Short Paper_____

Experienced Report on Assessing and Evaluating Change Impact Analysis Through a Framework and Associated Models

OBETEN O. EKABUA AND MATTHEW O. ADIGUN
Centre of Excellence for Mobile e-Services for Development
Department of Computer Science
University of Zululand
KwaDlangezwa, 3886 South Africa

In our evolving computing environment with heterogenously distributed information systems, products are continuously modified and changed. During this process a change to one part will, in most cases, results in changes to other parts. Therefore, in design and redesign for customization, predicting this change presents a significant challenge. Changes are required to fix faults or to improve or update products. This paper reports on the development of a change impact analysis factor adaptation model, a fault and failure assumption model and the implementation of a generic change propagation framework for evaluating and assessing utility service provisioning in a Grid service environment. While implementing the framework, data was collected for a period of 3 years which helped in predicting an immediate year. The obtained results from our pre-diction shows the framework, its associated models and Bayesian statistics as satisfying the criteria for a significant prediction accuracy in evaluating and assessing the effect of a change of service in a grid environment when compared to an unreported regression method.

Keywords: change impact analysis, service provisioning, software metrics, service maintenance, Bayesian statistics, grid environment

1. INTRODUCTION

A promising approach for developing dynamic and heterogenous service provisioning environments has emerged, through the combination of distributed object computing, component based computing and web-based concepts into what is now known as Service Oriented Architectures. This technology evolution, combined with 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 [1-3].

Computing systems' evolution (hardware and software) can be traced to changes in the original requirements, different hardware platform adoption and efficiency improve-

_____

ment. Maintenance management approaches indicate different possible changes during the maintenance process and this is seen as an indication of evolutionary changes. Due to the complexity involved, error probability is high and some of these errors could have 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 [4]. The use of formal methods is crucial as it enhances system understanding to the point of unfolding undetected propagating changes [5].

System level interoperability and dependability are important issues resulting from the integration of different technologies and middleware into the same distributed systems [6]. To effectively study issues of interoperability 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 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. CHANGE IMPACT ANALYSIS

An impact is the effect or impression of one thing on another. Impact can be thought of as the consequences of a change. Impact analysis (IA) is used to determine the scope of a change request as the basis for accurate resource planning 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 quailtative and quantitative effects of that change on other part of the item are the major concern of the study of CIA [7].

Experience has shown that a comprehensive up-front analysis of requirement during software development pays high dividend by reducing the risk of costly rework 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.

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: (i) comprehend the state changes that occur in the assumptions and constraints that each is making over time, and (ii) are able to take advantage of those changes. We are specifically interested in 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 endemic to software artefacts and the services provided by these artefacts. When a change is effected in a particular service connected to grid, it is often difficult to determine the propagation of this service changes. We therefore present a change propagation framework shown in Fig. 1 to support change automation in any grid engineering methodology.
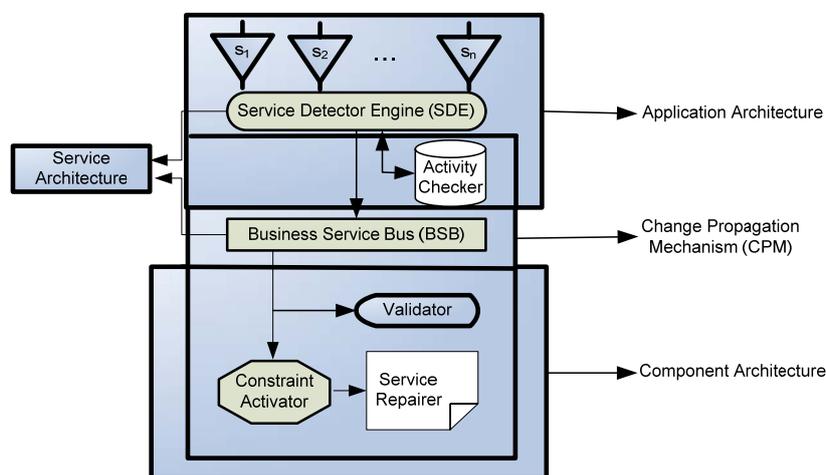

Fig. 1. Generic change propagation framework.

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) [8] 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 invoked 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.

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 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 unforeseen 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 inorder 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 generic 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 what services do I need?, what services are available to me?, what alternative services are available?, what services will operate together and what services are connected to me [9]. Our framework is generic because it can be applied to a general service provisioning engineering methodology 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)

When a change of service is considered in a system, it becomes ideal to identify 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 to account 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 a given change K in a service P, 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:

$$IAF(K, P) = A * (\sim\rho) + A'$$

where *, +, ~ denotes the usual Boolean operators: conjunction, disjunction and negation respectively

K = a given change
P = a given service
A = there is an association between K and P
ρ = K is derived from the change service
A' = there is an occurrence of aggregation link between K and P
IAF = Impact Analysis Factor

This expression implies that a service in association (A) with K and not derived ($\sim\rho$) from the change service K or services that are in aggregation link (A') with K, 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 the static nature of the provisioning system. This implies that impacts have a likelihood of propagation [10, 11].

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:

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

where

a_l = 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:

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

Meaning that the system fails if at least one of the identified failure occurs. Our failure is recorded as a Boolean value (0, 1) with respect to t_p.

Increasing redundancy degree may lead to increase in possible sources of failure resulting in a potential decrease in dependability [1]. 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:

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

where a_l and t_p are as previously defined and $ss_p = t_m, i, d$

t_m = time (in months) when a fault is detected
$t_m = x, y \ \{0 \geq x \leq 6; \ 6 \geq y \leq 12\}$ 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. EVAUATING AND ASSESSING WITH BAYESIAN STATISTICS

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 evaluation and assessment in a service provisioning environment can be computationally intensive and NP-hard in its algorithmic implications. Evaluating and assessing 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 proactive decision making.

We express the Bayesian approach as

$$P(H \backslash q, r) = \frac{P(H/r)P(q/H, r)}{P(q/r)}. \tag{1}$$

The term $P(H \backslash q, r)$ 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 $P(H\backslash q, r)$ is calculated using the chain rule of probability

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

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

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

With Eq. (3), the Bayesian environment Structure then encodes the assertion of conditional independence in Eq. (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:

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

Upon substituting Eq. (4) term maximum likelihood into Eq. (1) and computing using other terms, the updated predictive a posteriori $P(H\backslash q, r)$ is obtained. With this computational updating, the service provisioning environment is predictable.

7. FRAMEWORK IMPLEMENTATION

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 generic framework has proven to offer the potential to improve the stability and efficiency of service provisioning and cut the cost of maintenance. The change propagation framework was implemented and the results obtained from a hypothetical period of 3 years were as recorded in Table 1.
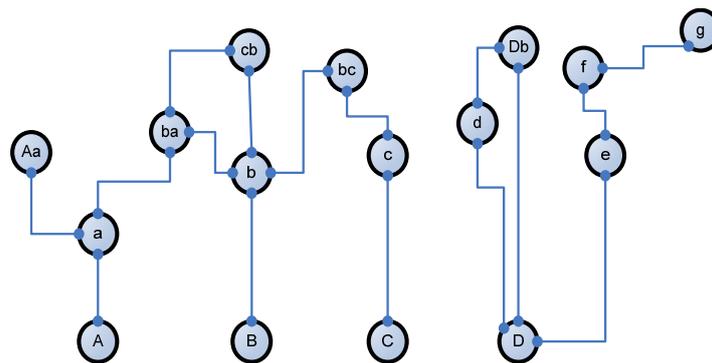
Fig. 2. Casually related set of services connected to the framework.

Table 1. Experimentally obtained results.

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

8. RESULTS ANALYSIS

Since from our adaptation model we have defined the results as Boolean, 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. The values are recorded within a period of 6 months (x) and 12 months (y) respectively. So for each year, you find the first Boolean value representing changes made or not within the first 6 months (x) and the second value representing changes made or not within 12 months (y) of the year.

Since changes propagates, we use the recorded values over the period of three years, through Bayesian statistics, to predict the effect of this propagation that will require changes for the next and subsequent years. Our predicted result is as recorded in Table 2.

Maintenance has been recognized as the most costly phase in the software life cycle [11]. 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 [12]. This work has

Table 2. Experimentally obtained and evaluated results.

Main Services	Linked Services	Year 1 (x, y)	Year 2 (x, y)	Year 3 (x, y)	Year 4 (x, y)	Year 5 (x, y)	Year 6 (x, y)
A	a	0, 0	0, 0	1, 0	0, 0	0, 1	0, 0
	ba	0, 0	0, 0	1, 0	0, 0	0, 1	0, 0
	Aa	0, 0	0, 0	1, 0	0, 0	0, 0	0, 0
B	b	0, 1	0, 0	1, 0	0, 1	0, 0	0, 0
	ba	0, 1	0, 0	1, 0	0, 1	0, 0	0, 0
	cb	0, 1	0, 0	1, 0	0, 1	0, 0	0, 1
	bc	0, 1	0, 0	1, 0	0, 1	0, 0	0, 1
C	c	0, 1	0, 0	1, 0	0, 1	1, 0	1, 1
	bc	0, 1	0, 0	1, 0	0, 1	1, 0	1, 1
D	d	0, 0	0, 0	0, 0	1, 0	1, 0	1, 1
	e	0, 0	0, 0	0, 0	0, 0	1, 0	1, 1
	Db	0, 0	0, 0	0, 0	1, 0	0, 1	1, 1
	f	0, 0	0, 0	0, 0	0, 0	0, 1	1, 1
	g	0, 0	0, 0	0, 0	0, 0	0, 1	1, 1

the potential to improve service provisioning to customers, thereby cutting cost during service delivery. Using change propagation framework will help to achieve the following:

- Understand the nature of the services needed by a consumer.
- Estimate the effort devoted to a project.
- Determine the quality of service.
- Predict the maintainability of service with respect to the derived benefits.
- Validate best practices for service providers in a frequent changing requirement community.
- 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 [11].

9. CONCLUSION AND FUTURE WORK

Based on the results obtained (Table 2) through the use of Bayesian maintainability prediction model, alongside our fault and failure assumption model for the framework, we can say 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. What is not confirmed but provides an interesting direction for future work, is whether the accuracy through Bayesian model is a dependent function 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. D. Cotroneo, C. D. Flora, and S. Russo, "Improving dependability of service oriented architectures for pervasive computing," in *Proceedings of the 8th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems*, 2003, pp. 74-81.
2. N. Davies and H. W. Gellersen, "Beyond prototypes: Challenges in deploying ubiquitous systems," *IEEE Pervasive Computing*, Vol. 1, 2002, pp. 26-35.
3. K. Bennett, M. Munro, N. Gold, P. Layzell, N. Mehandjiev, D. Budgne, and P. Brereton, "An architectural model for service-based flexible software," in *Proceedings of the 25th Annual International Conference on Computer Software and Application*, 2001, pp. 137-142.
4. X. Liu, H. Yang, and H. Zedan, "Formal methods for the re-engineering of computing systems," in *Proceedings of the 21st IEEE International Conference on Computer Software and Application*, 1997, pp. 409-411.
5. S. Zhou, H. Zedan, and A. Cau, "A framework for analyzing the effect of 'change' in legacy code," in *Proceedings of IEEE 15th International Conference in Software Maintenance*, 1999, pp. 411.
6. C. D. Turnitsa, "Extending the levels of conceptual interoperability model," in *IEEE Proceedings of Summer Computer Simulation Conference*, 2005, pp. 56-69.
7. M. L. Lee, "Change impact analysis of object-oriented software," Technical Report No. ISE-TR-99-06, George Mason University, 1998.
8. H. Hao, "What is service-oriented architecture," O'Reilly Media, Inc., 2003, http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html.
9. D. Sprott and L. Wilkes, "Understanding service-oriented architecture," *Microsoft Architect Journal*, 2004, http://msdn.microsoft.com/en-us/library/aa480021.aspx.
10. H. Kabaili, R. K. Keller, and F. Lustman, "Cohesion as changeability indicators in object-oriented systems," in *Proceedings of IEEE 5th European Conference on Software Maintenance and Reengineering*, 2001, pp. 39-46.
11. M. A. Chaumum, H. Kabaili, R. K. Keller, and F. Lustman, "A change impact model for changeability assessment in object-oriented software systems," in *Proceedings of IEEE 3rd European Conference on Software Maintenance and Reengineering*, 1999, pp. 130-138.
12. M. O. Elish and D. Rine, "Investigation of metrics for object oriented design logical stability," in *Proceedings of the 7th European Conference on Software Maintenance and Reengineering*, 2003, pp. 193-200.

Obeten O. Ekabua holds B.Sc. and M.Sc. degrees in Computer Science in 1995 and 2003 respectively and has since 1998 been lecturing in the University of Calabar, Nigeria. He started his M.Phil./Ph.D. degree in London South Bank University in 2005 and transfer in 2007 to the Department of Computer Science at the University of Zululand, South Africa. His research interest is software measurement and maintenance. He particularly considers issues of change impact analysis for service provisioning environment. He has made immense contribution on national issues relating to his area of specialization and has presented papers at national and international conferences.

Matthew O. Adigun received his Ph.D. in Computer Science at Obafemi Awolowo University, Nigeria in 1989. He is currently the Professor and Head of the Department of Computer Science, University of Zululand, South Africa, a position he has held since 1989. His research interests are in Software Engineering and Architecting of Mobile and Pervasive Systems. He has presented papers at national and international conferences in his and related areas of research interests. As a principal investigator, he has led research in the National Research Foundation's Research Niche Area titled Software Infrastructure for e-Commerce and e-Business with a group of CS, IS and Business Management researchers from inside and outside of the University of Zululand, South Africa.