An Approach for Comparison of Architecture Level Change Impact Analysis Methods and their relevance in Web Systems Evolution

Zafar Mehboob, Didar Zowghi, David Lowe
Faculty of Engineering and Information Technology
University of Technology, Sydney,
Australia
{zafar,didar}@it.uts.edu.au, david.lowe@uts.edu.au

Abstract

Change impact analysis (CIA) methods have been developed to identify the consequences of making changes to system artifacts and to support decision making with regards to that change. There is a growing body of research on CIA methods that specifically addresses changes and their impacts at a system architecture level. Most of the methods have been developed and validated on software system domain. However, there is little research consensus on: (i) the features that architectural CIA methods should comprehensively address; and (ii) which existing methods are comparatively suitable in a particular system domain such as Web systems. This paper presents a comparison approache that offer guidance on the selection of the most appropriate method for CIA activity and suitability of these methods in the context of Web systems.

1. Introduction

Change impact analysis (CIA) is a crucial part of the change management process, as software systems are generally exposed to changing requirements. Bohner and Arnold [1] define impact analysis as 'identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change'. This definition is general and talks about the scope of a change not limited to any development phase or activity. Most research about impact analysis focuses on program code level, although CIA undoubtedly plays an important role in the entire system life cycle such as for requirements [2], architectural design [3, 4] and testing phases [5].

The problem of developing and maintaining complex and evolvable systems has led to a realization that architecture description can play an important role in successfully understanding and managing complex and volatile systems [6]. Researchers and practitioners have also recognized that quality attributes such as maintainability of systems are mainly constrained by the architecture [7] and this is why it is also crucial to specifically investigate change impacts during architecture design. To address this concern, there are different CIA methods developed specifically to address changes and their impacts at system architecture level. However, it has been argued that the phenomena of system evolution are not universal and vary according to systems domain [8] and thus a developed method may not be suitable for all other domains except the one which it is developed for. Given the importance of architecture level CIA and possible variation of these methods in different system domains, there is surprisingly little consensus on (i) the features that an architecture level CIA method should address and (ii) on the suitability of these methods in a particular system domain such as Web systems.

A comparison of existing methods can better inform consideration of the desirable features of CIA methods and their usefulness in a specific system domain (an explanation of system domain are given in section III A). We have developed an approach to compare different architecture level CIA methods. The main contributions of this paper are (i) to present a comparison approach based on the features that architecture level CIA methods should comprehensively address in order to offer guidance on the selection of the most appropriate CIA method. By utilizing these comparison results (ii) to further discover the shortcoming of these methods with respect to the Web systems domain and to better inform on the suitability of these methods in Web systems domain. An initial work for the comparison of CIA methods has been done at program code level and focused more on

cost-precision tradeoff [9]. Similarly another comparison of dynamic CIA methods has been done to specifically focus on the scalability feature of selected methods at code level [10]. Both of these comparisons are limited in their application at the program code level only and exclusively address the tradeoff analysis such as cost-precision, time-space overhead and scalability aspects. Our comparison approach is different in three ways. Firstly, it is concerned with the study of architecture level CIA methods. Secondly, it is based on the features that these methods should address and more importantly illustration of these features as selection criteria and lastly it offers guidance on the suitability of these methods in Web systems domain. Furthermore, we have developed our comparison approached based on a set of evaluation elements and categorized these elements into four components of NIMSAD framework [30].

Given the importance of architecture, there are surprisingly few methods developed to assess the changes and their impact at architecture level. For our comparison, we have selected three most representative architecture level CIA methods that largely encompass other available methods. These three architecture level CIA methods are based on design rationale [4, 11], architectural slicing [3], and crosscutting in architectural design [12] respectively. In the rest of the paper we will refer to these three methods as method 1, method 2 and method 3 respectively. In the next section of this paper, we will discuss the nature of the Web development process in general and characteristics of Web systems architecture in particular. Section 3 presents our approach for methods comparison and its components. Section 4 provides an overview of architecture level CIA methods and application of our comparison approach. Section 5 provides discussion and comparison results. Conclusions and future work can be found in section 6.

2. Characteristics of Web Systems and their relevance in CIA

There is a growing body of literature regarding the differences in the development processes of Web systems and traditional software systems. Numerous authors have reported that the Web development process is different from traditional software processes in many ways [13-15]. These differences include aspects such as: co-evolution of business and solution under development [16] and rapidly changing requirements [17]. Furthermore, the architecture of Web systems tend to be characterized by a tighter linkage between business architecture with both complex information architecture and a highly component-based technical architecture [18]. In this paper we will specifically focus on two distinct characteristics of Web systems including (i) co-evolution of business processes and solution under development (ii) tighter linkage between business model and technical architecture. Subsequently we will report the shortcomings of current architecture level CIA methods while addressing these characteristics.

2.1. Co-evolution of business processes and solution under development

In traditional software development, where architecture design is typically preceded by requirement elicitation, change identification and analysis of architecture (such as functional and system architecture) are well supported by suitable CIA methods. Conversely in Web systems, business needs and architecture solutions both co-evolve mainly as a result of lack of domain understanding and domain uncertainty [19]. Catering for the changes and their impacts early when business processes and supporting architecture is emerging, current CIA methods do not seem adequate for identification of impacts on the architecture as a result of changing business needs.

This specific characteristic is most noticeable in highly incremental web development processes which tend to often remove the distinction between requirements specifications and architectural design [14]. The iterative/incremental development in Web systems, however, is intended mostly not to evaluate solutions against a known set of business needs (as it is the case with traditional software systems) but rather to actually help the client and let developers formulate architecture design solution [20]. As a consequence, many of the requirements are actually captured during architectural design [21].

The reliance on an architectural solution/ specification and the incremental development indicates that we need to support high degree of cohesion between changing business needs and their possible impact on supporting architectural. Any flaw while analyzing the impacts of a change from business models to architecture design solution (such as the inability to adequately reflect changing business needs at a suitable level of architecture abstraction) will result in derelict architecture and ultimately inadequate solutions. Furthermore, at the stage of joint exploration of business needs and supporting architecture solution, an early identification of change impact may become rather complex while addressing other characteristics of Web systems such as tight coupling between

business model and supporting technical architecture [18].

2.2. Tighter linkage between business processes and technical architecture

The architecture of Web systems is highly constrained by the technological and infrastructure supports (e.g. limitation of Web browsers, data and documents format-XML/DTD, security and availability constraints etc.) [18]. It means that there is a high degree of limitations places on the form of the architecture that a specific architecture solution may take. As a result of these constraints the architecture is much more directly related to the business needs being addressed and the resultant business models. This aspect creates a tight coupling between business models and supporting architecture. Due to relative tight coupling, changes in business models often lead to fundamental changes in the solution and potentially influences supporting architecture [15]. It has been urged that Web systems architecture must be a clear reflection of rapidly changing businesses requirements [17]. With a high degree of architecture interdependencies, Web system maintenance may become difficult and negatively effect web-based business success largely as a result of late identification of impact at latter stages of development. In general, changes in Web systems are largely as a result of high degree of volatility and frequent modifications to improve the quality and functionality they offer. The early identification of these changes and their impacts is crucial for development/maintenance and can possibly results in cost and risk reduction. Many of the secondary impacts caused by change-propagation may go undetected or identified very late while using the existing CIA methods [22].

Previous research highlights that the CIA methods are developed by both an understanding of the interdependencies between components or sub-systems [23] as well as the process [24, 25] which derives from a set of requirements through design and ultimately to the implementation. This focus may lead us to the supposition that if the system architecture or the process of development is fundamentally changed, then the CIA methods should also be changed accordingly. This is particularly relevant in the context of Web systems, given that the literature indicates that these systems often have a specific set of characteristics related to differences both in their architecture and the process through which they are developed. As Web systems become ubiquitous it has become increasingly important to be aware of the various characteristics-which are different from software systems-and hence the methods and techniques for change assessment. That is to say, unless we improve our understanding of Web systems characteristics within the scope of change analysis methods, we may not be able to utilize the real strength of change impact analysis.

The difference in the characteristics and natures of changes in Web systems motivate us to compare existing architectural CIA methods and to investigate the suitability of these methods in the context of Web systems. Our comparison approach has drawn upon a number of different previous work including the comparison framework and survey for modeling [26], design [27], analysis [28] and evaluation [29] methods. These comparison approaches are quite valuable in guiding and motivating us for the comparison of architecture level CIA methods.

3. Description of comparison approach

Based on the characteristics of Web systems and the lack of adequate research with a focus on explicitly comparing architectural CIA methods, we propose a comparison approach that is based on two sources. The first is a well known framework for comparing information system methods: the NIMSAD (Normative Information Model-based Systems Analysis and Design) framework [30]. This framework for method comparison has also been widely used in other contexts including product line development [27] and software engineering [28, 29]. Since the NIMSAD framework uses the entire problem solving process as the basis for comparison [30], we believe it can be used to evaluate methods in any category. According to NIMSAD, there are four essential components for method comparison. Firstly, the method is evaluated from the perspective of the problem situation, i.e. the method context. The second component is the problem solver, i.e. the user/stakeholder of the method. The third component is the problem solving process, i.e. the method itself. The last component is the validation of the method. The second source for our approach is the applications of NIMSAD framework for the comparison of component-based system development methods [31], architectural evaluation methods [29], and architectural analysis methods [28]. Before discussing and describing each component and elements of our approach, it is important to define other key concepts i.e. architecture and change to architecture.

A commonly used definition for architecture is proposed [32] as *'The software architecture of a program or computer is the structure or structures of the system, which comprises software components, the*

Table 1. Criteria for comparison of Architecture level CIA methods

NIMSAD Components	Elements of Comparison Approach	Descriptions
Context	System Architecture definition	Does the method explicitly considered a particular definition of System architecture?
	Specific goal	What is the particular goal(s) of the method?
	Applicable stage	Which is the most appropriate development phase to apply the method?
	Input & output	What are the inputs required and outputs produced?
	System domain	What is the system domain where method mostly applied?
Stakeholder	Stakeholders' involvement	Which stakeholders are required to participate in the CIA activity?
	Process support	How much support is provided by the methods to support other processes?
Contents	Method's activities	What are the activities to be performed and in which order to achieve the goals?
	SA description	What form of SA description is required?
	CIA approaches	What types of CIA approaches are used by the method?
	Tool support	Are there tools to support the method?
Reliability	Maturity of method	What is the level of maturity of method?
	Validation of method	Has the method been validated?

externally visible properties of those components, and the relationships among them.' This definition emphasises on system structure and we will also use the same concept of architecture in this paper. A change to architecture could refer to the addition, deletion or modification to the underlying architecture entities, component, connectors (relationships) and links from one component to other components, or a combination of these factors [33]. We have based our comparison approach on NIMSAD framework and adopt the basic criterion (i.e. context, stakeholder, content, and validity). Additionally, the elements of each component are drawn from other sources including extensive literature review of CIA methods, similar research for methods comparison and desirable features as reported by CIA researchers. We have categorised these elements and mapped them to four components of NIMSAD. Furthermore, we have provided a short description of each element as shown in table 1 and covered in section 3.1. We believe that an architecture level CIA method should address these elements as desirable features of a method.

3.1. Context

Architectural definition- A well defined description of architecture is important, as it is substantially related to the scope of the method. Most of the methods do not specifically provide any concrete definition for architecture. *Methods goal* - The goal of the method is important while attaining maximum benefit of a method, guiding user during decision making and selecting desired goal(s) from a set of objective. CIA can be performed for a number of goals. Mainly it helps to reduce the cost and risk [1, 2] caused by unanticipated impacts. At architecture level, the goal of CIA method is to determine architecture artefacts affected by a change and thus facilitate tracing the change effect to other artefacts.

Applicable stage- Architecture level CIA is mostly performed before the implementation phase, or after the development of architecture design (for example adding quality attribute after development [34]). Previous research reported three significant stages for architecture evaluation including early, middle, and post-deployment [28]. These stages are typically mapped to (i) initial high-level architectural design (early), (ii) during the elaboration of the architecture design (middle), (iii) and after complete system design, its implementation and deployment (post-deployment) [28]. We believe that architecture level change assessment should be performed as soon as potential changes are being identified. This may facilitate CIA activity at early stages of development life cycle and can significantly reduce the maintenance cost at later stages of development [22].

Input and output- As a part of process activities, inputs and outputs are important to specify the starting point and result of a method. Most of the methods require some form of formal, informal or Architectural Description Languages (ADL) based on architecture representation and provide impacts set as output.

System domain- By system domain, we mean a collection of problems that have something in common, usually (but not always) the nature of the problem [35]. Previous researchers have reported different system domains (e.g., avionics, mobile robotics, software systems, web systems, method

engineering, and languages development) along with their specific requirement, architecture, development methodology and languages to address specific domains requirements [35]. Given the fundamental difference that system domains have [36], there is an intrinsic need to first adequately understand the characteristics of that domain whilst selecting any method.

3.2. Stakeholder

Involved stakeholder- A stakeholder is an entity (person or organization) that can be affected by the results of software development project [37]. It is evident that stakeholder may influence architectural change decision in various ways [38]. For a successful CIA method it is important to identify all the stakeholders along with their individual and common objectives (e.g. impacts' prioritization and decisions making [39]) during change impact analysis process.

Process support- CIA comprises several number of activities (including examine change specification, trace potential impact, perform changes and verify changes [1]), input, output and the users of the process. The process support is vital in any architecture level CIA method to provide guideline, input, output, and a sequence of steps to follow.

3.3. Contents

Method activities- In support of the process, there are usually a number of activities that a method should employ to perform CIA. The number, sequence, complexity and granularity of activities may differ from one method to another method. Most of the methods suggest at minimum a common set of activities including examination of proposed change, selection of architectural chunk/component, application of analysis approach, and implication of method results.

Architectural description- Architectural description indicates different levels of information details for communication and expression of stakeholders needs [32]. Given that most of systems are developed with different Architecture Description Language (ADL) and with different architectural views, it is evident that there is a lack of single standard of architectural modeling and recognized a number of architectural views. Therefore, it is important that an architecture level CIA method should provide specific details of required architectural description.

Change impact analysis approaches- There are two basic approaches of CIA methods referred in research literature, one is dependency analysis and another is traceability analysis [1]. However, at architecture level, many methods employ a relatively fine-grain analysis approach such as static or dynamic dependency analysis [40, 41]. Broadly, dependency analysis refers to the analysis of system artefacts at the same level of abstraction (e.g., source code to source code or design to design) and attempts to assess the effects of a change based on the semantic dependencies (static or dynamic) among system entities.

Tool support- Automation and tool support for CIA are essential while solving some important problems such as change propagation, inconsistency detection, traceability and dependency analysis [42]. Mainly tools are used to reduce maintenance effort and execution time [1] while performing CIA activities.

3.4. Reliability

Maturity of method- Maturity of a method can significantly influence its selection from a set of alternatives. Therefore, it is one of the important criterions for any CIA method. Previous related work defines three level of method maturity as: inception, development, and refinement [28].

Validation of method- Method validation is an essential mean to encourage users and to foster their confidence on methods' usage [43, 44]. It is also important to validate a method in different domains to demonstrate their general applicability.

4. Comparison of architecture level CIA methods

In this section we will describe three architecture level CIA methods and compare them one by one with the elements of our comparison approach.

4.1. Design rationale (Method 1)

An initial method based on a structured set of architectural decision knowledge was developed to use design decisions whilst predicting change impact at an architecture level [11]. This basic method is further extended by exploring dependencies between architectural change decision and architectural elements; and then uses probability calculus to quantify these dependencies [4]. We will use the extended version of this method for the purpose of comparison due to its comprehensive approach and implementations.

4.1.1. Context. System architecture definition- Method 1 follows the IEEE 1471-2000 standard's

definition for architecture where multiple view-points of architecture is used including requirement, information, computation and engineering [45]. Specific goals- The goal is not sufficiently concrete to state for method 1. However, generic goal(s) has been mentioned by researchers such as (1) reduction of maintenance cost and (2) better prediction of CIA cost. Applicable stage- A combination of requirement and architecture change decisions are focused in method 1. More specifically, this method is applicable during and after the development of architecture design. When a change decision is made, method 1 support to analyze the underlying causes of that design change at different architectural elements including design components, data models and implementation components [4]. Input & output- Inputs of method 1 are architecture elements along with the knowledge of their design rationale (e.g. design change decision, constraints, assumptions, design alternative and criteria for design selection), whereas output is a precise set of architecture element (refined and modified after the implementation of architectural changes). System domain- Method 1 is developed and applied in the domain of traditional software systems where architecture design and modeling process is normally proceeded by business requirement process (i.e. once the requirements are sufficiently being elicited) [46]. In other systems such as Web systems most of the requirements are captured during initial architectural specification. It means that in Web systems, classically architecture design may be treated as a mean for requirements elicitation and may be a more volatile system artifacts [20]. Additionally method 1 focuses on those system domains where change identification and assessment process typically initiated at later stages of detail architecture design [4].

4.1.2. Stakeholder. Stakeholders' involvement- Method 1 assist designers and system architect to understand design decision dependencies and to carryout CIA activities. Process support- Method 1 is supported by both the architecture design process and the design reasoning process. Specifically, this method is intertwined with design modeling process to serve the dual-purpose (i) facilitates design specification and (ii) leverage design rationale representation.

4.1.3. Contents. Method's activities- There are four activities in method 1 including architectural rationalizing (ARM) to capture architecture rationale and element linkage, developing a graphical model for characterization of captured linkage (referred as AREL model), building BBN(Bayesian Belief Network) to represent AREL model, and reasoning about change impact with AREL. Architectural description- Method 1 uses the UML based notation and stereotypes both for design elements and design rationale modeling. Change impact analysis approach- Method 1 employ static dependency analysis approach to capture the relationships between architectural decisions and design elements, and to quantify theses relationships using BBN. Tool support- An integrated set of tools support is developed to support multiple activities including to facilitate architecture design process, to capture design rationale, to quantify relationship between architectural decisions and architectural elements, and to provide traceability and integration support. Method 1 employ a UML tool named as Enterprise Architect, Netica, and AREL tools [4].

4.1.4. Reliability. Method's maturity- Method 1 is in the refinement stages as it is being implemented in commercial setting. However, its implementation is limited only to software system domain at the moment. Method's validation- A few experimental and commercial implementations of method 1 have been reported for the validity purpose [4]. However, a large numbers of industrial implementations or repeated practices are necessary for a rigorous validation.

4.2. Slicing and chopping (Method 2)

Method 2 supports CIA at architecture level by utilizing architectural slicing and chopping techniques [3]. Method 2 uses a formal representation of architecture named as WRIGHT.

4.2.1. Context. System architecture definition- Method 2 does not provide any explicit definition of architecture. The perception of architecture is more focused toward WRIGHT specification and perceives it as a combination of components and connectors. The goal(s) of method 2 are not explicitly stated in corresponding research. However, as changes are being analysed early at the architecture level and it mainly do not require implementation details, therefore authors implies that method 2 is an efficient way to reduce the cost of changes management process. CIA can be performed at different phases/stages of system development. Among these stages, method 2 focuses on architectural evolution and maintenance stages. WRIGHT architectural specifications are the main input for method 2. The output of method 2 is a set of information about components and connectors that may affect, or be affected directly or indirectly by any change made to the architecture. Method 2 is developed and validated suitably in software systems

Table 3. The comparison results of architecture level CIA methods using our approach

Methods / Elements	Method 1	Method 2	Method 3
System Architecture definition	Provided	Not specifically provided	Left for user to perceive
Specific goal	Reduction in maintenance cost and better prediction of change impacts	Partially address the reduction in change management cost	Conservative impacts set
Applicable stage	Mainly during and after the development of architecture design	During architecture evolution and maintenance	During development and maintenance of architecture design
Input & output	Input: Architecture elements along with their design rationale & Output: modified architecture elements	Input: WRIGHT architecture specification Output: Set of information about impacted architecture component	Input: dependency matrix & Output: impacts set consist of two subsets: (i) set of elements that need to be changed and (ii) set of elements that need to be preserved
System domain	Software systems	Software systems	Software systems
Stakeholders' involvement	Architects and system designers	Various stakeholder including maintenance programmer, architects and project manager	Maintenance programmer
Process support	Intertwined with design modeling process	Not explicitly addressed	Not explicitly addressed
Method's activities	4 activities	3 phases whereas phase 2 and 3 comprises of 2 activities each	3 activities
System Architecture description	UML based graphical notation and stereotypes	WRIGHT architecture representation	An informal description of architecture
CIA approaches	Static dependency analysis	Static dependency analysis	Static analysis of dependencies
Tool support	An integrated tools support is available	Partially available	Not available
Maturity of method	Refinement stage	Development phase	Inception phase
Validation of method	Validated on software systems domain	Not results reported on validation	Not validated yet

domain where architecture level interdependencies and the coupling between business and technical architecture are not as complex and exaggerated as in Web systems domain.

4.2.2. Stakeholder. Stakeholders' involvement- Method 2 describes various stakeholders' roles including maintenance programmer, architecture designer and project manager. A significant involvement of stakeholders in method 2 sets up a common understanding of CIA to leverage future system maintenance. Process support- Method 2 does not provide knowledge of how to support overall process of CIA. As such the guideline on when and what actions are taken along with the inputs/outputs are typically embedded within the method metaphors.

4.2.3. Contents. Method's activities-There are three phases in method 2 including building architectural flow graphs (AFG), computing architectural slices and computing chops. Phase 2 and 3 both uses two activities each: (i) finding slices and chops and (ii) then deriving architectural slices and chops. Method 2 provides detailed information for the activities of last two phases but does not significantly explain activities to develop AFG. Architectural description- In method 2 a formal representation of architecture (WRIGHT) is required. This architectural description is used to specify system structure in term of components, connector and their configuration. Change impact analysis approach- Method 2 uses static change impact analysis approach during each phase. Additionally, WRIGHT representations are also analysed statically. Tool Support- For method 2, there is no research work reported as a tool support. However, a preliminary description of Ciasa tool [3] was described but no industrial implementations of this tool is available yet.

4.2.4. Reliability. Method's maturity- It can be inferred that method 2 is at the development phase as it

is being realized by the implementation of Ciasa [3]. Method's validation-There is no significant empirical results are reported for the validation of method 2.

4.3. Crosscutting (Method 3)

Method 3 supports CIA at the architecture level by employing cross-cutting technique. Method 3 investigates crosscutting dependencies between architecture elements and describe that it may yield a conservative impact set.

4.3.1. Context. System architecture definition- Method 3 does not clearly provide any specific definition of system architecture. However, it leaves the definition for users to perceive themselves. Specific goals- A relatively imprecise goal of method 3 is identification of a conservative impact set. This set consists of both the artefacts that need to be change and that need to be preserved. Applicable stage- Method 3 can be typically applied during the development and maintenance of system architecture. Moreover, it also provides a broad coverage while synchronising the architecture changes both with requirement and implementation changes. Input & output- The input for method 3 is dependency matrix as a starting point, whereas the output is impact set that consists of two subsets: (i) set of elements that need to be change and (ii) set of elements that need to be preserved. System domain- The domain of method 3 are those systems that typically employ change impact analysis during architecture modification and after architecture design [28]. Therefore we may assume that the nature of these systems is like that where changes in business requirements do not necessary influence on architecture during the early stage of system development. As a result during the early stages of architecture design, potential changes are usually minimal or less important to analyse.

4.3.2. Stakeholder. Stakeholders' involvement- The maintenance programmer is the main stakeholder as described by method 3. We believe that the role of designer is being played by the same maintenance programmer while updating variety of system elements including requirement, design and code elements. Process support- There is no explicit process support or any such description provided by method 3.

4.3.3. Contents. Method's activities- There are three steps described in method 3 as: (i) trace dependency relationship (i.e. dependency matrix) between the source and the target of change (ii) construct crosscutting matrix from dependency matrix and (iii) apply impact analysis algorithm. Details of each step are provided in method 3 description. Architectural description- Method 3 requires an informal description of architecture that can facilitate users adequately to depict the dependency information among architectural elements. Change impact analysis approach- Method 3 is based on static approach to analyze the dependency information, specifically from architecture design to requirements, to implementation and vice versa. Tool Support- There is no tool support available for method 3. However, authors have proposed a tool as a future work to scale method 3 in industrial projects.

4.3.4. Reliability. Method's maturity- There is no progression after the initial work done two years ago. It implies that method 3 is in the inception phase. Method's validation-There is no empirical work or case study reported for method 3. Authors have proposed validation of method 3 as a future research work.

5. Discussion and comparison results

The goals of this paper are to present a comparison approach to evaluate different architectural CIA methods to offer guidance on the selection of most appropriate method for CIA activity, and thus to discover methods shortcomings and suitability with respect to the Web systems domain. To achieve these goals, firstly we have focused on discovering similarities, differences among architectural CIA methods and their comparison as described in table 2. These comparison results better inform consideration of desirable features of architectural CIA methods and offer guidance on the selection of most appropriate method. Secondly, we have demonstrated how our comparison approach (based on the features that a CIA method should comprehensively address), can be used to identify possible shortcomings of architectural CIA methods whilst addressing the specific characteristics of Web systems (as shown in table 3).

From table 2, the comparison results reveal that several elements are supported by most of the methods. Almost all methods specify either precise or an imprecise goal(s), applicable stages, inputs/outputs and systems domain in which method should be applied. There is no standard definition of software architecture available for all the methods to follow. Method 2 and method 3 do not specifically provide any definition of architecture and mainly left it for users understanding. Method 1 has adopted IEEE 147-2000 definition for architecture. The involvement of stakeholder(s) is also duly addressed in all three methods. However, the roles

of stakeholders vary from one method to another. The process support is only adequately described in method 1. Method activities, architecture description, CIA approaches are clearly describe in all three methods. However, method 3 does not provide adequate details about the architecture description as a starting point for the method. All these methods also reveal that none of the available tools alone can support method automation. Therefore, method 1 employs an integrated set of tools for implementation coverage. Most of the methods investigated in this paper are at the inception or refinement phase of their maturity. This may be due to lack of adequate industrial experiments, empirical works or case studies to validate these methods.

Table 3. The relevance of architecture level CIA methods in Web systems domain

Web systems' Characters-tics / Methods	Co-evolution of business process and solution under development	Tighter linkage between business process and their supporting architecture
Method 1	Focus only on detail design level	Discuss functional/system level architecture inter-dependencies
Method 2	Primarily discuss detail design level change and their impacts	Does not address architecture and business coupling
Method 3	Not explicitly addressed	Not explicitly addressed

Method validation encourages users to select a method from a set of alternatives [44]. This means that it will be beneficial to validate a method by employing them in a number of system domains to exhibit their broad usability. However, our approach implies that most of the methods are neither characteristically developed nor adequately validated to other system domains except software system domain as described in Table 2 (see row 6). As discussed in section II that the specific characteristics of Web systems differentiate these systems from software systems, and these characteristics are important to address while selecting any architectural CIA method in the context of Web systems. The suitability and possible shortcomings of three architectural CIA methods whilst addressing the specific characteristics of Web systems are shown in table 3. It has also been described in table 3 that method 1 and method 2 primarily focus on detail design level change impacts and do not adequately address the change impacts resulting from the co-evolution of business process and their supporting architecture. Additionally, both of these methods support change impact identification at functional and system architecture and therefore they may overlook impacts arising due to the tighter linkage between business process and architecture. Whereas, method 3 neither explicitly supports the identification of change impact due to co-evolution nor due to tight coupling between business process and their supporting architecture. It has been reported that as a result of this tight coupling, a single change in business process can often leads to fundamental impact on architecture [18]. Based on the above supportive arguments, it can be deduced that current architectural CIA methods do not adequately address the complex nature of change and architecture interdependencies at early stage- where business process and supporting architecture solution both emerges together in Web systems domain. Conversely, most of the methods studied in this paper support software system domains where change identification and their assessment typically initiated at later stages of detail system design. As we have discussed in section II that change impacts resulting from co-evolution of architecture design and complexity of impacts need to be addressed early in Web systems development before it becomes prohibitively expensive to address them. Furthermore, It has also been reported that the problems of maintaining Web systems have led to a realization that architecture evaluation at early stage of architecture design can play an important role in successful evolution and maintenance of these systems [34]. Tables 2 and 3 better inform us on the suitability of architectural CIA methods in Web system domains (based on the elements of comparison approach- where the system domain is an important criterion for the selection of architectural CIA method). Additionally, these results also assist web developers to make well informed decisions while recognizing the possible shortcomings of existing architectural CIA methods.

The discussion from section IV reveals that most of the architecture level CIA methods developed specifically to address software system domain and therefore are suitable to apply in that domain. It will be worthwhile to recognise that these architecture level CIA methods may not be suitable to apply in other system domain such as Web systems. This is largely true due to the potential differences in the development process, characteristics of architecture and nature of architectural changes in Web systems. However, we believe that current architectural CIA methods can be

extended to specifically address the differences that Web systems have as a separate system domain. A workable architecture level CIA method for Web systems should be able to capture the changes that arise as a consequence of co-evolution of business process and system under development. Similarly, a method should also adequately address the complex interdependency and multi-dimension interactions at early stage of architecture design resulting from tighter linkage between business processes and their supporting architecture [17].

6. Conclusions and future work

In this paper we have presented an approach for comparing different architecture level CIA method and to offer selection criteria for the selection of suitable method. Given that the selection of a suitable method depends on how well the comparison elements fit into a specific need, we have also proposed the concept to understand the architecture characteristics for different system domains. This aspect leverages our perception of how method suitability may differ from software systems domain to Web systems domain.

The paucity of research focus on handling characteristics of Web systems at architecture level CIA, and based on earlier related work [14, 16, 18] we can speculate that a architecture level CIA method can be extended or customised to accommodate the specific characteristics of Web systems. Considerable work still remains to be carried out in this area, both in terms of examining further supportive arguments to extend current methods, and more exploration of other architecture level CIA methods. We formalise this as our future research work to develop a model as a possible extension of current architecture level CIA method specifically for Web systems domain.

7. References

[1] S. A. Bohner and R. S. Arnold, *Software Change Impact Analysis*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1996.

[2] M. Lindvall, "An Empirical Study of Requirements-Driven Impact Analysis in Object-Oriented Software Evolution," in *Dept. of Computer and Information Sciences*, Ph.D. thesis, University of Linköping, Linköping, Sweden, 1997, p. 252.

[3] J. Zhao, H. Yang, L. Xiang, and B. Xu, "Change impact analysis to support architectural evolution " *Journal of Software Maintenance*, vol. 14, no.5, pp. 317-333, 2002

[4] A. Tang, A. Nicholson, Y. Jin, and J. Han, "Using Bayesian belief networks for change impact analysis in architecture design," *J. of Soft. Maint.*, vol. 80, no.1, pp. 127-148, 2007.

[5] A. Orso, T. Apiwattanapong, and M. J. Harrold, "Leveraging Field Data for Impact Analysis and Regression Testing," in *Proc. of the 9th European software engineering conference*, Helsinki, Finland 2003, pp. 128 - 137

[6] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, New York: Addison-Wesley, 2002.

[7] P. Bengtsson, "Towards Maintainability Metrics on Software Architecture: An Adaptation of Object-Oriented Metrics," in *First Nordic Workshop on Software Architecture (NOSA'98)*, Ronneby, Sweden, 1998.

[8] S. Cook, R. Harrison, M. Lehman, and P. Wernick, "Evolution in software systems: foundations of the SPE classification scheme," *Journal of Software Maintenance and Evolution*, vol. 18, no. 1, pp. 1-35, 2006.

[9] A. Orso, T. Apiwattanapong, J. Law, G. Rothermel, and M. J. Harrold, "An Empirical Comparison of Dynamic Impact Analysis Algorithms," in *Proceedings of the 26th International Conference on Software Engineering*, Scotland, UK: IEEE Computer Society, 2004.

[10] B. Breech, M. Tegtmeyer, and L. Pollock, "A Comparison of Online and Dynamic Impact Analysis Algorithms," in *Ninth European CSMR'05*, Manchester, UK: IEEE Computer Society, 2005.

[11] L. Bratthall, E. Johansson, and B. Regnell, "Is a Design Rationale Vital when Predicting Change Impact? A Controlled Experiment on Software Architecture Evolution," in *Proceedings of the Second International Conference on Product Focused Software Process Improvement*, Oulu, Finland, 2000, pp. 126-139.

[12] K. v. d. Berg, "Change Impact Analysis of Crosscutting in Software Architectural Design," in *Workshop on Architecture-Centric Evolution (ACE 2006)*,Nantes, France, 2006.

[13] A. McDonald and R. Welland, "Web Engineering in Practice," in *Fourth WWW10 Workshop on Web Engineering*, Hong Kong: ACM Press, 2001.

[14] R. S. Pressman and D. Lowe, *Web Engineering- A practitioner's Approach*, New York: McGraw Hill, 2008.

[15] D. Lowe and J. Eklund, "Development issues in specification of Web systems," in *AWRE'2001: 6th Australian Workshop on Requirements Engineering*, Sydney, Australia, 2001.

[16] N. Yusop, D. Lowe, and D. Zowghi, "Impacts of Web Systems on their Domain," *Journal of Web Engineering*, vol. 4, no. 4, pp. 313-338, 2005.

[17] D. Lowe and B. Henderson-Sellers, "Impacts on the development process of differences between web systems and conventional software systems," in *SSGRR 2001*, L'Aquila, Italy, 2001.

[18] D. Lowe and B. Henderson-Sellers, "Characterising Web Systems: Merging Information and Functional

Architectures," in *Architectural Issues of Web-Enabled Electronic Business*, N. S. Shi and V. K. Murthy (Eds.), London: Idea Group Publishing, 2003, pp. 227-293.

[19] G. Sinha, "Build a Component Architecture for E-Commerce," in *e-Business Advisor*, 1999.

[20] D. Lowe and J. Eklund, "Client Needs and the Design Process in Web Projects," *Journal of Web Engineering*, vol. 1, no.1, pp. 23-36, 2002.

[21] L. Gates, "Analysis and Design: Critical yet Complicated," in *Application Development Trends*, 2001.

[22] H. Kagdi and J. I. Maletic, "Software-Change Prediction: Estimated + Actual," in *IEEE SE'06*, Philadelphia, Pennsylvania, USA: IEEE Computer Society, 2006.

[23] M. D. Jacyntho, D. Schwabe, and G. Rossi, "A Software Architecture for Structuring Complex Web Applications," *Journal of Web Engineering*, vol. 1, no. 2, pp. 37-60, 2002.

[24] B. Ramesh, J. Pries-Heje, and R. Baskerville, "Internet Software Engineering: A Different Class of Processes " *Annals of Software Engineering*, vol. 14, no. 4, pp. 169-195, 2002.

[25] Z. Walter and G. Scott, "Management issues of internet/web systems " *Commun. of ACM*, vol. 49, no.3, pp. 87-91, 2006.

[26] M. H. Alalfi, J. R. Cordy, and T. R. Dean, "A Survey of Analysis Models and Methods in Website Verification and Testing," in *ICWE'07*, Como, Italy, 2007, pp. 306-311.

[27] M. Mari, "Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, KobrA and QADA," in *Proc. of the 26th International Conference on Software Engineering*, IEEE Computer Society, 2004.

[28] L. Dobrica and E. Niemelä, "A Survey on Software Architecture Analysis Methods" *IEEE Transaction on Software Engineering*, vol. 28, no. 7, pp. 638-653, 2002.

[29] M. A. Babar and I. Gorton, "Comparison of scenario-based software architecture evaluation methods," in *(APSEC'04)* Busan, Korea: IEEE Computer Society 2004.

[30] N. Jayaratna, Understanding and evaluating methodologies: NIMSAD: a systematic framework, London: McGraw-Hill, 1994.

[31] N. Boertien, M. W. Steen, and H. Jonkers, "Evaluation of Component-Based Development methods," in *the 6th CAISE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Interlaken, Switzerland, 2001.

[32] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd Edition ed., New York: Addison-Wesley, 2003.

[33] L. Francisco, F. M. S. III, R. Furuta, U. Karadkar, and A. Arora, "Perception of Content, Structure, and Presentation Changes in Web-based Hypertext," in *HT'01*, Aarthus, Denmark, ACM Press, 2001.

[34] P. Bengtsson, N. Lassingb, J. Boschc, and H. v. Vliet, "Architecture-level modifiability analysis (ALMA)" *Journal of Systems and Software*, vol. 69, pp. 129-147, 2004.

[35] R. L. Glass and I. Vessey, "Toward a taxonomy of software application domains: history," *Journal of Systems and Software*, vol. 17, pp. 189-199, 1992.

[36] I. Vessey, "Focusing on the Application Domain: Everyone Agrees It's Vital, but Who's Doing Anything About It?" in *Proceeding of 31st Annual Hawaii Interernational Conference on System Sciences*, Kohala Coast, Hawaii, USA, 1998, pp. 187-196

[37] D. M. Ahern, A. Clouse, and R. Turner, *CMMI Distilled: A Practical Introduction to Integrated Process Improvement*, Indiana:Addison-Wesley Professional, 2003.

[38] B. Sonia, L. Chung-Horng, and F. Mark, "A stakeholder-centric software architecture analysis approach," in Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints '96) on SIGSOFT '96 workshops, San Francisco, California, United States, ACM press, 1996.

[39] P. Jonsson and C. Wohlin, "A Study on Prioritisation of Impact Analysis Issues: A Comparison Between Perspectives," in *5th Conference on Software Engineering Research and Practices in Sewden-(SERPS'05)*, Vasteras, Sweden, 2005, pp. 11-19.

[40] G. F. Dror, O. S. A. Tokunbo, B. Daniel, E. Yoav, M. Gabor, P. O. Esteban, S. Sameer, S. Karlkim, X. Minhui, and R. S. Stephen, "Fine-grain analysis of common coupling and its application to a Linux case study," *Journal of Systems and Software*, vol. 80, no. 8, pp. 1239-1255, 2007.

[41] F. Tie and I. M. Jonathan, "Applying Dynamic Change Impact Analysis in Component-based Architecture Design," in *Proc. of the 7th ACIS Int. Conf. on Soft. Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*: IEEE Computer Society, 2006.

[42] T. Mens and T. D'Hondt, "Automating Support for Software Evolution in UML," *Automated Software Engg.*, vol. 7, no. 1, pp. 39-59, 2000.

[43] E. C. Jonathan and L. W. Alexander, "Software process validation: quantitatively measuring the correspondence of a process to a model," *ACM Trans. Softw. Eng. Methodol.*, vol. 8, no. 2, pp. 147-176, 1999.

[44] S. Mary, "The coming-of-age of software architecture research," in *Proceeding of the 23rd International Conference on Sofware. Engineering*, Toronto, Ontario, Canada, IEEE Computer Society, 2001.

[45] IEEE, "IEEE Recommended Practice for Architecture Description of Software-Intensive System," IEEE Computer Society, New York, 2000.

[46] P. Soffer, "Scope Analysis: Identifying the Impact of Changes in Business Process Models," in Journal of Software Process: Improvement and Practices, vol. 10, no.1, pp. 393–402, 2005.