOA development will target precisely those applications and business processes that have the greater reuse potential. Mili et al., [31] has published a variety of RCWR factor values that have been developed since the early 1990's based on the experiences of a number of sources. Discussion about cost estimation for SOA implementation also appears in industry. Linthicum [32] outlines some general guidelines for estimating the cost of an SOA application. According to these guidelines, the calculation of SOA cost can be expressed as a sum of several cost analysis procedures.

Cost of SOA = (Cost of Data Complexity
+ Cost of Service Complexity
+Cost of Process Complexity
+ Enabling Technology Solution)

Furthermore, Linthicum also provides some detailed specification. For example, the basic element Complexity of the Data Storage Technology is figured as a percentage between 0% and 100% (Relational is 30%, Object-Oriented is 60%, and ISAM is 80%). Nevertheless, the other aspects of the calculation are suggested to follow similar means without clarifying essential matters. Meanwhile, Linthicum reminds that the notable problem is that this approach is not a real metric. Additionally, SOA based software is inevitably more complicated than traditional software [2]. It is therefore doubtful that Data Complexity, System

Complexity, Service Complexity and Process Complexity are sufficient to represent the complexity of SOA-based systems. As shown, both academia and industry have published little work relating to estimating costs for SOA-based software. In particular, there is not a solution to satisfy the development cost estimation for SOA-based software. We attempt to address these issues by providing a SOA cost estimation framework in this paper.

III. Methodology

a) *SOA-Based Software Cost Estimation Using Work Breakdown Structure (WBS) Approach*

WBS approach is a "Division of labour" or "Divide and conquered" method which can be traced back to as early as 200BC [33], when the Babylonian reciprocal table of Inakibit-Anu was used to facilitate searching and sorting numerical values. However, the first description of the divide and conquered algorithm appears in John Mauchly's article discussing its application in computer sorting [33]. Nowadays, the approach is applied widely in areas such as Parallel Computing [34], Clustering Computing [35], Granular Computing [36], and Huge Data Mining [37]. The principle underlying WBS is shown in Figure 1. That is to recursively decompose SOA into sub problems (services) until all the sub-problems are sufficiently simple enough, and then to solve the sub-problems (cost the services). Resulting solutions (costs) are then recomposed to form an overall solution. Adopting this principle will lead to different subroutines for different sub-problems. Normally, some or all of the sub-problems are of the same type as the input problem, thus WBS procedure can be naturally expressed recursively. The QuickSort [33] algorithm has such procedure.

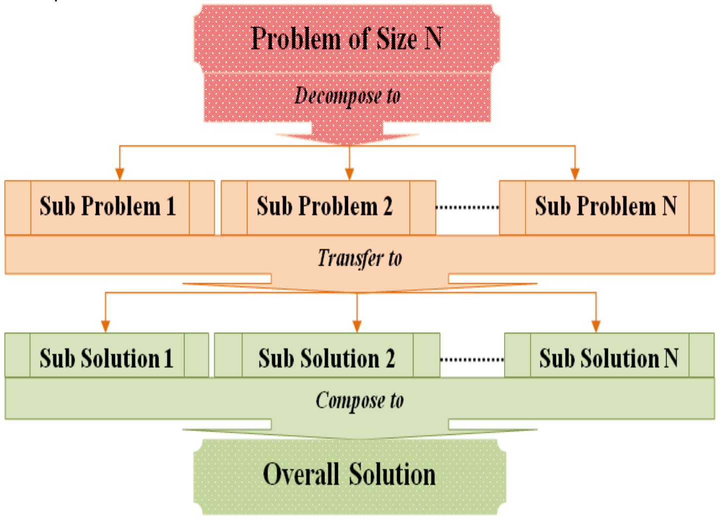

Figure 1: Principle of Work Breakdown Structure (WBS)

The advantages of applying WBS approach to SOA problems are numerous, and can be classified as followings:

- *Structural Simplicity* : Profiting from perhaps the simplest structuring technique, WBS is a high priority strategy to resolve problems not only in the SOA field but also in computing generally, politics and sociology fields. No matter where the approach is applied the solution structure can be expressed explicitly in a program-like function such as: Solution(x) is equivalent to:

```
IF IsBase(x)
Then SolveDirectly(x)
Else Compose(Solution(Decompose(x)))
```

Where x is the original problem that will be solved through *Solution* procedure, *IsBase* is used to verify whether the problem x is primitive or not, which returns TRUE if x is a basic problem unit, or FALSE otherwise. *Solve Directly* presents the conquer procedure. *Decompose* is referred to as the decomposing operation, while *Compose* is referred to as the composing operation

- *Computational Efficiency* : WBS can be used for designing fast algorithms. In appropriate application scenarios, the approach leads to asymptotically optimal cost for solving the problems. A problem of size N can be broken into a bounded number P of sub-problems of size N/P step by step, and all the basic sub-problems have constant-bounded size. Then the algorithm will have O($N\log N$) worst-case program execution performance. Normally, the consequence is more flexible because the size and the number of tasks can be decided at run-time.
- *Parallelism* : Since sub-problems in the individual division stage are logically and physically independent, the WBS approach can be naturally executed in parallel procedures. For computing problems, it is suitable for application in parallel machines due not only to the independent problem grains but also the efficient use of cache and deep memory hierarchies [38]. In fact, it has been considered as one of the well-known parallel programming paradigms.
- *Capability of Solving Complexity* : Through Dismantle of an overall goal into smaller and independent sub-problems, the SS strategy can provide adaptation scalability and variability, and can be used in the areas of engineering to reduce and manage complexity. Those complicated cases, such as resolutions for conceptually difficult problems, and approximate algorithms for NP-hard problems, are usually based on the divide and conquer principle. Given these merits, WBS can be considered a suitable and effective approach to accommodate complex problems such as cost estimation for SOA-based software development, where individual measures must be carried out independently. The following sections discuss its applications in SOA cost estimation.

b) *Service Classification*

Implementing SOA could be complex and onerous, while complexity measurement for SOA-based system is still an open question [39]. Chaos [40] claims that the complexity is restricting some SOA implementations. For the same reason, there are also many challenges to estimate the cost and effort of SOA-based software development. Fortunately, the major advantages of SOA are mainly reusability and composability with an emphasis on extensibility and flexibility, at a high level of granularity and abstraction. In other words, SOA-based software can be naturally divided into a set of loosely coupled services. These services can then be classified through their different features. Krafzig et al. [41] has identified that distinguishing services into classes is extremely helpful when properly estimating the implementation and maintenance cost, and the cost factors may vary depending on the service type. However, there is no standard way to categorize services. Service classification can be different for different purposes, for example differentiating services according to their target audience [2], categorizing services through their business roles and responsibilities [28], and classifying services by using their background techniques and protocols [42]. Services in our work are characterized as follows:

- *Available Service* (basic service type), when the service already existing i.e. it may be provided by a third party or inherited from legacy SOA based systems.
- *Migrated Service* (basic service type), is the service to be generated through modifying or wrapping reusable traditional software component(s).
- *New Service* (basic service type), is the service to be developed from scratch.
- *Combined Service* is the service arising from the combination of any above three types of basic services or other combined services.

Through this type of classification, four different development areas are identified in SOA projects. These areas present both a decomposition process that results in Service Discovery, Service Migration, and Service Development, and a recomposition process that is Service Integration. The cost estimation for overall SOA-based software development can then be separated into these smaller areas with corresponding metrics. Therefore, the WBS approach is a feasible attempt for SOA-based software cost estimation following this development oriented service classification.

c) *WBS Cost Estimation Frameworks*

The proposed cost estimation framework for SOA-based software follows the WBS principle. Firstly, through the service-oriented analysis, the SOA project is divided into basic services recursively. Secondly, different sets of metrics are adopted to satisfy the cost and effort estimation for different service development processes. The total cost and effort of the SOA project will be calculated through the service integration procedure as shown in Figure 2.

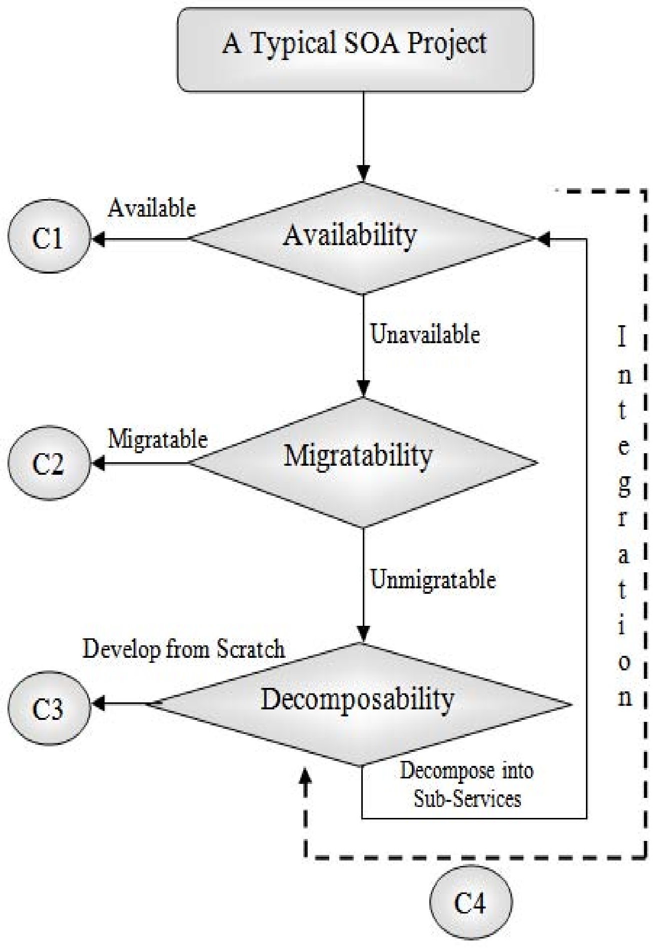

Figure 2 : Procedure of SOA Project Development Cost Estimation based on WBS.

C1 is the cost estimation model or software size measurement used to accomplish modelling or sizing work for discovering available services, C2 represents migrating potential services, C3 represents developing new services, and C4 is the cost estimation model or size measurement for calculating the service integration effort. The Decomposability condition depends on the design and real situations whether the current service should be further divided or developed as a whole. The framework in Figure 2 presents the generic process of SOA cost estimation using the WBS method. To precisely describe the WBS based cost estimation for SOA-based software development, the complete process was expressed in pseudo code (Table 1). We define the stage that service division occurs as the service levels, and the combined service stands in a higher level next to its successive component services.

Table 1: Algorithm of SOA Project Development Cost Estimation Based on WBS

```
//Treat the project at the highest-level service S to be analyzed.
double SoaCostEstimation(service S) {
double cost = 0;
//Determine the type of S according to the design and real situations.
switch (the type of S) {
case AVAILABLE:
cost += The cost of service discovery;
break;
case MIGRATABLE:
cost += The cost of service migration (service wrapping);
break;
case NEW:
cost += The cost of service development;
break;
default:
//Divide S into component services at lower level.
for each component service in S
cost += SoaCostEstimation(component service);
cost += The cost of service integration for component services in S;
break;
}
return cost;
}
```

The SOA project itself is treated at the highest-level coarse-grain service, which is also the initial input parameter of SoaCostEstimation function. Within the body of SoaCostEstimation function, the cost of the input service development will be estimated directly if the service belongs to those three basic types, or recursively calculated by analyzing and composing the cost and effort of the development for component services. When composing individual service development costs into the overall SOA-based software development cost, the strategy of supposed service integration is progressed level-by-level instead of integrating the services all at once. The reason of adopting such a strategy is that, according to our work, service integration occurring in different levels will make different contributions to the total cost and effort of the project development. A real example can be used to show the application process of the WBS based cost estimation framework for SOA-based software development in practice, which is demonstrated in the next section.

IV. Implementation

We employ Visualization RCD Beam for Service Oriented Architecture (VisRCDBeam for SOA) implemented by Yusuf et al [43] as an application case study. There are two reasons for choosing this case: The VisRCDBeam for SOA case study characterizes all the service types listed in the previous section, and there are a limited number of services which are adequate for illustrative purpose in this paper.

Global Journal of Computer Science and Technology Volume XI Issue XV Version I September 2011

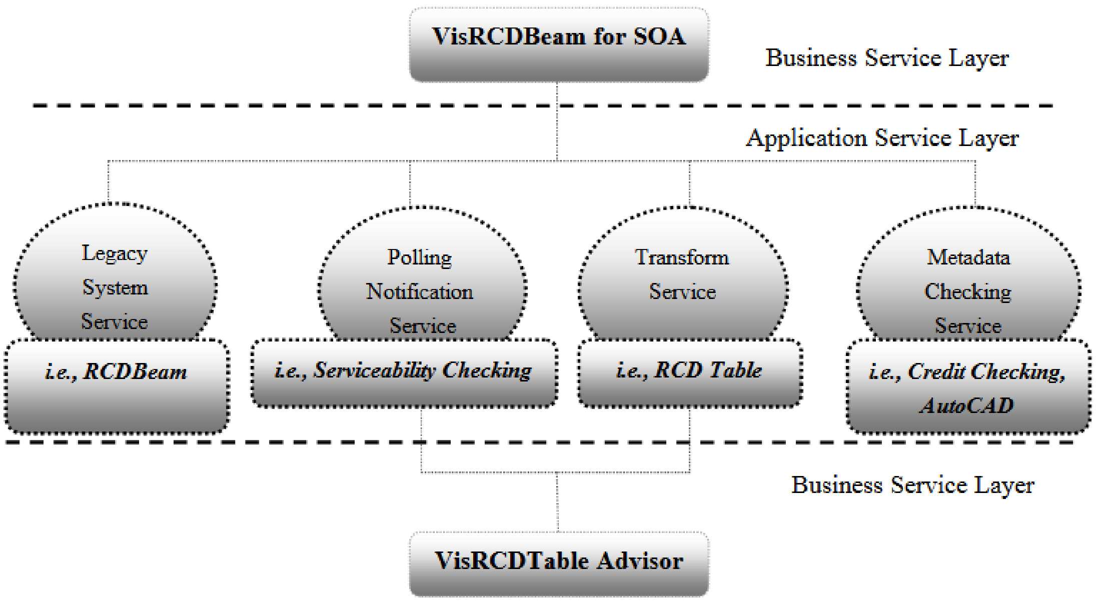

Figure 3 : Redesigned Automation System of VRCDSOA

VisRCDBeam for SOA is a SOA tool for the analysis and design of Reinforced Concrete Structures. To improve the working efficiency of Reinforced Concrete Designers, a service-oriented analysis was conducted, which decomposed the business process logic into series candidates. The tool revealed the requirements of two business services in higher level and four application services in lower level. The improved automation system is represented in Figure 3 following current disciplines:

a. RCDBeam interface is the Legacy System Service which is migrated from the previous project.
b. Serviceability checking represents the Polling Notification Service which is a coarse grain service containing check for minimum and maximum area of steel and check for deflection functional services. The Transform Service is the RCD table for picking bar sizes. These are new services that were developed from scratch.
c. Credit checking for authentication and security, and AutoCAD interface for visualization purposes is represented by the Metadata Checking Service. They are the available service provided by third party.
d. "VisRCDBeam for SOA" Service and VisRCDTable Advisor Service are both combined services containing all or some of above basic services. The procedure of cost and effort estimation for developing this redesigned service-oriented project is illustrated in Figure 4. The detailed steps are elaborated as follows:

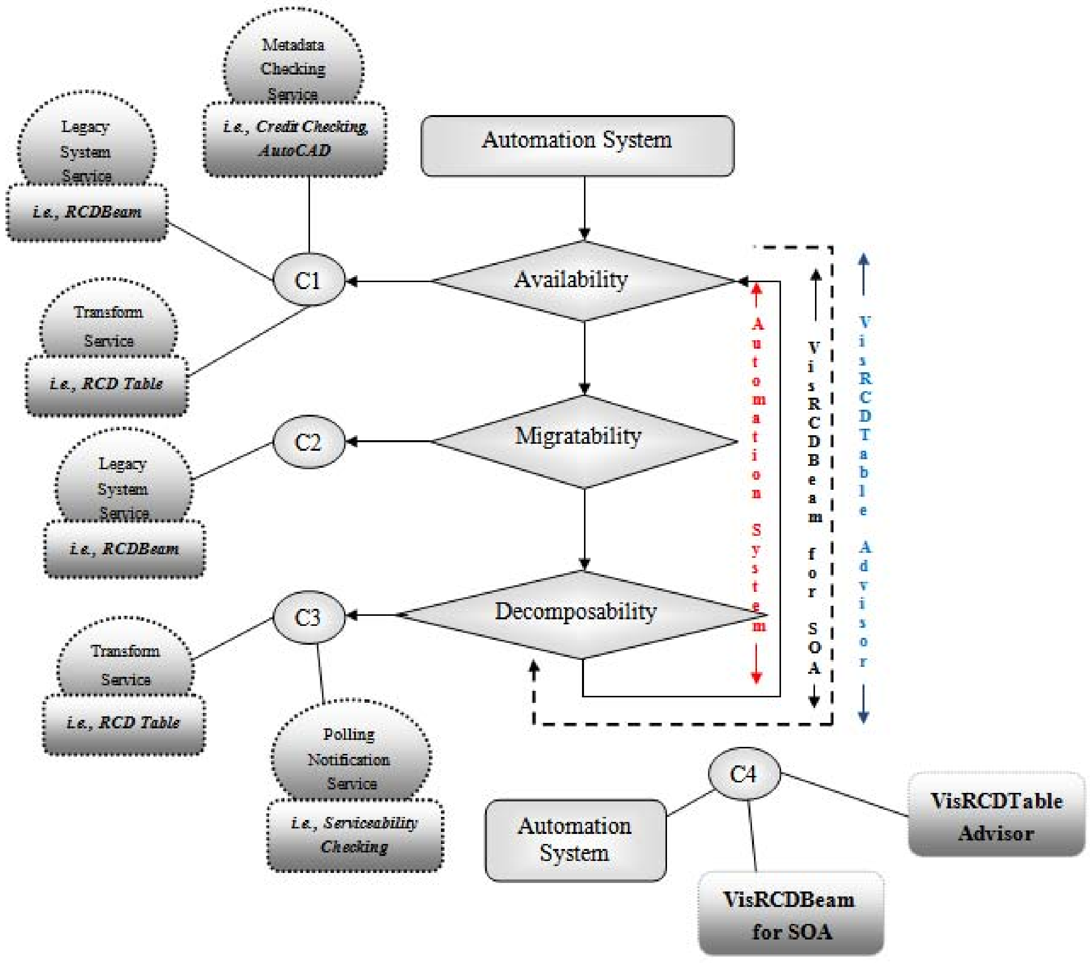

Figure 4 : Procedure of Cost and Effort Estimation for "VisRCDBeam for SOA" case study

a. Divide the Automation System into *VisRCDBeam for SOA* Service and *VisRCDTable Advisor* Service.
b. Divide the *VisRCDBeam for SOA* Service into its four basic component services.
c. Estimate the cost and effort of discovering the available Metadata Checking Service *(i.e., Credit checking and AutoCAD services)* by using corresponding metrics C1.
d. Estimate the cost and effort of migrating the Legacy System Service *(i.e., RCDBeam)* by using corresponding metrics C2.
e. Estimate the cost and effort of developing the Polling Notification Service (i.e., Serviceability checking) and Transform Service *(i.e., RCD Table)* by using corresponding metrics C3.
f. Estimate the cost and effort of integrating the above four component services into the *VisRCDBeam for SOA* Service by using corresponding metrics C4.
g. Divide the *VisRCDTable Advisor* Service into its two basic component services.
h. Notice that Legacy System Service *(i.e., RCDBeam)* and Transform Service *(i.e., RCD Table)* have both been taken into account.
i. Estimate the cost and effort of mining the Legacy System Service *(i.e., RCDBeam)* and Transform Service *(i.e., RCD Table)* by using corresponding

metrics C1. Since these two services are in the same project and can be directly identified, the cost and effort here can be treated as zero in this special case.

j. Estimate the cost and effort of integrating the above two component services into the *VisRCDTable Advisor* Service by using the corresponding metrics C4.

k. Estimate the cost and effort of integrating the *VisRCDBeam for SOA* Service and *VisRCDTable Advisor* Service into the Automation System by using the corresponding metrics C4.

l. Sum up all the estimation results to calculate the total cost and effort of the Automation System development. Through the demonstration of the *VisRCDBeam for SOA* case, the WBS framework is proven helpful for simplifying and regulating the SOA-based software cost estimation. Moreover, all the simplified cost estimation problems are independent enough to be solved in parallel. The uniform and explicit working procedure within this WBS framework is then a feasible attempt to SOA based software cost estimation.

V. SOA Benefits

SOA benefit organizations in different ways, depending on the respective goals and the manner in which SOA is applied. We have generalized the list of common benefits and certainly not exhaustive. It is merely an indication of the potential this architectural platform has to offer.

a) *Improved integration (and intrinsic interoperability)*

SOA can result in the creation of solutions that consist of inherently interoperable services. The net result is intrinsic interoperability, which turns a cross-application integration project into less of a custom development effort, and more of a modeling exercise. The cost and effort of cross-application integration is significantly lowered when applications being integrated are SOA-compliant.

b) *Inherent reuse*

Service-orientation promotes the design of services that are inherently reusable. Building services to be inherently reusable results in a moderately increased development effort and requires the use of design standards. Subsequently leveraging reuse within services lowers the cost and effort of building service-oriented solutions.

c) *Streamlined architectures and solutions*

The concept of composition is another fundamental part of SOA. It is not, however, limited to the assembly of service collections into aggregate services. The WS platform is based in its entirety on the principle of composability. This aspect of service-oriented architecture can lead to highly optimized automation environments, where only the technologies required actually become part of the architecture. Realizing this benefit requires adherence to design standards that govern allowable extensions within each application environment. Benefits of streamlined solutions and architectures include the potential for reduced processing overhead and reduced skill-set requirements (because technical resources require only the knowledge of a given application, service, or service extension).

d) *Leveraging the legacy investment*

The industry-wide acceptance of the Web services technology set has spawned a large adapter market, enabling many legacy environments to participate in service-oriented integration architectures. This allows IT departments to work toward a state of federation, where previously isolated environments now can interoperate without requiring the development of expensive and sometimes fragile point-to-point integration channels. Though still riddled with risks relating mostly to how legacy back-ends must cope with increased usage volumes, the ability to use what we already have with service-oriented solutions that we are building now and in the future is extremely attractive. The cost and effort of integrating legacy and contemporary solutions is lowered. The need for legacy systems to be replaced is potentially lessened.

e) *Establishing standardized XML data representation*

On its most fundamental level, SOA is built upon and driven by XML. As a result, an adoption of SOA leads to the opportunity to fully leverage the XML data representation platform. A standardized data representation format (once fully established) can reduce the underlying complexity of all affected application environments. Past efforts to standardize XML technologies have resulted in limited success, as XML was either incorporated in an ad-hoc manner or on an "as required" basis. These approaches severely inhibited the potential benefits XML could introduce to an organization. With contemporary SOA, establishing XML data representation architecture becomes a necessity, providing organizations the opportunity to achieve their goal, the cost and effort of application development is reduced after a proliferation of standardized XML data representation is achieved.

f) *Focused investment on communications infrastructure*

Because Web services establish a common communications framework, SOA can centralize inter-application and intra-application communication as part of standard IT infrastructure. This allows organizations to evolve enterprise-wide infrastructure by investing in a single technology set responsible for communication. The cost of scaling communications infrastructure is reduced, as only one communications technology is required to support the federated part of the enterprise.

g) "Best-of-breed" alternatives

Some of the harshest criticisms laid against IT departments are related to the restrictions imposed by a given technology platform on its ability to fulfill the automation requirements of an organization's business areas. This can be due to the expense and effort required to realize the requested automation, or it may be the result of limitations inherent within the technology itself. Either way, IT departments are frequently required to push back and limit or even reject requests to alter or expand upon existing automation solutions. SOA won't solve these problems entirely, but it is expected to increase empowerment of both business and IT communities. A key feature of service-oriented enterprise environments is the support of "best-of-breed" technology. Because SOA establishes a vendor-neutral communications framework, it frees IT departments from being chained to a single proprietary development and/or middleware platform. For any given piece of automation that can expose an adequate service interface, we now have a choice as to how we want to build the service that implements it. The potential scope of business requirement fulfillment increases, as does the quality of business automation.

h) Organizational agility

Agility is a quality inherent in just about any aspect of the enterprise. A simple algorithm, a software component, a solution, a platform, a process all of these parts contain a measure of agility related to how they are constructed, positioned, and leveraged. How building blocks such as these can be realized and maintained within existing financial and cultural constraints ultimately determines the agility of the organization as a whole. Much of service-orientation is based on the assumption that what you build today will evolve over time. One of the primary benefits of a well-designed SOA is to protect organizations from the impact of this evolution. When accommodating change becomes the norm in distributed solution design, qualities such as reuse and interoperability become commonplace. The predictability of these qualities within the enterprise leads to a reliable level of organizational agility. However, all of this is only attainable through proper design and standardization. Change can be disruptive, expensive, and potentially damaging to inflexible IT environments. Building automation solutions and supporting infrastructure with the anticipation of change seems to make a great deal of sense. A standardized technical environment comprised of loosely coupled, composable, and interoperable and potentially reusable services establishes a more adaptive automation environment that empowers IT departments to more easily adjust to change. Further, by abstracting business logic and technology into specialized service layers, SOA can establish a loosely coupled relationship between these two enterprise domains. This allows each domain to evolve independently and adapt to changes imposed by the other, as required. Regardless of what parts of service-oriented environments are leveraged, the increased agility with which IT can respond to business process or technology-related changes is significant. The cost and effort to respond and adapt to business or technology-related change is reduced.

VI. Discussion

The aim of ***visRCDBeam for SOA*** is to upgrade its automation system so that it could remain competitive with other RCD tools and continue its business relationship with its primary client. We proceeded with a service-oriented analysis that decomposed its business process logic into a series of service candidates. This revealed the need for the following potential services and service layers: (a) A business service layer consisting of two tasks centric business services namely visRCDBeam for SOA and VisRCDTable Advisor, (b) An application service layer comprised of four application services. Each business process was represented with a task-centric business service that would act as a controller for a layer of application services. Reusability and extensibility in particular were emphasized during the design of the application services. We intend to have the initial SOA to consist of services that supported both of its current business processes, while being sufficiently extensible to accommodate future requirements without too much impact. To realize the visRCDBeam for SOA tool, we compose these services into a two-level hierarchy where the parents VisRCDBeam for SOA and VisRCDTable advisor business services coordinate the execution of all application services. Unlike many of the current cost estimation approaches, the proposed WBS framework uses a set of metrics to satisfy the development cost estimation for SOA-based software. The WBS concentrates on the software development process. It list Service Discovery as an individual cost estimation area as well as Service Migration, Service Development, and Service Integration. The framework estimates overall cost and effort through the independent estimation activities in four different development areas of an SOA application. WBS framework is generic and flexible by switching different types of metrics; it could satisfy different requirements of SOA-based software cost estimation such as building cost estimation model, measuring software size, and predicting the overall cost ultimately. One issue is that there are currently few available metrics for the detailed cost estimation for SOA-based software development. Future research should develop new metrics to resolve this issue. Meanwhile, some reusable existing metrics can be integrated into the proposed WBS framework, for example Tansey and Stroulia's work [18] [related to

Service Development and SMART method [26] are related to Service Migration. Over all, instead of trying to enumerate SOA project types, the WBS framework unifies and regulates the cost and effort estimation for SOA-based software development.

VII. Conclusion

Poor project management could bring failure to SOA. Gone are the days when one person is the SOA architect, developer, data architect, network architect and security specialist. The complexity of SOA should not be underestimated. Failure to implement and adhere to SOA governance is an imperative issue; the development effort is shifting from building services to consuming services. Vendors could be allowed to drive the architecture but relying too much on vendors can be a disaster. Software cost estimation plays a vital role in software development projects, especially for SOA-based software development. Current cost estimation approaches for SOA-based software are inadequate due to the architectural difference and the complexity of SOA applications. This paper offers a WBS cost estimation framework for SOA-based software development. Based on the principle of Divide and Conquer theory, this framework can be helpful for simplifying the complexity of SOA cost estimation. By hosting different sets of metrics, this generic framework will be suitable not only for the complete cost estimation work but also for the partial requirements, such as building estimation model, and measuring the size of SOA applications. We have fulfilled our original goals by producing proper costing of an SOA project that supports two service-oriented solutions. Online transaction is now possible. New requirements can be accommodated with minimal impact. The standard application service layer will likely continue to offer reusability functionalities to accommodate the fulfilment of new requirements. And any functional gaps will likely be addressed by extending the services without significantly disrupting existing implementations. Furthermore, should we decide to replace our task-centric business services with an orchestration service layer in the future, the abstraction established by the existing application service layer will protect the application services from having to undergo redevelopment. We have established a legacy system service (which is essentially a wrapper service for graphics drawing) as part of its application service layer it has opened up a generic and point that can facilitate integration. There is an old saying that you cannot manage what you cannot measure. By increasing the number of "moving pieces" in IT solutions, SOA increases the number of pieces that require measurement. Given the relative immaturity of the SOA paradigm, it is particularly important now, when best practices have not yet been established and the understanding of cause and effect is limited. Indeed, the inability to collect cost and schedule data at the task level may be part of the reason why so many case studies in SOA only present project-level estimates of averted cost.

References Références Referencias

1. Cresswell, Anthony M., (2004). Return on Investment In Information Technology: A Guide for Managers, Center for Technology in Government, University at Albany, August 2004.
2. Josuttis, N. M. (2007). SOA in Practice: The Art of Distributed System Design, Sebastopol: O'Reilly Media, Inc.
3. Cardoso, J. (2005). "How to Measure the Control-Flow Complexity of Web Processes and Workflows ," Workflow Handbook 2005, Layna Fischer, Apr. 2005, pp. 199-212.
4. Jamil, E. (2009). "SOA in Asynchronous Many-to One Heterogeneous Bi-Directional Data Synchronization for Mission Critical Applications," We Do Web Sphere, Jul. 2009. [Online]. Available: http://wedowebsphere.de/news/1528/SOA%20in%20Asynchronous%20Many-toone%20 Heterogeneous %20BiDirectional%20Data%20 Synchronization%20. [Accessed: Nov. 2009].
5. Robert D. Bryan, Brand K. Niemann and Kartik Mecheri (2011). Perspectives on SOA ROI. Karsun Solutions Enterprise Modernization Expert. Geoff Raines mitre.org/news/digests/ enterprise_ modernization/09_08/external.html.
6. Brown, A; Johnston, S., and Kelly, K. (2002). *Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications.* Santa Clara, CA: Rational Software Corporation.
7. Lloyd Brodsky (2010). Meanindful Cost-Benefit Analysis for Service-Oriented Architecture Projects. Proceedings of the Seventh Annual Acquisition Research Symposium Thursday Sessions Volume II, Monterey, California, U.S. Government or Federal Rights License.
8. Maxfield, J., Fernando, T. and Dew, P. (1995). A Distributed Virtual Environment for Concurrent Engineering", in IEEE Proceedings on Virtual Reality Annual International Symposium, March 1-15, Research Triangle Park, North Carolina, pp.162-170.
9. Dong, A and Agogino, A.M. (1998). Managing design information in enterprise-wide CAD using 'smart drawings', Computer-Aided Design, 30 (6), 425-435.
10. Roy, U. and Kodkani, S.S. (1999). Product modelling within the framework of the World Wide Web, IIE Transactions, 31 (7), 667-677.
11. Huang, G.Q. and Mak, K.L. (2000). WeBid: A Web-based Framework to Support Early Supplier

Involvement in New Product Development, Robotics and Computer Integrated Manufacturing, 16 (2-3), 169-179.

12. Kan, H.Y., Duffy, V.G. and Su, C.J. (2001). An Internet Virtual Reality Collaborative Environment for Effective Product Design, Computers In Industry, 45 (2), 197-213.
13. Yusuf Lateef Oladimeji, Olusegun Folorunso, Akinwale Adio Taofeek and Adejumobi, I.A. (2011). "Service Oriented Application in Agent Based Virtual Knowledge Community". Computer and Information Science Journal, Vol. 4, No. 2; March 2011.
14. Lewis, Grace and Wrage, Lutz. (2011). *Approaches to Constructive Interoperability* (CMU/SEI-2004-TR-020). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www. sei. cmu.edu/publications/documents/04.reports/04tr02 0.html.
15. Senn, James A. (1987). *Information Systems in Management*. 3rd ed. Belmont, California: Wadsworth Publishing.
16. Mishan, E. J. (1976). *Cost-Benefit Analysis.* 2nd ed. New York: Praeger Publishers.
17. Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E. R., Madachy, D.J., Reifer, and Steece, B. (2000). Software Cost Estimation with COCOMO II. New Jersey: Prentice Hall PTR, Aug. 2000.
18. Tansey, B. and Stroulia, E. (2007). "Valuating Software Service Development: Integrating COCOMO II and Real Options Theory," Proc. the First International Workshop on the Economics of Software and Computation, IEEE Press, May 2007, pp. 8-8, doi: 10.1109/ESC.2007.11.
19. Stojanovic Z. and Dahanayake, A. (2005). Service-Oriented Software System Engineering: Challenges and Practices. Hershey, PA: IGI Global, Apr. 2005.
20. Sommerville, I. (2006). Software Engineering, 8th ed.. London: Addison Wesley, Jun. 2006.
21. Santillo, L. (2007). "Seizing and Sizing SOA Application with COSMIC Function Points," Proc. the 4th Software Measurement European Forum, Rome, Italy, May 2007.
22. Liu, J., Xu, Z., Qiao, J., and Lin, S. (2009). "A Defect Prediction Model for Software Based on Service Oriented Architecture using EXPERT COCOMO," Proc. Chinese Control and Decision Conference (CCDC '09), IEEE Press, Jun. 2009, pp. 2591-2594, doi: 10.1109/CCDC .2009. 5191800.
23. Bergey, J., O'Brien, L., and Smith, D. (2002). "Using the Options Analysis for Reengineering (OAR) Method for Mining Components for a Product Line," 316-327. *Software Product Lines: Proceedings of the Second Software Product Line Conference (SPLC2).* San Diego, CA, August 19-22, 2002. Berlin, Germany: Springer, 2002.
24. O'Brien, L. (2009). "A Framework for Scope, Cost and Effort Estimation for Service Oriented Architecture (SOA) Projects," Proc. 20th Australian Software Engineering Conference (ASWEC'09), IEEE Press, Apr. 2009, pp. 101-110, doi: 10.1109/ASWEC.2009.35.
25. Lewis, G., Morris, E., O'Brien, L., Smith, D., and Wrage, L. (2005). "SMART: The Service-Oriented Migration and Reuse Technique," CMU/SEI-2005-TN-029, Software Engineering Institute, USA, Sept. 2005.
26. Umar, A. and Zordan, A. (2009). "Reengineering for Service Oriented Architectures: A Strategic Decision Model for Integration versus Migration," Journal of Systems and Software, vol. 82, Mar. 2009, pp. 448-462, doi: 10.1016/j.jss.2008.07.047.
27. Bosworth, A. (2001). "Developing Web Services," Proc. 17th International Conference on Data Engineering (ICDE 2001), IEEE Press, Apr. 2001, pp. 477-481, doi: 10.1109/ICDE. 2001. 914861.
28. Erl, T. (2005). Service-Oriented Architecture: Concepts, Technology, and Design. Crawfordsville: Prentice Hall PTR, Aug. 2005.
29. Progress Actional (2008). "Web Services and Reuse" http://www.actional.com/resources/ whitepapers/SOA-Worst-Practices-Vol-I/Web-Services-Reuse.html 28 March 2008.
30. Poulin Jeffery (2006). "The ROI of SOA Relative to Traditional Component Reuse," Logic Library.
31. Mili, H., A. Mili, S. Yacoub and Addy, E., (2002). Reuse-Based Software Engineering: Techniques, Organization and Controls. John Wiley and Sons Ltd.
32. Linthicum, D. (2007). "How Much Will Your SOA Cost?," SOAInstitute.org, Mar. 2007. [Online]. Available: http://www.soainstitute.org/articles/article/ article/how-much-willyour-soa-cost.html. [Accessed: Nov. 2011].
33. Knuth, D. E. (1998). The Art of Computer Programming: Volume 3, Sorting and Searching, 2nd ed.. Reading, MA: Addison-Wesley Professional May 1998.
34. Bai, Y. and Ward, R. C. (2007). "A Parallel Symmetric Block-Tridiagonal Divide-and-Conquer Algorithm," Transactions on Mathematical Software (TOMS), vol. 33, Aug. 2007, pp. A25, doi: 10.1145/1268776.1268780.
35. Khalilian, M. F., Boroujeni, Z., Mustapha, N., and Sulaiman, M. N. (2009). "KMeans Divide and Conquer Clustering," Proc. the 2nd International Conference on Computer and Automation Engineering (ICCAE 2009), IEEE Press, Mar. 2009, pp. 306-309, doi: 10.1109/ICCAE.2009.59.
36. Lin, T. Y. (2009). "Divide and Conquer in Granular Computing Topological Partitions," Proc. Annual Meeting of the North American Fuzzy Information

Processing Society (NAFIPS 2009), IEEE Press, Jun. 2005, pp. 282-285, doi: 10.1109/NAFIPS.2005. 1548548.

37. Hu F., and Wang, G. (2008). "Huge Data Mining Based on Rough Set Theory and Granular Computing," Proc. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'08), IEEE Press, Dec. 2008, pp. 655-658, doi: 10.1109/WIIAT.2008.84.
38. Zhang, C. and Xue, B. (2009). "Divide-and-Conquer: A Bubble Replacement for Low Level Caches," Proc. the 23rd International Conference on Supercomputing (ICS'09), ACM, Jun. 2009, pp. 80-89, doi: 10.1145/1542275.1542291.
39. Norfolk, D. (2007). "SOA Innovation and Metrics," IT-Director.com, Dec. 2007. [Online]. Available: http://www.itdirector.com/business/change/content. php?cid=10146. [Accessed: Nov. 2009].
40. Chaos, D. (2009). "SOA is not dead, but complexity is killing some implementations," Technoracle (a.k.a. "Duane's World"), Jan. 2009. [Online]. Available: http://technoracle.blogspot. com/2009/01 /soa-isnot-dead-but-complexity-is.html. [Accessed: Jul. 2009].
41. Krafzig, D., Banke, K., and Slama, D. (2004). Enterprise SOA: Service-Oriented Architecture Best Practices, Upper Saddle River: Prentice Hall PTR, Nov. 2004.
42. Davies, J., Schorow, D., Ray, S. and Rieber, D. (2008). The Definitive Guide to SOA: Oracle Service Bus, 2nd ed.. New York: Apress, Sept. 2008.
43. Yusuf Lateef Oladimeji, Olusegun Folorunso, Akinwale Adio Taofeek and Adejumobi, I.A. (2011). "Visualizing and Assessing a Compositional Approach to Service-Oriented Business Process Design Using Unified Modelling Language (UML)". Computer and Information Science Journal, Vol. 4, No. 3; May 2011.

This page is intentionally left blank