American Journal of Software Engineering and Applications
2013; 2(6): 133-140
Published online November 20, 2013 (http://www.sciencepublishinggroup.com/j/ajsea)
doi: 10.11648/j.ajsea.20130206.12

Extended implementation of change impact analysis model-based framework to enhance predicting the effect of a change of service in a grid environment

Obeten Obi Ekabua

Department of Computer Science, North-West University, Mafikeng Campus, Mmabatho 2735, South Africa

Email address:
obeten.ekabua@nwu.ac.za, ekabuao@yahoo.com

To cite this article:
Obeten Obi Ekabua. Extended Implementation of Change Impact Analysis Model-Based Framework to Enhance Predicting the Effect of a Change of Service in a Grid Environment. *American Journal of Software Engineering and Applications.* Vol. 2, No. 6, 2013, pp. 133-140. doi: 10.11648/j.ajsea.20130206.12

Abstract: Continuous monitoring of changes to utility services and products in a distributed information system is an interesting issue in software engineering. These changes affect the semantics and structural complexity of the system, as a change to one part will in most cases, result in changes to other parts. Therefore, in design and redesign for customization, predicting this change presents a significant challenge. Changes are intended to fix faults, improve or update products and services. Lack of validated, widely accepted, and adopted tools for planning, estimating, and performing maintenance contributes to the problem. One effective way of assessing changeability effect is to assess the impact of changes through a well validated model and framework. This research paper is an extended report on the implementation of a change propagation framework, together with it's associated change impact analysis factor adaptation model, and a fault and failure assumption model to predict the effect of a change of a service in a grid environment. While implementing the framework, data was collected for three hypothetical years, thus helping to predict the next two (2) years consecutively. Significant results corresponding to the impact analysis factor were obtained showing the viable practicality of the use of Bayesian statistics (as against unreported regression method) satisfying best-fit prediction. We conclude that, the higher the number of dependent services on a faulty service requiring a change, the higher the impact due to fault propagation.

Keywords: Change Impact Analysis, Service Provisioning, Software Metrics, Service Maintenance, Bayesian Statistics, Grid Environment

1. Introduction

Software engineering empirical research is expressed as a rigorous activity as it hinges on the formulation of hypothesis and a framework for the evaluation of the hypothesis. Any measurement in software engineering is targeted for assessment and prediction. A model alone is insufficient for prediction except if it is accompanied by the model parameter determination and results interpretation. Therefore, any prediction system must consist of a model, model parameter determination and results interpretation [1].

Computing systems' evolution (hardware and software) can be traced to changes in the original requirements, different hardware platform adoption and efficiency improvement. Maintenance management approaches indicate different possible changes during the maintenance process and this is seen as an indication of evolutionary changes. As a result of the complexity involved, error probability becomes high and some of these errors could result into undeterministic consequences such as loss of life, money, time and damage to the environment. This makes system evolution management an important phase in system development and maintenance. Hence, a maintainer faces the challenge of how to respond rapidly, correctly, and efficiently to change. This is because the maintainer in most cases is not directly involved in the system development, as responding to change requires system understanding and change identification before performing the change [2]. The use of formal methods is crucial as it enhances system understanding to the point of unfolding undetected propagating changes [3].

A combination of distributed object computing, component based computing and web-based concepts into what is now known as Service Oriented Architectures has

emerged as an approach for developing dynamic and heterogenous service provisioning environments. This technology evolution, combined with the web revolution, poses new challenges in the context of service provisioning. With the massive diffusion of the internet as a distributed environment for service provisioning, people's interaction with computers has dramatically changed. People relate not to their own computer, but rather, to their point of presence within the service provisioning environment [4, 5, 6].

System level interoperability and dependability are important issues resulting from the integration of different technologies and middlewares into the same distributed systems [7]. To effectively study issues of interoperabilty and dependability in Service Oriented Architecture (SOA), it is equally necessary to analyze it along measurements and maintenance dimension through Change Impact Analysis (CIA) technique. The CIA dimension will improve these issues from the end-user perspective. To this aim, we address the issues from the point of the need for a service change resulting from fault and failure. Bayesian technique is used to predict the need for a service change over a certain period of time, to forstall breakdown in operation and enhance maintenance as a means of quick resolution of expected service failure. This will make SOA unique when compared to traditional middleware-based systems and will also help the operational life of the service configuration and remove stress from the component.

2. Background on Change Impact Analysis

The effect of one thing on another or the consequences of a change is defined as an impact. Impact analysis (IA) is used to determine the scope of a change request as the basis for accurate resource plannning and scheduling, and to confirm the cost/benefit justification. Service change impact analysis (CIA) estimates what will be impacted in service and related documentation if proposed service change is made. It determines the scope of the change and the complexity of the change. The qualitative and quantitative effects of that change on other part of the item are the major concern of the study of CIA [8].

Experience has shown that a comprehensive up-front analysis of requirements during software development pays high dividend by reducing the risk of costly re-work and the potential of errors in planning estimates. CIA makes the effect of a change visible before the change is implemented. CIA can be used as a measure of the cost of a change. The more the change causes other changes, the higher the cost of the change.

The resulting challenges of the idea of interoperability can be viewed from two perspectives – technical interoperability and dynamic interoperability. Technical interoperability involves the existence of a protocol for exchanging data and information between participating services. A communication infrastructure is established to allow information to be exchanged between services with unambiguously defined underlying networks and protocols. While dynamic interoperability is defined from the point that as services are requested, provided and consumed over time, the state of that service will change and this includes the assumptions and constraints that affect information interchange. If services have attained dynamic interoperability, they comprehend the state changes that occur in the assumptions and constraints that each is making over time, and they are able to take advantage of those changes. The interest is specifically on the effects of operations as it becomes increasingly important that the effect of the information exchange is unambiguously defined.

3. Change Propagation Framework

Changes are endermic to software artefacts and the services provided by these artefacts. When a change is effected in a particular service connected to grid, it isoften difficult to determine the propagation of this service change.

We therefore present a change propagation framework shown in Fig.1 to support change automation in any grid engineering methodology.

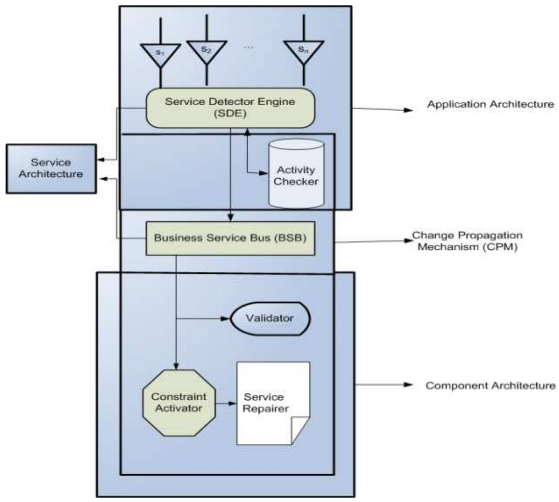

Figure 1. *Change Propagation Framework [16, 17]*

The *Service Detector Engine* (SDE) contains all consumer made available set of grid services (s_1, s_2, . . ., s_n) under utilization. SDE liases with *Business Service Bus* (BSB), a concept developed by Component Based Development and Integration (CBDI) [9] and incorporated into our framework to form *Service Architecture* (SA) responsible for providing a bridge between the implementation and the consuming application, creating a logical view of a set of services, which are available for use and invoke by a common interface and management architecture. The *Activity Checker* (AC) is responsible for the specification of the constraints that a well-formed service design should satisfy in order to check whether the application's design is in conformance to the main host design. Violation of the rules governing the activity checker will trigger a constraint violation event from the *Constraint Activator* (CA) to be

returned to the *Change Propagation Mechanism* (CPM). This informs the *Service Repairer* (SR) of a triggered event calling for a way of fixing the violated constraint by performing actions, which change the application's design and keeps record of the ripple effect. The mechanism *Validator* (V) is responsible for checking the consistency of the change to the design (through the AC), which can result in further actions [16].

There are three major architectural perspectives for SOA namely: Application Architecture, Service Architecture and Component Architecture and our framework has these incorporated into it. The architecture has two perspective views: Consumer and Provider. The salient aspect of the architecture is that the consumer of a service should not be interested in the implementation detail of a service, but the service provided. This is because the implementation architecture could vary from provider to provider, but still deliver the same service. Additionally, the provider should not be interested in the application that the service is consumed in, because new unforseen application will reuse the same set of services. The consumer's main interest is in the application architecture and the services used, but not in the detail of the component architecture. The interest is in some level of details in the general business objects that are of mutual interest, for example, provider and consumer need to share a view of what is a subscription. But the consumer does not need to know how the service component and database are implemented. Also, the provider is focused on the component architecture and the service architecture, but not on the application architecture. Again, they both need to understand certain information about the basic application in order to be able to set any sequencing rules including pre and post conditions.

SOA provides the need to be able to manage services as first order deliverables. The communication key between the provider and the consumer is service. There is the need therefore, for a service provisioning architecture in the form of this framework, that will ensure that services are not reduced to the status of interfaces, but have an identity of their own and can be managed individually and in sets. BSB as shown in our framework is incorporated to meet this requirement by providing a logical view of the available services for any business domain. BSB answers such questions as: (1) What services do I need? (2) What services are available to me? (3) What alternative services are available? (4) What services will operate together? (5) What services are connected to me? [10]. Our framework is generic because it can be adapted in any general service provisioning engineering methodologiy that can enhance monitoring change propagation. The most important component of the framework is the Change Propagation Mechanism (CPM), which is represented and implemented within the service provisioning architecture and the component architecture. CPM detects any change service due to the triggering effect generated and validated. CPM notifies the SR of the ripple effect for immediate action of fixing the service.

4. Change Impact Analysis Factor Adaptation Model (CIAFAM)

Maintainability refers to a situation where a software system or a component is modified to correct faults, improve performance or adapt to a change environment [11]. Increase in maintainance cost has become a concern to developers and users of software systems. Unfortunately, developers and managers underestimate the time and effort required to perform changes. Also, lack of validated, widely accepted, and adopted tools for planning, estimating, and performing maintenance contributes to the problem. Changeability is vital to maintainability mostly in frequent requirement changing environment. But one effective way to assess changeability is to assess the impact of changes through an impact model [12, 13].

While considering the need for a change of service in a system, importance should be placed on identifying system components that may be impacted after such a change. This enables the system to keep running perfectly after a change implementation. A system absorbs a change easily if the impacted components is of a small number. One effective method of accounting for changes in services is to perform CIA and our framework is accessed by the impact model described. Our main concern is pivoted on how the system reacts to changes that leads to propagation.

For any given change M in a service N, we can describe a set of impacted service as a boolean expression. The Impact Analysis Factor (IAF) for such hypothetical change can be given by:

$$IAF\ (M,N) = A*(\sim \rho) + A'$$

Where

*, +, ~ denotes the usual boolean operators: conjunction, disjuction and negation respectively

M = a given change

N = a given service

A = there is an association between M and N

ρ = K is derived from the change service

A' = there is an occurence of aggregation link between M and N

IAF = Impact Analysis Factor

This expression implies that a service in association (A) with M and not derived (~ ρ) from the change service M or services that are in aggregation link (A') with M, are impacted. It is important to state that this impact model only predicts, which services would be impacted if a change was really made. If a service is really impacted, it means there is the propensity of propagation in which case the IAF becomes 1. We concentrate on changes that have a synthetic impact, therefore, appropriate measures are based on impacts that are dependent on the static nature of the provisioning system. This implies that impacts have a likelihood of propagation [13, 14].

5. Fault and Failure Assumption Model

Depending on the architectural level, time phased and other specific service parameters, SOA failure modes may change. In modern SOA, common failures are due to unavailable infrastructure, client crash, service failure, server crash, session failure and component failure. Therefore, a generic failure F_k is defined as:

$$F_k = f(al, t_p, ss_p)$$

Where;

al = the architectural level of the faulty components

t_p = the time phase during which the fault occur

ss_p = the set of specific service parameters identifying the state of the particular service involved in the failure.

If each failure F_k is identified, the system failure modes can be represented as:

$$F = F_1 \mid F_2 \mid .. \mid F_n = \sum_{k=0}^{k=n} F_k$$

This implies that the system fails if at least one of the identified failuresoccurs. Our failure is recorded as a boolean value (0, 1) with respect to tp. Increasing redundancy degree may lead to increase in possible sources of failure resulting in potential decrease in dependability [2]. To understand the impact of redundancy, dependability and interoperability on our framework, the failure model is necessary.

Our fault assumption is based on a fail-silent assumption where either a service is actively operating or does not answer at all. This assumption is justified on the basis of our CIAFAM whose IAF is a boolean (0,1). When the value is 1, it indicates a fault (requiring change), but when the value is 0, it is in its active state.

To analyze the error type that a faulty service may induce in a grid environment, we formulate the concept of failure that will enhance change prediction as:

$$F = f(al, t_p, t_m, i, d)$$

where al and t_p are as previously defined and ssp = t_m, i, d where

t_m = time (in months) when a fault is detected

$t_m = x, y \ \{0 \le x \le 6; 6 < y \le 12\}$ months

i = the particular service item involve in failure

$i \in I = \{a, b, c, \ldots, z\}$ where I is the set of available services.

d = the descriptor of the faulty session

6. Bayesian Approach Used for Prediction

Bayesian statistical approach has proven useful for both inferential exploration of previously undetermined relationships among services as well as descriptions of these relationships upon discovery. The process of service change prediction in a service provisioning environment can be computationally intensive and NP-hard in its algorithmic implications. Predicting a change of service in an SOA service provisioning environment for a solution to a problem is usually NP-hard problem resulting in a combinatorial explosion of possible solutions to investigate. This problem is often ameliorated through the use of heuristics, or sub-routines to make worthwhile choices along the SOA decision tree. We have used Bayesian approach to replace heuristic methods by introducing a method where the probabilities of SOA decision tree are updated continually during predictive decision making.

We express the Bayesian approach as

$$P(H \setminus q, r) = \frac{P(H / r) P(q / H, r)}{P(q / r)} \quad 1$$

The term $P(H \setminus q, r)$ is defined as the "posterior probability" which is being continuously updated. It is the probability of H after considering the effect of r on q. The term $P(q / H, r)$ is the marginal probability known generally as the likelihood, and gives the probability of the evidence assuming the hypothesis H and the background information r is true. The term $P(H / r)$ is called the prior probability which measures the strength of belief probabilistically of the services, prior to any execution of experiment and may depend on the strength r service having the assumption of being true. For computational exigency of discrete nodes for SOA services, the updated services marginal probability density function $P(H \setminus q, r)$ is calculated using the chain rule of probability

$$p(x_1, x_2, \ldots\ldots x_n / \eta) = \prod_{i=1}^{n} p(x_i / x_1, x_2, \ldots x_{i-1}, \eta) \quad 2$$

Thus for the product rule of probability, (2) for each x_i, $\prod_i \subseteq \{x_1, \ldots\ldots, x_{i-1}\}$ are set of services that renders x_i *and* $\{x_1, x_2, \ldots\ldots, x_{n-1}\}$ conditionally independent in an SOA system. Therefore we have:

$$p(x_1, x_2, \ldots\ldots, x_n / \eta) = (p(x_i / \prod_{\kappa}, \eta) \quad 3$$

With equation (3), the Bayesian environment structure then encodes the assertion of conditional independence in equation (2). Thus by this assumption, a Bayesian structure is a directed acyclic graph such that (i) each variable in the domain of the environment corresponds to a node in the SOA and (ii) the parents of the node corresponding to x_i are the nodes formulated from binomial distributions:

$$p(x) = \frac{n!}{x!(n-x)!} p^n (1-p)^{n-x} \quad 4$$

Substituting equation (4) into equation (1), we obtain the updated predictive a posteriori P(H\q, r). With this computational updating, the service provisioning environment is predictable.

7. Framework Implementation

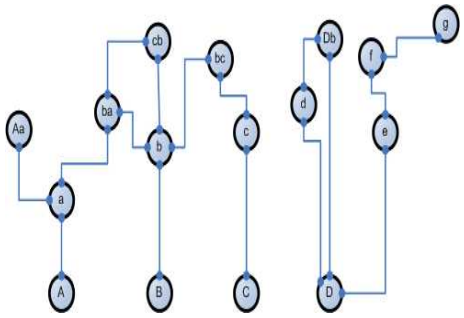

Figure 2. *Casually related set of services connected to the framework [17]*

Fig.2 defines a set of A, B, C, D services which are connected to our framework at the point of s_1, s_2, s_3, and s_4 of the SDE respectively. All service interconnection is defined by a causal relationship. This causal relationship can be affected by a failure need of a change in service resulting from either unavailable infrastructure, client crash, service failure, server crash, session failure and component failure at any point in time. Service a, ba and Aa are causally related to service A, while service b, ba, cb and bc are causally related to service B. Also service c, and bc are causally related to C while d, e, Db, f, g are causally related to D. Service maintenance is costly and difficult. It is not always clear what the impact of any type of change to service will have across the whole services. This CIA technique shows the maintainer what the effect of any change will be on the system. Our framework has proven to offer the potential to improve the stability and efficiency of service provisioning and cut the cost of maintenance. The change propagationframework was implemented and the results obtained for a hypothetical period of 3 years are shown in table 1.

Table 1. *Experimentally Obtained Results*

Main Services	Linked Services	1st yr. (X, y)	2nd yr. (x, y)	3rd yr. (x, y)
	a	0,0	0,0	1,0
A	ba	0,0	0,0	1,0
	Aa	0,0	0,0	1,0
	b	0,1	0,0	1,0
	ba	0,1	0,0	1,0
B	cb	0,1	0,0	1,0
	bc	0,1	0,0	1,0
	c	0,1	0,0	1,0
C	bc	0,1	0,0	1,0
	d	0,0	0,0	0,0
	e	0,0	0,0	0,0
D	Db	0,0	0,0	0,0
	f	0,0	0,0	0,0
	g	0,0	0,0	0,0

8. Results Interpretations

The adaptation model defines expected results to be obtained as boolean, where a value of 0 signifies no changes made, while a value of 1 signifies that there was a syntactic impact, meaning a change was effected. Values are recorded within a period of 6 months (x) and 12 months (y) respectively. Therefore for each hypothetical year, you find the first boolean value representing changes been made or not, within.

The first 6 months x and the second value representing changes that have either been made or not within 12 months (y) of the year. We have previously explained that changes propagates, hence we use the recorded values over a hypothetical period of three years, through Bayesian statistics, to predict changesfor the next year (4th year). Now with four years values at hand, a second time predictionwas made for the following year (5th year). The obtained and predicted results are as recorded in table 2a and table 2b respectively. On a general note, the consequence of obtaining the value 1 is indicative of low comprehensibility, hence low reliability. This value also serves as quality measure, as the value of 1 actually indicates low value of thequality attribute.

Table 2a. *Experimentally Obtained and Predicted Results*

Main Services	**Linked Services**	**1st yr. (x, y)**	**2nd yr. (x, y)**	**3rd yr. (x, y)**	**4th yr. (x, y)**
	A	0,0	0,0	1,0	0,0
A	Ba	0,0	0,0	1,0	0,0
	Aa	0,0	0,0	1,0	0,0
	B	0,1	0,0	1,0	0,1
	Ba	0,1	0,0	1,0	0,1
B	Cb	0,1	0,0	1,0	0,1
	Bc	0,1	0,0	1,0	0,1
	C	0,1	0,0	1,0	0,1
C	Bc	0,1	0,0	1,0	0,1
	D	0,0	0,0	0,0	1,0
	E	0,0	0,0	0,0	0,0
D	Db	0,0	0,0	0,0	1,0
	F	0,0	0,0	0,0	0,0
	G	0,0	0,0	0,0	0,0

Table 2b. *Experimentally Obtained and Predicted*

Main Services	**Linked Services**	**1st yr. (x, y)**	**2nd yr. (x, y)**	**3rd yr. (x, y)**	**4th yr. (x, y)**	**5th yr. (x, y)**
	a	0,0	0,0	1,0	0,0	0,1
A	ba	0,0	0,0	1,0	0,0	0,1
	Aa	0,0	0,0	1,0	0,0	0,0
	b	0,1	0,0	1,0	0,1	0,0
	ba	0,1	0,0	1,0	0,1	0,0
B	cb	0,1	0,0	1,0	0,1	0,0
	bc	0,1	0,0	1,0	0,1	0,0
	c	0,1	0,0	1,0	0,1	1,0
C	bc	0,1	0,0	1,0	0,1	1,0
	d	0,0	0,0	0,0	1,0	1,0
	e	0,0	0,0	0,0	0,0	1,0
D	Db	0,0	0,0	0,0	1,0	0,1
	f	0,0	0,0	0,0	0,0	0,1
	g	0,0	0,0	0,0	0,0	0,1

The attribute that is being measured here is a service change for productivity, hence quality. Productivity is an

external attribute of the service, which is clearly dependent on many aspects of the process and the quality of service delivered. Service change is a maintenance issue with service quality. Our mention of reliability isdue to the need for prediction. This is because the values are obtained on the basis of observing times between faults leading to failures during service provisioning operation, and are used as parameter estimates to make statements about future reliability. Of particular note, is the fact that reliability requires collection of inter-failure data during service provisioning operation [1], see table 1. 0From results obtained in the scenario example and recorded in table 2b, we extract the characteristics of services where fault occurs, and calculated their dependencies and the corresponding number of faults propagated. You may recall that in our CIAFAM, we mentioned that our interest was where syntactic impact is involved. That is, we concentrated on changes that have a syntactic impact; therefore, appropriate measures were based on impacts that were dependent on the static nature of the provisioning system. We therefore represent the details in the following table 3 showing the actual impact set.

We represent the details of table 3 in a dependency – fault propagation relationship as shown figure 2.

Table 3. *Example Scenario Dependency – Fault PropagationCharateristics*

Linked Service Fault Source	No. of dependency	No. of fault Propagated
B	4	5
A	3	2
C	2	1
d	1	0
Db	0	0

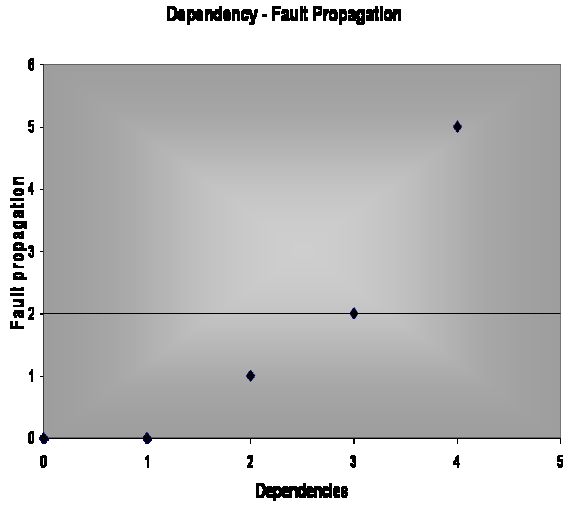

Figure 2. *Dependency-Fault Propagation Characteristics*

The propagation process is necessary because if not carefully controlled after a change, itmight result in an avalanche of faults which may increase the rate of inconsistencies knowing that the goal of change propagation is to ensure consistency after a change. As it isnot enough to check the level of propagation only, but also necessary to check what impact these changes have on the overall service which will give the actual impact set. Therefore, from the actual impact set, we obtained the number of changes and compute their corresponding impact as expressed in table 4 below:

Table 4. *Example Scenario Change – Impact Charateristics*

Sources of Fault	No. of changes	No. of impacted Services
b	4	5
a	3	4
c	2	3
d	1	2
Db	0	1

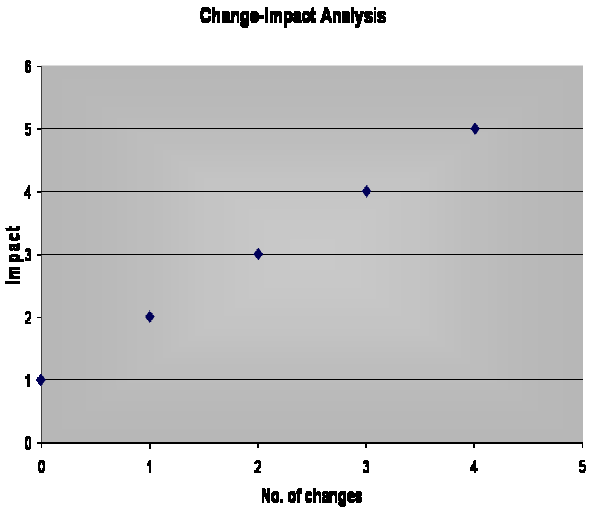

Figure 3. *Change – Impact AnalysisCharacteristics*

Figure 3 below shows the change – impact Analysis characteristics derived from table 4.

Maintenance has been recognized as the most costly phase in the software life cycle [14]. Since software has been consumed as services, service maintenance effort has been estimated to be frequently more than 50% of the total life cycle cost [15]. This work has the potential to improve service provisioning to customers, thereby cutting cost during service delivery. Using change propagation framework will help to achieve the following:

(i) Understand the nature of the services needed by a consumer.
(ii) Estimate the effort devoted to a project.
(iii) Determine the quality of service.
(iv) Predict the maintainability of service with respect to the derived benefits.
(v) Validate best practices for service providers in a frequent changing requirement community.
(vi) Provide optimal maintenance solutions.

By identifying potential impacts before making a change, the risks associated with embarking on a costly change can be reduced, because the cost of unexpected problems generally increases with the lateness of their discovery. The more a particular change causes other changes, the higher the cost. Carrying out CIA will allow an assessment of the cost of the change and help management to choose between alternative changes. It will also allow managers and engineers to evaluate the appropriateness of a proposed

modification. If a proposed change has the possibility of impacting large, disjoint sections of a service, the change will need to be re-examined to determine whether a safer change is possible [14].

9. Validity of Measures from Concepts

We acknowledge the fact that predictive measurements require predictive systems involving a model and a set of predictive procedures for determining the model parameters and applying the results [1]. Therefore, in validating our measures in the sense of assessment, we have demonstrated empirically that the representation condition is satisfied by the productivity and quality attribute being measured. We have also demonstrated the validity of the measure as it correlates with the expected values in CIAFAM. Hence the measure is a good predictor of effort in productivity and an indicator of quality of service (Q_oS) during change of a service in our grid-based environment.

We measured change impact based on the set of services that are affected by the change. Consequently, we concentrated on the number of services that are affected and their dependencies to describe the level of fault propagation. This is why we extracted only services that have a fault and their dependencies, and expressed this in Table 3 and consequently determined the characteristics of these dependencies and the corresponding fault propagation graphically as shown in figure 2. The general understanding obtained from this graph is that, the higher the dependencies, the higher the rate of fault propagated, and the greater the number of changes required to keeping the main service in a consistent state. The affected services' complexity often determines how severe the change was. The higher the number of service dependencies, the higher the level of complexity and invariably the more severe the change. Considering our example scenario, changes that do not affect any other services because their number of dependency was either one or zero were limited in scope (e.g. linked services d and Db) and therefore have zero fault propagation.

Determining the severity of a change is a function of the impact the change has on the other services. We therefore obtained the number of changes required and computed their corresponding impact as describe in table 4. The relationship between the change and impact was describe in the graph represented in figure 3, indicating that, as the number of changes increased, the impact also increased. As noted earlier, the number of changes was also affected by the number of dependent services. Therefore, if the dependencies are high, the number of changes will be high, and consequently the impact will as well be high.

10. Conclusion and Future Work

Results obtained (Table 2a and 2b) using CIAFAM, Bayesian statistics, alongside the fault and failure assumption model of the framework indicate that the framework satisfies the criteria of an accurate prediction. Although we have attempted regression based model (not reported in this paper), we thus confirm that Bayesian method is a useful technique for service maintainability prediction by achieving significantly better prediction accuracy as compared to the unreported regression method. In conclusion, the higher the service dependencies, the higher the rate of fault propagated, and the greater the number of changes required to keeping the main service in a consistent state. The affected services' complexity often determines how severe the change was. Therefore, the higher the number of service dependencies, the higher the level of complexity and invariably the more severe the change.

What is not confirmed but provides an interesting direction for future work, is whether the accuracy through the Bayesian model is dependent on the fault and failure assumption model and the change impact analysis factor adaptation model (CIAFAM). Another interesting direction would be using a Bayesian model with a different service structure (whose relationship is static not causal), for example, a tree augmented Naïve-Bayes' classifier to predict service provisioning effort.

References

[1] Fenton, N. , (1994): Software Measurement: A Necessary Scientific Basis. IEEE Transactions on software Engineering. Vol. 20, No. 3

[2] Bennett, K., Munro, M., Gold, N., Layzell, P., Mehandjiev, N., Budgne, D., Brereton, P. (2001): An architectural model for service-based flexible software. In Proceeding of the 25th Annual International Conference on Computer Software and Application (COMPSAC 2001) IEEE Computer Society, 137-142

[3] Liu, X., Yang, H., and Zedan, H. (1997): Formal Methods for the Re-engineering of Computing Systems. In Proceedings of the 21st IEEE International Conference on Computer Software and Application (COMPSAC'97), pages 409-411, Washington, D.C. IEEE Computer Society.

[4] Zhou, S., Zedan, H., and Cau, A.(1999): A framework for Analysing the Effect of 'Change' In Legacy Code. In IEEE 15th International Conference in Software maintanance (ICSM'99), pp 411.

[5] Cotroneo, D., Di Flora, C. And Russo, S.(2003): Improving Dependability of Service Oriented Architecture for Pervasive Computing. Proceedings of the 8th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems. ISBN 0-7695-1929-6/03.

[6] Davies, N., Gellersen, H. W. (2002): Beyond Prototypes: Challenges in deploying Ubiquitous Systems. In IEEE Pervasive Computing **1**(1): pp26-35

[7] Turnitsa, C. D. (2005): Extending the levels of Conceptual Interoperability Model. IEEE proceedings of Summer Computer Simulation Conference. IEEE Computer Society Press.

[8] Lee, M.L. (1998): *Change Impact Analysis of Object-Oriented Software*. Technical Report ISE-TR-99-06, George Mason University.

[9] Hao, H. (2003): *What is Service-Oriented Architecture*. CTO of SoftTouch Information Technology Pty. webservices.xml.com

[10] David, S. and Lawrence, W. (2004): *Understanding Service-Oriented Architecture*. .NET Architecture Centre. Microsoft Architect Journal, January.

[11] IEEE Standard (1990). IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY.

[12] Hayes, J. H., Patel, S. C., and Zhao, L.(2004): A Metrics-Based software Maintenance Effort Model. Proceedings of IEEE 8th European Conference on Software Maintenance and Reengineering (CSMR'04).

[13] Kabaili, H., Keller, R. K., and Lustman, F. (2001): A Cohesion as Changeability Indicators in Object-Oriented Systems. Proceedings of IEEE 5th European Conference on Software Maintenance and Reengineering.

[14] Chaumum, M., Kabaili, H., Keller, R., and Lustman, F. (1999): A Change Impact Model for Changeability Assessment in Object-Oriented Software Systems. Proceedings of IEEE third European Conference on Software Maintenance and Reengineering.

[15] Elish, M. O. and Rine, D. (2003): *Investigation of Metrics for Object Oriented Design Logical Stability*. Proceedings of 7th European Conference on Software Maintenance and Reengineering. pp.193-200.

[16] Ekabua, O. O., Olugbara, O. O. and Adigun, M. O. 2007: A Generic Change Propagation Framework to Enhance Service Provisioning in a Grid Environment. Asian Journal of Information Technology, 6(10): 1015-1019, ISSN: 1682-3915

[17] Ekabua, O. O. and Adigun, M. O. (2009): Experienced Report on Assessing and Evaluating Change Impact Analysis through a Framework and Associated Models. Journal of Information Science and Engineering. 25, 363-373.

Biography

Obeten O. Ekabua is a Professor and Departmental Chair of the Department of Computer Science in the North West University, Mafikeng Campus, South Africa. He holds BSc (Hons), MSc and PhD degrees in Computer Science in 1995, 2003, and 2009 respectively. He started his lecturing career in 1998 at the University of Calabar, Nigeria. He is the former chair of the Department of Computer Science and Information Systems, University of Venda, South Africa. He has published several works in several International and National journals, and also in several career conferences. He has also pioneered several new research directions and made a number of landmark contributions in his field and profession. He has received several awards. His research interest is in software measurement and maintenance, Cloud and GRID computing, Cognitive Radio Networks, Security Issues and Next Generation Networks.