Computer Science and Information Systems 15(2):321–346 https://doi.org/10.2298/CSIS161130041J

Managing Software Requirements Changes through Change Specification and Classification

Shalinka Jayatilleke, Richard Lai, and Karl Reed

Department of Computer Science and Information Technology
La Trobe University, Victoria. 3086, Australia
{s.jayatilleke, R.Lai, k.reed}@latrobe.edu.au

Abstract: Software requirements changes are often inevitable due to the changing nature of running a business and operating the Information Technology (IT) system which supports the business. As such, managing software requirements changes is an important part of software development. Past research has shown that failing to manage software requirements changes effectively is a main contributor to project failure. One of the difficulties in managing requirements changes is the lack of effective methods for communicating changes from the business to the IT professionals. In this paper, we present an approach to managing requirements change by improving the change communication and elicitation through a method of change specification and a method of classification. Change specification provides a way such that communication ambiguities can be avoided between business and IT staff. The change classification mechanism identifies the type of the changes to be made and preliminary identification of the actions to be taken. We illustrate the usefulness of the methods by applying them to a case study of course management system.

Keywords: Requirements change, change specification, change classification, ontology, terminology.

1. Introduction

The inevitable development of globalization, service-oriented environments and continuous technological advances compel organizations to change their strategies and business processes to meet customer demand. In addition, there is the impact of software evolution and maintenance. Although change is an evident factor in today's highly competitive business environment, many organizations find themselves at the losing end of this game. Volatile nature of business requirements usually increases the cost of development [1-6] and also poses a threat to the project schedule [3]. Changing requirements are considered one of the main contributors to project failure [7-9]. The real problem is not the changing nature of requirements, but the lack of understanding of this volatility. Change management, therefore, is a critical task for organizations.

A preliminary version of this paper was presented at the 2013 Australian Software Engineering Conference

Requirements engineering consists of a set of core activities that are in reality interleaved and iterative [10]. Requirements change is part of this requirements engineering process and it is not a standalone activity but consists of several core activities that can be described as a process. This process begins with communicating the requirements change (change request). Successfully completing this step will result in the elicitation of the correct goals in relation to the changes (change goals), which is the next step in the process. . Understanding the change goals leads to the proper execution of the third step, which is representing the change in the system design. The second and third steps effectively assist the analysis of the requirements change to assess its appropriateness and whether it should be accepted. The final step in the process is based on the results of the analysis. Depending on the outcome, a change can be accepted or rejected. Therefore, the final outcome of the change request depends heavily on the first step. This process is iterative, usually due to the inability of management to agree to the change request and due to insufficient information. It is further hindered due to poor change communication, misinterpretation of change goals, incorrect representation of changes in the system design, discrepancies in analysing the changes, and inaccurate decision making in relation to the requested changes.

One of the key reasons for difficulty in managing change occurs at its initiation. Effective interpretation and communication change, from the customer to the development level has proved to be a challenging task [11-14]. Some literature suggests that this is due to the lack of a formal process specifying change [11, 14]. The specification method used by change originators should be understood by both business and IT personnel since it is the bridge between the change originators (users, customers, etc.) and the change implementers (system analysts, designers, developers, etc.) [15-17]. Therefore, being able to specify and understand the requirements change should make in the process of incorporating the change into the existing design or system more seamless.

In this paper, we present an approach to managing requirements change by improving the change communication and elicitation through a method of change specification and a method of classification. Change specification provides a way such that communication ambiguities can be avoided between business and IT staff. This is the first step towards better and effective management of requirements change in rapidly changing business environments. The change specification process is incomplete without classifying the changes. The change classification mechanism identifies the type of the changes to be made and preliminary identification of the actions to be taken. To aim readers to have a better understanding of the change specification and classification methods, we use a simple mail order system as a running example. Finally, we illustrate their usefulness by applying them to a case study of course management system.

A preliminary version of this paper was presented at the 2013 Australian Software Engineering Conference [18]. The following items are contained in this paper but not in [18]:

(i) a discussion on the related work to give better understanding of our methods;
(ii) a description of the overview of the methods;
(iii) a justification of the use of Goal Question Metrics (GQM) and Resource Development Framework (RDF) approach; and
(iv) to illustrate the usefulness of our methods, the results of applying them to a running example and a case study.

2. Overview of the Methods

In this section, we present an overview of our approach to managing requirements change through a method of change specification and a method of classification. Managing change begins with an understanding of what is involved in this phenomenon. But as previous studies have proven, there is no real consensus on the nature of change, rather there are disparate multifaceted views and approaches. We therefore see the need for a versatile, consolidated, solution that brings these together. Based on previous research work and also through industrial interviews described later, we were able to pinpoint the gap in change identification. There is an inadequacy in applying change identification in the practical context. Figure 1 using the IDF0 notation shows the broad layout of the methods aiming to overcome this limitation. Once a change is requested, the layout follows two steps:

1) Change specification
2) Change classification

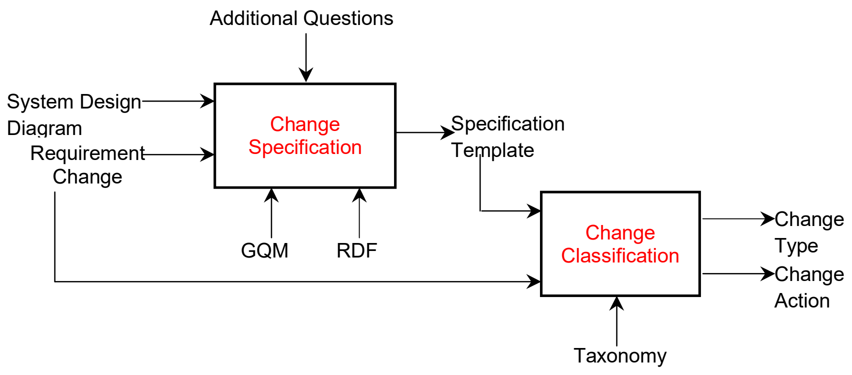

Fig. 1. Layout of overview of the methods

Change specification denotes a way of specifying a change so that communication ambiguities can be avoided between business and IT staff. Once a requirement change has been initiated from the client side, this method will use the system design diagram as an input to map out the location of the change. In order to create the specification template we will use two established methods, i.e. Goal Question Metrics (GQM) [19] and Resource Description Framework (RDF) [20]. We will also use a set of additional questions to enable better identification when using the speciation template output. The purpose of using GQM and RDF is to establish terminology and ontology (respectively) concepts in the specification method. The use of terminology will enable the specification template to have standardized terms whilst ontology will ensure a logical connection between the terms used in the specification template. The purpose of using both terminology and ontology is further discussed in section 3.2.1. The outcome of the specification template will be the identification of the location, purpose and focus of the change.

GQM approach, which was developed by Basili and Weiss and expanded by Rombach [19], is the most widely known goal-focused approach for measurement in software. One of the reasons for its success is that it is adaptable to many different organizations (e.g. Philips, Siemens, NASA) [19]. Another reason for the success of GQM is that it aligns with organizational directions and goals. Rather than using a bottom-up method (generally problematic) [21], metrics are defined top-down. This way the measurements are linked to organizational goals [21-23]. This same concept can be applied in describing change. If the changes described are linked to goals, then understanding and application of such changes could be far more efficient [24].

Introduced by Tim Berners-Lee in 1998, RDF is an ontology language for making statements about resources [20]. It was designed for describing Web resources such as Web pages. However, RDF does not require that resources be retrievable on the Web. RDF resources may be physical objects, abstract concepts, in fact anything that has an identity. Thus, RDF defines a language for describing just about anything. Furthermore software modeling languages and methodologies can benefit from the integration with ontology languages such as RDF in various ways, e.g. by reducing language ambiguity, enabling validation and automated consistency checking [25]. Given the benefits of both GQM and RDF, it was deemed appropriate to use these methods for specifying requirements changes. With these being the general benefits of GQM and RDF, their specific purpose and use in the specification method are described in detail below.

The change classification method uses the outcome of the specification template to expand on the type of change along with preliminary guidance for action to be taken in managing the change. The classification itself is based on the concepts of change taxonomy that was found in existing change management literature and refined using unstructured interviews of practitioners in the field of change management. The outcome of the change classification will provide software developers with a better understanding of what the change is and the preliminary guidance on how to incorporate the change into the existing system. We believe the combination of change specification and classification leads to a better realisation of changes requested.

2.1 A Running Example

To aim readers to have a better understanding of the change specification and classification methods, we will use a simple mail order system for CDs and DVDs as a running example which is described below.

Diskwiz is a company which sells CDs and DVDs by mail order. Customer orders are received by the sales team, which checks that customer details are completed properly on the order form (for example, delivery address and method of payment). If they are not, a member of the sales team contacts the customer to get the correct details. Once the correct details are confirmed, the sales team passes a copy of the order through to the warehouse team to pick and pack, and a copy to the finance team to raise an invoice. Finance raises an invoice and sends it to the customer within 48 hours of the order being received. When a member of the warehouse team receives the order, they check the real-time inventory system to make sure the discs ordered are in stock. If they are, they are collected from the shelves, packed and sent to the customer within 48 hours of the order being received, so that the customer receives the goods at the same time as the invoice. If the goods are not in stock, the order is held in a pending file in the warehouse until the stock is replenished, whereupon the order is filled. This process can be illustrated by the following system design diagram.

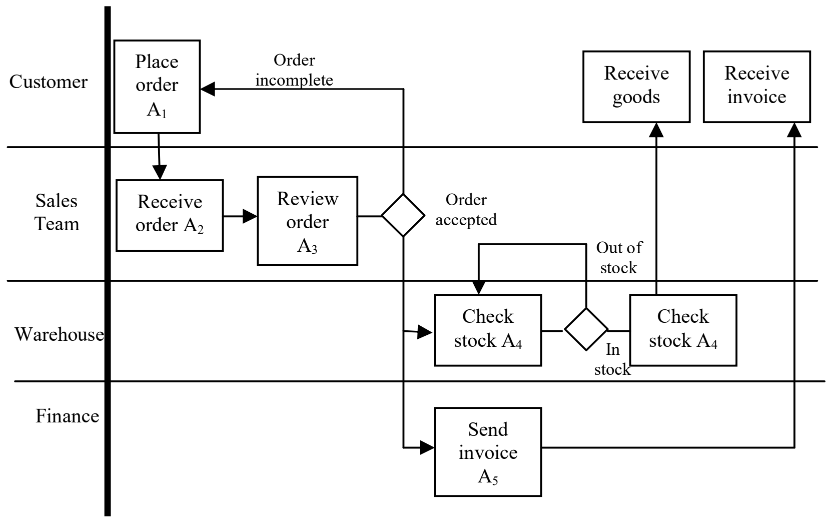

Fig. 2. Diskwiz customer order fulfillment process diagram

The example consists of a scenario where the specification method is applied in specifying the change and the change classification method is used to identify change type and corresponding action. The scenario is as follows:

The management is not satisfied with some parts of the process and point out that the following issue should be rectified: “It is identified, due to a design error, there is no communication between finance and the warehouse to confirm discs are in stock so that the order can be shipped. Therefore finance could be raising invoices when the order has not been sent.”

3. The Change Specification Method

Figure 3 represents collaboration of the different entities of the change specification method. The change specification consists of three key elements: *a system design diagram*, *a specification template* and *additional questions*. The foundation of specification component is made up of GQM and RDF. The GQM-RDF combination is a result of amalgamating ontology and terminology which in this paper, we refer to as onto-terminology. A detailed description of the onto-terminological concept and the interaction of the three elements in specifying changes are explained in the following sections. We point out that in fact, or method is “system description technique agnostic”, and, could be used in any environment where a systematic system description methodology has been used, reducing the adoption casts.

According to Figure 3, an important input is the use of system design diagrams. In this cases where the initiation of the change takes place on the business side. Therefore, the initial part of the change specification should be familiar to the business personnel involved. To achieve this, system design diagrams are used as part of the change specifying process where the notations and the language used are more business related. Any business analyst communicating a requirement change to the IT side should be capable of understanding and interpreting a system design diagram.

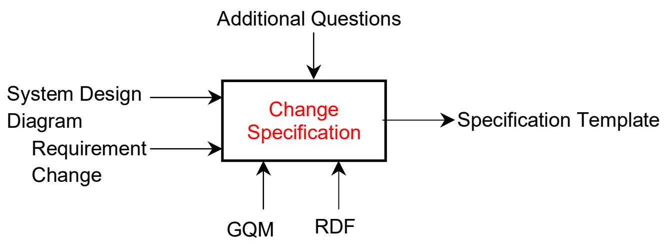

Fig. 3. Layout of the change Specification

The successful application of the change specification calls for a few key assumptions. First, the specification of changes may take place at the operational level of the organization. We believe that as changes flow from an executive level (top) to the operational level (bottom), they become less abstract, making it easier to feed the change into the specification and classification methods. Second, in reality, for a system to be stable, the changes being made are proportionately small (5% – 10%) in comparison to the complete system [26]. On the other hand, if the change requires more than a 50% change to the system, it is usually implemented in a successive release of the current system. Finally, a design diagram (preferably the system design diagram) should be available for mapping the change to the system.

3.1 Specification Prerequisites

Although there is a plethora of ways to describe change, most fall into ad-hoc methods of communication. In the authors' view, a void exists which could be filled by a more effective and efficient template and a set of guidelines that can be used to communicate requirements change. Given the current trend of business being more service-oriented, the change specification should be a bridge between customer requirements and the final product [27]. The new specification template introduced in this paper will reflect this. The following two key properties are essential for a specification method to be both functional and constructive [27].

A primary objective for the specification method is user friendliness to ensure ease of adoption. It is important to recognize that the process of specifying either requirements or changes to requirements is a human activity process [27-29]. Therefore, the method used for such specifications should be human friendly [27]. The initial response to a new method is generally resistance and an unwillingness to use it [27, 30]. This is usually because the difficulty level of the new method is unknown to the users. Also, both businesses and IT stakeholders involved in the change management process tend to trust tried and tested methods of specifying change simply because there are no "surprises" in store. For these reasons, rather than inventing an entirely new method, we have opted to use a combination of existing methods which we believe has the most desirable qualities of a specification method and with which the users are familiar. This, in our view, will minimize the short-term productivity losses associated with learning new process, and also reduce the likelihood of opposition.

The second property is the method style. Text-based specification methods are formed using either natural language or formal language [27]. Although easier to understand, the drawback in using natural language is that it may be interpreted in different ways, resulting in ambiguities. Whereas a mathematically influenced formal language may be ideal for a computer, it may not be human friendly. Therefore, it is important to find a balance in textual illustration. Also equally important is that both business and IT stakeholders involved in the process understand the specification method. To achieve this, we introduce a semi-formal method which is aided by system design diagrams.

3.2 Onto-terminology Framework

The Purpose of Ontology and Terminology. The specification method introduced in this work is a means of semi-formal communication of requirements change. And for this method to be both informative and useful, it needs to satisfy several conditions. A specification method should take into consideration: standardised terms, the usage of the terms, connotative information and linguistic relationships as well as a logical and philosophical point of view of the standardised terms [31]. We point out that these features stem from two different concepts i.e. terminology and ontology. The relationship between terminologies and ontologies has been the subject of analysis by others, as we see from the following discussions.

Terminology is a “set of designations belonging to one special language” [32]. The main purpose of using terminology in a specification method is to eliminate ambiguity and ensure the use of standard terms [31]. International standards state that the goal of terminology is to clarify and standardize concepts for communication between humans [32]. This is a crucial property of our proposals as this is a method of conveying changes in requirements from business personnel to IT personnel. However, terminology generally lacks computational representation as well as logic [33]. Of these, our concern with regard to change specification is logic. Logical accuracy will ensure that the action taken to implement the change is correct. Therefore terminology, on its own, cannot be considered for the semi-formal framework of the change specification method.

Ontologies are similar to terminologies in that both the communication of concepts. According to Gruber [34], ontology describes a concept and its relationships in a way that can be manipulated logically. The way ontology defines a concept depends entirely on the formal language used for the communication of the concept. Ontology is not a terminology [31]. In fact, ontology lacks the standardized terms and linguistic relationships of a concept which are key features in terminology [31]. These features are imperative to change specification as they build the actual form of communication terms to be used in the specification.

The conceptualization of the change specification method needs to be guided by both linguistic and logical principles. Given the strengths and weaknesses of terminology and ontology, the combination of these two concepts will provide a better framework for the specification. Onto-terminology, which results from this combination, formally defines the concept (ontology logic) as well as explains the term and its usage from a linguistic point of view (terminology).

Building the relationship between GQM and RDF. To ensure the correct combination of logic and terminology, we have selected two well-known methods where one represents terminology and the other represents ontology. A generalization of GQM is used as the linguistic function of the specification method representing terminology. It is important to note that the abstraction of GQM relates to the goal specification and not to the questions or the metrics. The purpose of using GQM is that it enables the extraction of specific terms that define the requirements change. Since these terms have been successfully utilized to extract business goals [21, 22], we found it's use satisfactory in change specification. The logical connections for the terms are sourced from RDF representing the ontology component specification. However, it can also be used to link information stored in any information source that can be ontologically defined [33].

Three terms are extracted from the goal specification of GQM that can best describe a requirement change; *Object*, *Purpose* and *Focus* (of change). The meanings of these three elements have been adjusted for the purpose of describing change. The *Object* needs to be changed due to the *Purpose* using the *Focus*. The terms extracted from RDF are *Object, Attribute* and *Value*, which is referred to as the RDF triplet [33]. The logical relationship of the RDF triplet can be stated as Object O has an Attribute A with a Value V (Professor; Reads; a Book). The rationale behind the correspondence between RDF triplet and to the GQM terms is due to the similarity and the meanings of the terms, which is described in Table 1.

Table 1. Rationale of RDF and GQM relationship

RDF term	GQM term	Correspondence	Rationale
Object	*Object*	One-to-one	Same concept
Attribute	*Purpose*	One-to-one	Both terms are activities. *Purpose* is an activity that is generated due to various business requirements.
Value	*Focus*	One-to-one	*Value* of RDF creates the significance for *Attribute* (of RDF). *Focus* of GQM creates the significance for *Object* (of GQM) by activating the term *Purpose* of GQM.

GQM terms alone could have been used if the three terms have a logical connection; and we have explained above as to why it is important to have this logical connection in a specification language. The main reason for using RDF is hence to create the logical relationship between GQM terms. Figure 4 represents the relationship mapping between RDF and GQM. As such, the logical relationships between GQM terms can be stated as *Object* O needs *Purpose* P by using *Focus* F. Given the logical connection established, any change specified (regardless of the application of the system) using the GQM terms will satisfy the requirements of a semi-formal method of communication as stipulated above (see 4.2.1). From now, we shall use these three terms in the specification method.

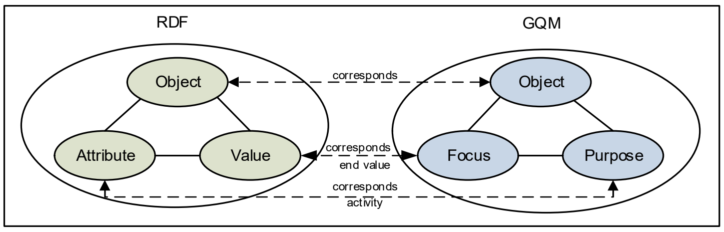

Fig. 4. RDF-GQM Relationship

The framework presented in Figure 5 is based on the above relationship and is the foundation of the specification method. The three elements OBJECT, PURPOSE and FOCUS are used to capture the requirement change. The OBJECT of change is any activity in the system design which needs a PURPOSE to change. This purpose is created as a result of changing business goals, customer requirements, etc. The object is changed by the FOCUS of change, where any change type can denote the focus. Therefore, each activity in the system design is an object, each changing business goal and customer requirement is a purpose and each change type is a focus.

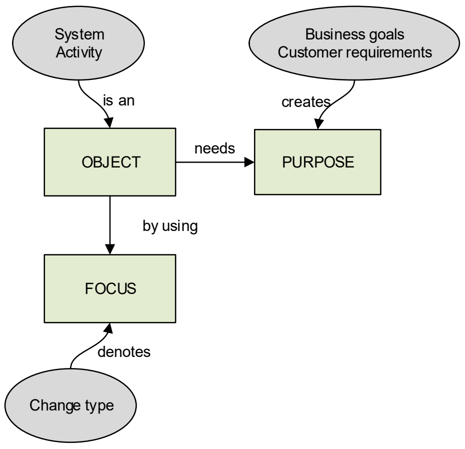

Fig. 5. Onto-terminology Framework

3.3 Text-based Specification Tool

During the preliminary studies we examined several different types of change request forms from industry to understand what information is vital for understanding a requirement change and how it was presented. We discovered two common denominators that should be included in our specification tool. First, the type of change which assists the system designers to understand the action they need to take in order to accomplish the change. Second, the reason for change which gives a better insight as to why the change was requested.

The template designed for the change specification based on the framework in Figure 4 is given in Table 2. By selecting the object of change using the system design diagram, designers and decision makers can accurately locate the main target of change, resulting in a clarification of the **location** of change. Knowing the reason for the change through the purpose ensures that change implementers are able to clarify the **need** for the change. The focus of change acts as advice on the basic implementation needed to execute the change, resulting in the clarification of the **action** of change. It indicates to the designers what to do instead of how to do the change. We believe that clearly describing the location, need and action of a change request using this template will resolve much of the existing miscommunication issues.

Table 2. Template for change specification

	Description
OBJECT	The activity name according to the system design diagram
PURPOSE	The reason for the change (can be descriptive)
FOCUS	Select from Add, Delete, Modify or Activity Relocation (description given in table 6)

An additional question (see Table 3) is used along with the above template based on the focus of change that investigates additional inputs and/or outputs required for the change. Answer to this question will be used as input for the change classification method, which is discussed below.

Table 3. List of addition questions

Focus of change	**Additional question**
Add	Need addition Input/output?
Delete	Connected to neighbor activity with input/output?
Modify	Input/output modification?
	If Yes; Input modification? Output modification?
Activity Relocation	Relocation requires input /output?

3.4 Results of Applying It to the Running Example

By applying the change specification method to the running example, we obtain the following results.

Table 4. Application of the Change specification method

	Description
OBJECT	A_4 and A_5
PURPOSE	Resolution of design error
FOCUS	Add
Additional Question	Need addition Input/output? Y

We have used the templates given in Tables 2 and 3 in order to populate the information in Table 4. It is mentioned in the change scenario that this change is required due to a design error. Therefore, the purpose of this change is listed as a resolution for a design error. The activities that are affected by the change are identified through the design diagram to be Check Stock (A_4) and Send Invoice (A_5). This is again based on the change scenario. The analyst then needs to decide with which focus this change will be executed. In this particular case, it is determined that a new activity needs to be added to handle the change. The next step is to identify if the addition of the new activity would cause new input/output between the existing activities (A_4 and A_5) and the new activity. As we are trying to bridge the communication between A_4 and A_5, based on Table 3 it is most likely that such input/output would be generated and therefore the answer to the additional question is 'Yes'.

4. The Change Classification Method

The main purpose of change classification method is to ensure that change implementers are able to identify and understand unambiguously the requirement change [11, 35]. Therefore it is essential that the classification itself is not complex. The change specification method is incomplete without having to classify the change as it provides a further understanding of the underlying causes of requirements change [35, 36]. This is the first step towards better and effective management of requirements change in this rapidly changing environment. Other studies [11, 37] also suggests that a classification of change is a scientific step to improve our ability in understanding requirements evolution.

4.1 Preliminary Studies

To explore the scope and complexity of the existing change classifications and determine the criteria for our change classification, two key investigative methods were undertaken. Firstly, a literature review of existing research on change management with a focus on change classification was undertaken. Keyword searches included change management, change classification, change types, change taxonomy, and change specification. The total result of 43 included journal papers and text books. This was filtered using selection criteria which were limited to articles referring to classification, type and taxonomy which yielded in 12 academic works [1, 3, 11, 13, 35-42]. These papers allowed us to extract the most common and regular change types used in the industry.

Secondly, unstructured interviews of 15 practitioners in the field of change management were conducted. Table 5 summarizes the important questions discussed and how they are related to this study. Respondents included project managers, business analysts, IT analysts, and software architects. Since these practitioners were from several software development organizations, the methods followed in change management was quite diverse. One of the key findings was the difficulty in relaying the business requirement change down the IT development line. A secondary related problem which arouse was the misinterpretation of the requirement change and business goal. There were many cases where parts of the final product did not meet the customer satisfaction as the changes requested had not been implemented appropriately. This justified our efforts in creating a change classification that facilitated better understanding of the requested change. We used these interviews to further confirm the change types identified through the literature survey and were able to gain better insight to improve the change classification.

Table 5. Key question of the interview

Question	Purpose
How often are changes requested and where do they originate from?	To understand the frequency of change request and where they are usually generated from
What are the types of changes that are often requested?	To identify the different types of changes
Is there a process for requesting change? If so, what are the details?	To identify the steps involved in a change request and what vital information needs to be captured
What are the difficulties in communicating change?	To understand the existing problems in the industry and what is lacking in their process of change communication
Is unambiguous communication of change important? If so, why?	To identify if there is a need for a new method of specification and classification of change

4.2 Taxonomy Development

Our classification is based on previous work-see [1, 3, 38, 39]. Table 6, demonstrates how each previous work has influenced the creation of taxonomy. However, further adjustment was made to improve the classification as mentioned above. The focus of change represents the most common forms of changes found in requirement change requests. Table 7 lists the detailed description of these basic changes. Changes *Add*, *Modify* and *Delete* were identified initially as the classification as a result of both previous literature and practitioner interviews. Change, *Activity Relocation* was included as a result of information gathered through the interviews as we discovered, is a frequent form of change requested. In normal circumstances, combinations of these basic change types can be used to represent more complicated change scenarios. These same change focuses were used in the specification method in-order to create a clear connection between the two methods.

Application of Table 7 in the classification method can be described as follows. The change focus and the answer to the additional question of the specification method will be used in the classification method as follows. For example, if ‘Add’ was selected as the change focus and the answer was ‘Yes’ to the question ‘Need additional input and/or output?’, then according to Table 4 the linking interface(s) of the new activity and the neighboring activities will mismatch. Therefore the change will be categorised under ‘Add’ change focus with ‘Mismatched links’. The 4th column in Table 6 represents the necessary action to be executed for each change type.

‘Modification’ change focus is divided into three types of change. Inner property modification will deal with modifications done to the variables and operation of an activity that does not affect its external links (input/output) to neighboring activities. Input and output data modification will respectfully affect neighboring activities linked to the input/output of the target activity as well as the internal properties of the target activity depending on the input and/or output added to it.

In ‘Delete’ change focus with ‘Matched links’, no modification is needed once the target activity has been removed. The rationale behind this action is that the deleted activity does not provide any output or take in any input from its neighbors. In contrast, with ‘Mismatched links’, once the target activity is deleted, the neighboring activities have to be modified depending on the input/output connection(s) to the deleted activity.

Activity relocation will involve moving an activity from its current location and linking it into a new location in the system design. This can be achieved in two ways. One, the activity being relocated is not linked to its neighbors through input/output and able to relocate to the new position without any modifications to the neighboring activity. Two, the target activity in the current location and the new location are affected through input/output and needs to be modified.

Table 6. Key literature used in creation of classification

Previous work	**Concepts extracted**	**Application to the classification**
Nurmuliani, Zowghi & Williams [1]	Common types of changes used (add, delete, modify) and classification of changes	Helped in creation of the most common focus types
McGee & Greer [3]	Change causes and use of experts in defining a taxonomy	Leading to different change activities and the use of change practitioners
Nurmuliani, Zowghi & Williams [38]	Categories of change	Helped in creation of the most common focus types
Xiao, Quo & Zou [39]	Primitive changes in business functions	Further expression of change types

Table 7. Detailed change description

Change focus	**Answer to Additional Question**	**Change type**	**Action**
Add	No	Matched links	Add new activity without changing the current activity or any connected links
	Yes	Mismatched links	Add new activity by changing the activity and/or connected links
Modification	No	Inner property modification	Modify the implementation of a activity without changing the connected links
	Yes	Input data modification	Modify the input link and internal properties of a activity
	Yes	Output data modification	Modify the output link and internal properties of a activity
Delete	No	Matched links	Delete activity without changing the activity or connected links
	Yes	Mismatched links	Delete activity by changing the activity and/or connected links
Activity Relocation	No	Relocation with matched links	Relocate existing activity without changing the activity or connected links
	Yes	Relocation with mismatched links	Relocate new activity by changing the activity and/or connected links

At implementation time, the key elements of the two methods (specification and classification) are incorporated into a single table (see Table 8). In the table, change number refers to the number given to each change as they are requested. The object, purpose and focus in Table 8 correspond to the information given in Table 2 i.e. activity name according to the system design diagram (this is the activity affected by the change), reason for change and select from Add, Delete, Modify or Activity relocation respectively. The additional question selected from Table 3 will be based on what has been selected for the focus and the information provided through the content of Table 2. Change type and action can be sourced from Table 7 based on the information provided for object, focus and additional question respectively. The possibility columns represent how each change may be described using different focuses. This may not apply to all changes. The ability to create multiple possibilities which will be based on the experience of the analyst and complexity of the change. This feature was added to the implementation template to provide more diversity and flexibility of communicating a change. Having multiple possibilities also provides flexibility of how the change can be implemented.

Table 8. Template for implementation

	Change No.	Possibility 01	Possibility 02	Possibility n
Specification Method	OBJECT			
	PURPOSE			
	FOCUS			
	Additional Question			
	RESULT			
Classification Method	CHANGE TYPE			
	ACTION			

4.3 Results of Applying It to the Running Example

By applying the template for implementation for the above scenario, we obtain the following result as given in Table 9:

Table 9. Application of the implementation template

Change 01	Possibility 01	Possibility 02
OBJECT	A_4 and A_5	A_4 and A_5
PURPOSE	Resolution of design error	Resolution of design error
FOCUS	Add	Modify
Additional Question	Need addition Input/output? Y	Input/output modification? Y
Result		
Change Type	Add new activity between A_4 and A_5 (Mismatched links)	Inner property modification and Output data modification A_4 and input data modification of A_5
Action	Add new activity by changing the activity and/or connected links of A_4 & A_5	Modify A_4 to send message to A_5

In Table 9, we describe the two possibilities for the scenario provided in the running example. For both possibilities, the object and the purpose remains the same and coincide with what has been discussed in Table 4. We are of the opinion that there are two ways this change can be described and the focus of each possibility demonstrates this fact. Possibility 1 was introduced in Table 4. The sections above the Results row of Table 9 is based on applying Tables 2 and 3 of change specification and were discussed in section 3.4. Based on the information provided for the Focus and Additional question, change type and action can be extracted from Table 7. This extraction is shown in Table 9, for each possibility based on the different change Focus which has been identified. In the case of Possibility 1, the Focus identified is 'Add' and the Additional question has been given an answer 'yes'. When this information is mapped to Table 7, it provides a Change type of 'Mismatched links', which requires a change Action of 'Add new activity by changing the activity and/or connected links'. When adding the new activity between A_4 and A_5, connections need to be made with both activities. Therefore, both A_4 and A_5 will be directly affected by this addition. The modification possibility of A_4 will directly affect A_5 as there will be link input from A_4 to A_5. In both possibilities, all activities that are connected to A_4 and A_5 will be indirectly affected by the alterations.

5. An Application of the Methods

Yin [43, 44] explained the usefulness of using case studies to explore the merits of an application of a research idea/ hypothesis. We therefore demonstrate the usefulness of the change specification and classification methods by applying them to a software project case study. We make two key assumptions with the case study that the project is in a state where the requirements elicitation has occurred and the process diagram has been established. We have already used a simple case study as a running example. The case study introduced in this section enable us to illustrate the versatility of the methods by way of using various change focus, various change types and how the outcome of the change classification differs with the need for input/output modifications.

5.1 The Case Study

Figure 6 represents a partial system design diagram of a course management system adopted from [45]. The diagram illustrates the relationships and some dependencies the activities have with each other. The relationships denoted in the diagram can be defined as follows:

- Requires (Req): An activity A_1 requires an activity A_2 if A_1 is fulfilled only when A_2 is fulfilled. A_2 can be treated as a pre-condition for A_1 [45].
- Refines (Ref): An activity A_1 refines an activity A_2 if A_2 is derived from A_1 by adding more details to it [45].
- Contains (Con): An activity A_1 contains information from $A_2 ... A_n$ if A_1 is the conjunction of the contained information from $A_2 ... A_n$ [45].

The identification of these relationships is beneficial in determining the impact of change when applying our methods to the case study. The detailed purpose of each activity is described as follows:

A1. The system allows end-users to provide profile and context information for registration.
A2. The system provides functionality to search for other people registered in the system.
A3. The system provides functionality to allow end-users to log into the system with their password.

A4. The system supports three types of end-users (administrator, lecturer and student).
A5. The system allows lecturers to set an alert on an event.
A6. The system maintains a list of events about which the students can be notified.
A7. The system notifies the students about the occurrence of an event as soon as the event occurs.
A8. The system actively monitors all events.
A9. The system notifies students about the events in the lectures in which they are enrolled.
A10. The system allows students to enroll in lecturers.
A11. The system allows lecturers to send e-mail to students enrolled in the lecture given by that lecturer.
A12. The system allows students to be assigned to teams for each lecture.
A13. The system allows lecturers to send e-mail to students in the same group.
A14. The system allows lecturers to modify the content of the lectures.
A15. The system gives different access rights to different types of end-users.
A16. The system supports two types of end-users (lecturer and student) and it will provide functionality to allow end-users to log into the system with their password.

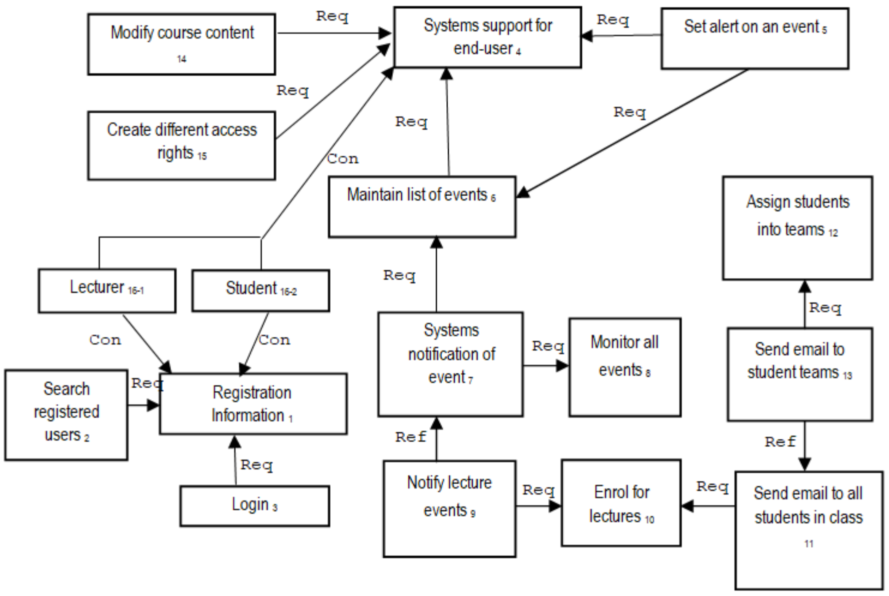
Modify course content 14
Req
Systems support for end-user 4
Req
Set alert on an event 5
Req
Create different access rights 15
Req
Req
Con
Maintain list of events 6
Assign students into teams 12
Req
Lecturer 16-1
Student 16-2
Con
Con
Systems notification of event 7
Req
Monitor all events 8
Req
Send email to student teams 13
Search registered users 2
Req
Registration Information 1
Ref
Ref
Req
Login 3
Notify lecture events 9
Req
Enrol for lectures 10
Req
Send email to all students in class 11

Fig. 6. Partial system design diagram of a course management system.

5.2 Applying Them to the Case Study

The example consists of two scenarios, where we apply the specification and classification methods. These scenarios are based on our observations as university academics who use similar course management systems. The following hypothetical new requirements are identified:

1. In an emergency, it would be more effective to send an SMS notification to students as well as an email.
2. Marking attendance manually tends to be rather ineffective, especially when a census needs to be carried out. It would be better to mark attendance electronically.

The application of the implementation template yields the following results.

Table 10. Change 01 result

Change 01	**Possibility 01**	**Possibility 02**
Object	Enrol for lectures A_{10}	Send email to all students A_{11}
Purpose	Functionality enhancement	Functionality enhancement
Focus	Add	Modify
Additional Question	Need additional Input / Output? Y	Input/output modification? Y
Result		
Change Type	Add new activity	Inner property + Output interface modification
Action	Add new activity by using information from A_{10}	Modify A_{11} internally and the output interface

Table 11. Change 02 result

Change 02	**Possibility 01**
Object	Enrol for lectures A_{10}
Purpose	Identification of new requirement
Focus	Add
Additional Question	Need additional Input / Output? Y
Result	
Change Type	Add new activity
Action	Add new activity by using information from A_{10}

5.3 Discussion of the Results

Tables 10 and 11 demonstrate how the specification and classification methods can be applied to this case study. The template given in Table 8 has been used for obtaining the result for each change.

Multiple possibilities can be created for each change event, depending on the event, availability of existing activities and various combinations that could be incorporated to realize the change. Such an instance has been provided for the 1st change event. In this change, the need to send SMS to students can be accomplished by either creating a new activity or modifying an existing activity (A_{11}). As such, when creating a new activity, it requires information from A_{10}. Therefore, the activity directly affected by the event is A_{10}. Rest of the table for the case study follows the process as explained through the simple stock control example.

In the second change event, we considered only one possibility. The requirement is to allow lecturers to mark attendance electronically. There doesn't seem to be any existing activity that can be modified to serve this purpose, therefore the only option is to create a new activity. As such a new activity is created that requires student information, which is provided by A_{10}. Therefore, the activity directly affected by the event is A_{10} and the rest of the table also follows the same principle as explained through the simple stock control example.

This example demonstrates how the specification and classification methods can be used to generate multiple possibilities for a single change. This outcome provides decision makers with the option of choosing the most appropriate way of implementing the change. The example above illustrates the way these methods can help both business and IT personnel involved, analyse business changes and thereby assist in the change management process. At the business level, the business analyst can use Tables 1 and 2 to define and describe the requirements change without any ambiguities. As a result of this IT personnel are able to not only understand the change but also understand the need for change and identify the location of change.

6. Comparison with Related Work

We shall describe what the literature has said about the related work and concepts like taxonomies and classification which are important concepts in studying change identification and classification.

6.1 Taxonomies

1) Research analysing change uses a plethora of techniques in order to build a taxonomy that can be used to identify changes as well as their impact. One such mechanism is the use of requirement engineering artifacts, such as use cases. The research done by Basirati et al. [46] establishes a taxonomy of common changes based on their observation of changing use cases that can then be used in other projects to predict and understand RCs. They also contribute to this research space by identifying which parts of use cases are prone to change as well as what changes would create difficulty in application, contributing also to the impact analysis of change.
2) The taxonomy developed by Buckley et al. [47], proposes a software change taxonomy based on characterizing the mechanisms of change and the factors that influence software change. This research emphasizes the underlying mechanism of change by focusing on the technical aspects (i.e. how, when, what and where) rather than the purpose of change (i.e. the why) or the stakeholders of change (i.e.

who) as other taxonomies have done. This taxonomy provides assistance in selecting tools for change management that assist in identifying the changes correctly.

3) McGee and Greer [3] developed a taxonomy based on the source of Requirements Change (RC) and their classification according to the change source domain. The taxonomy allows change practitioners to make distinctions between factors that contribute to requirements uncertainty, leading to the better visibility of change identification. This taxonomy also facilitates better recording of change data which can be used in future projects or the maintenance phase of the existing project to anticipate the future volatility of requirements.
4) Gosh et al. [48] emphasize the importance of having the ability to proactively identify potentially volatile requirements and being able to estimate their impact at an early stage is useful in minimizing the risks and cost overruns. To this effect, they developed a taxonomy that is based on four RC attributes i.e. phases (design, development and testing), actions (add, modify and delete), sources (emergent, consequential, adaptive and organizational) and categories of requirements (functional, non-functional, user interface and deliverable).
5) The taxonomy established by Briand et al. [41] is the initial step in a full-scale change management process of UML models. In their research, they establish that change identification is the first step in the better management of RCs. The classification of the change taxonomy is based on the types of changes that occur in UML models. They then use this taxonomy to identify changes between two different versions of UML models and finally to determine the impact of such changes.

6.2 Classifications

There are many benefits of using a classification, the main benefits being to manage change to enable change implementers to identify and understand the requirements of change without ambiguity [49, 50]. The classification of RC has been studied in various directions. Table 12 lists the different directions which have been the subjects of studies.

6.3 Other Change Identification Methods

1) Kobayashi and Maekawa [4] proposed a model that defines the change requirements using the aspects where, who, why and what. This allows the system analyst to identify the change in more detail, resulting in better impact identification as well as risk and effort estimation. This method consists of verification and validation and can be used to observe the RCs throughout the whole lifecycle of the system.
2) The change identification method usually has a pre-established base upon which its semantics are built. Ecklund's [42] approach to change management is a good example of this. The approach utilizes use cases (change cases) to specify and predict future changes to a system. The methodology attempts to identify and

incorporate the anticipated future changes into a system design in order to ensure the consistency of the design.

Table 12. Direction is change classification

Direction	Parameters	Comment
Type [40, 48, 50-54]	Add, Delete, Modify	The most common way of classifying change.
Origin [11, 48, 55]	Mutable, Emergent, Consequential, Adaptive, Migration	Derived from the places where the changes originated from.
Reason [13, 50, 51]	Defect fixing, Missing requirements, Functionality enhancement, Product strategy, Design improvement, Scope reduction, Redundant functionality, Obsolete functionality, Erroneous requirements, Resolving conflicts, Clarifying requirements, Improve, Maintain, Cease, Extend, Introduce	Helps determine the causes of change and understand change process and related activities.
Drivers [56]	Environmental change, RC, Viewpoint change, Design change	Helps change estimation and reuse of requirements.

6.4 Identifying limitations and comparison

We use the work listed in Table 13 (discussed above) to describe the limitations of the existing work and compare our methods to define what has been achieved.
An examination of the work reported above lead to the identification of four key limitations;

1. There is little agreement and commonality between the studies;
2. for the process of specification and classification of change to be used successfully, in our view it needs to be a part of the same process (change request); they complement each other by providing a better understanding of the requirements change;
3. there has been little emphasis on designing specification methods related to change; and
4. a common limitation of the above classifications is the lack of guidance in applying them to change management activities.

As a result, we believe that a void exists in the practical application of change specification and classification and our methods address this research gap.

Table 13. Comparison with the related work

Technique	**Limitations**	**What our methods can address**
Basirati et al. [46] and Ecklund [42]	Can only be applied if use cases are available or used in the development process.	They are applied at a design phase, which enables the identification of changes at an early stage. Can be used as long as there is a form of design diagram of the system.
Buckley et al. [47]	It did not directly address issues arising from miscommunication of change.	They can be directly used for managing changes for the purpose of identifying changes.
McGee and Greer [3] and Ecklund [42]	They are limited to providing assistance in predicting change.	They provide a way of communicating change as well identifying them in an early stage as to where and how the change should be applied.
Gosh et al. [48]	Only used for identification of change.	Provide preliminary guidance on how to manage the changes.
Briand et al. [41]	Can be used only if UML models are available.	Can be used as long as there is a form of design diagram for the system.
Kobayashi and Maekawa [4]	This is a complex method for verifying changes.	They address change management issues arising from miscommunication.

7. Conclusions and Future Work

The purpose of the change specification and classification methods presented in this paper is to manage requirements change by improving change communication and elicitation. Under normal circumstances, business changes flow from the business side to the IT side. Therefore, the impact of this study belongs to both these categories i.e. business and IT. First, considering the business side, we ensure a requirements change has been clearly communicated to the IT side. As mentioned earlier, there is often difficulty in promoting effective dialogue about the nature of the change between these two parties. Therefore, a change specification method would be essential for business analysts in communicating change.

Second, on the IT side, it is critical that change enablers have a mutual understanding of not only the precise nature of the change but also the reason for its existence, i.e. its purpose. This insight translates into a better realization of the requirements change. Equally important is a quick response from IT in redesigning the system to suit the requirements change. The three main categories: *object*, *purpose* and *focus* of the change specification method enhance understanding while the classification of the change type and the resulting action assists system designers to incorporate the change into the system design much faster.

Given the above impact of our methods, we believe that there are substantial benefits of specification and classification methods that will lead to improvements in the change management process. In our view, the benefits of these methods are:

- Promotes a mutual understanding of requirements change between business and IT through the templates provided by Tables 2 and 3.
- Supports the decision-making process by helping to determine the need for the change.
- Assists in determining the best course of action in implementing the requirements change through Table 7.

In future work, we plan to use the multiple change identification possibilities to evaluate the best course of action to enable system designers to respond quickly to change requests. Furthermore, we suggest it will be useful in evaluating the interdependencies of these change requests as they relate to interdependencies of the system requirements and its implementation. Identification of interdependencies between changes can lead to identification of conflicts between requirement changes. Also, it would be valuable if it were possible identify the difficulty level and priority of the changes so that resources such as time and effort can be allocated more effectively. Identifying the difficult level of the change would further result in assisting the decision of the plausibility of implementing the change.

References

1. Nurmuliani, N., Zowghi, D., Williams, S. P.: Requirements volatility and its impact on change effort: Evidence-based research in software development projects. In Proceedings of the 11th Australian Workshop on Requirements Engineering. (2006)
2. Williams, B. J., Carver, J., Vaughn, R .B.: Change Risk Assessment: Understanding Risks Involved in Changing Software Requirements. Software Engineering Research and Practice, pp. 966-971. (2006)
3. McGee, S., Greer, D.: A software requirements change source taxonomy. In Proceedings of the 4th International Conference on Software Engineering Advances, 51-58. (2009)
4. Kobayashi, A., Maekawa, M.: Need-based requirements change management. In Proceedings of the 8th nternational Conference and Workshop on Engineering of Computer Based Systems, 171-178. (2001)
5. Rajabi, B. A., Lee, S. P.: Change management in business process modeling survey. In Proceedings of the International Conference on Information Management and Engineering, 37-41. (2009)
6. André, C., Mallet, F.: Specification and verification of time requirements with CCSL and esterel. ACM Sigplan Notices, 2009, Vol. 44, No. 7, 167-176. (2009)
7. Dillingham, G.: Air traffic control: evolution and status of FAA's Automation Program. Washington, DC: United States General Accounting Office. (1998)
8. Lock, S., Kotonya, G.: An integrated framework for requirement change impact analysis. In Proceedings of the 4th Australian Conference on Requirements Engineering. (1999).
9. Sommerville, I., Sawyer, P.: Requirements engineering: a good practice guide. John Wiley & Sons, Inc. (1997)
10. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In Proceedings of the Conference on the Future of Software Engineering, 35-46. (2000)

11. Harker, S. D., Eason, K. D., Dobson, J.E.: The change and evolution of requirements as a challenge to the practice of software engineering. In Proceedings of the IEEE International Symposium on Requirements Engineering, 266-272. (1993).
12. Bohner, S.: Impact analysis in the software change process: A year 2000 perspective. In Proceedings of the International Conference on Software Maintenance, 42-51. (1996)
13. Nurcan, S., Barrios, J., Grosz, G., Rolland, C.: Change process modelling using the EKD-Change Management Method. In Proceedings of the European Conference on Information Systems, 513-529. (1999)
14. Chitchyan, R., Rashid, A., Rayson, P., Waters, R.: Semantics-based composition for aspect-oriented requirements engineering. In Proceedings of the 6th international conference on Aspect-oriented software development, 36-48. (2007)
15. Pohl, K.: Requirements engineering: fundamentals, principles, and techniques. Springer Publishing Company, Incorporated. (2010)
16. Guttag, J. V., Horning, J. J., Larch: languages and tools for formal specification. Springer Science & Business Media. (2012)
17. Jureta, I. J., Borgida, A., Ernst, N. A., Mylopoulos, J.: Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In Proceedings of the 18th IEEE International Requirements Engineering Conference, 115-124. (2010)
18. Jayatilleke, S., R. Lai, R.: A Method of Specifying and Classifying Requirements Change. In Proceedings of the 22nd Australian Software Engineering Conference (ASWEC), 175-180. (2013)
19. Van Solingen, R., Basili, V., Caldiera, G., Rombach, H. D.: Goal question metric (gqm) approach. Encyclopedia of Software Engineering. (2002)
20. Weiss, M.: Resource description framework. Encyclopedia of Database Systems, 2423-2425. (2009)
21. Koziolek, H.: Goal, question, metric. Dependability metrics: Springer, 39-42. (2008)
22. Berander, P., Jönsson, P.: A goal question metric based approach for efficient measurement framework definition. In Proceedings of the ACM/IEEE international symposium on Empirical software engineering, 316-325. (2006)
23. Subramanian, D. V., Geetha, A.: Adaptation of goal question metric technique for evaluation of knowledge management systems. Review of Knowledge Management, Vol. 1, No. 2, 4. (2011).
24. Basili, V., J. Heidrich, J., Lindvall, M., Munch, J., Regardie, M., and Trendowicz, A.: GQM + Strategies - Aligning Business Strategies with Software Measurement. In Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement, 488-490. (2007).
25. Happel, H. J., Seedorf, S.: Applications of ontologies in software engineering. In Proceedings of the Workshop on Sematic Web Enabled Software Engineering, 5-9. (2006)
26. Rolland, C., Salinesi, C., Etien, A.: Eliciting gaps in requirements change. Requirements Engineering, Vol. 9, No. 1, 1-15. (2004)
27. Tse, T., Pong, L.: An examination of requirements specification languages. The Computer Journal, Vol. 34, No. 2, 143-152. (1991)
28. Checkland, P. B.: Information systems and systems thinking: Time to unite?. International Journal of Information Management, Vol. 8, No. 4, 239-248. (1988)
29. Mumford, E.: Redesigning human systems. IGI Global. (2003)
30. Davis, A. M.: The design of a family of application-oriented requirements languages. Computer, Vol. 5, No. 15, 21-28. (1982)
31. Roche, C.: Ontoterminology: How to unify terminology and ontology into a single paradigm. In Proceedings of the 8th International Conference on Language Resources and Evaluation, 2626-2630. (2012)
32. Gillam, L., Tariq, M., Ahmad, K.: Terminology and the construction of ontology. Terminology, Vol. 11, No. 1, 55-81. (2005)

33. Castañeda, V., Ballejos, L., Caliusco, M. L., Galli, M. R.: The use of ontologies in requirements engineering. Global journal of researches in engineering, Vol. 10, No. 6. (2010)
34. Gruber, T. R.: A translation approach to portable ontology specifications. Knowledge acquisition, Vol. 5, No. 2, 199-220. (1993)
35. Nurmuliani, N., Zowghi, D., Powell, S.: Analysis of requirements volatility during software development life cycle. In Proceedings of the Software Engineering Conference, 28-37. (2004)
36. Lam, W., Loomes, M.: Requirements evolution in the midst of environmental change: a managed approach. In Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering, 121-127. (1998)
37. Lam, W., Shankararaman, V.: Managing change in software development using a process improvement approach. In Proceedings of the 24th Euromicro Conference, Vol. 2, 779-786. (1998)
38. Nurmuliani, N., Zowghi, D., Williams, S. P.: Using card sorting technique to classify requirements change. In Proceedings of the 12th Conference on Requirements Engineering Conference, 240-248. (2004)
39. Xiao, H., J. Quo, J., Zou, Y.: Supporting change impact analysis for service oriented business applications. In Proceedings of the Systems Development in SOA Environments, 6-6. (2007)
40. Gupta, G., Singh, Y., Chauhan, D. S.: A Dynamic Approach to Estimate Change Impact using Type of Change Propagation. JIPS, Vol. 6, No. 4, 597-608. (2010).
41. Briand, L. C., Labiche, Y., Sullivan, L.: Impact analysis and change management of UML models. In Proceedings of the International Conference on Software Maintenance, 256-265. (2003)
42. Ecklund Jr, E. F., Delcambre, L.M., Freiling, M. J.: Change cases: use cases that identify future requirements. ACM SIGPLAN Notices, Vol. 31, No. 10, 342-358. (1996)
43. Yin, R. K.: Discovering the future of the case study method in evaluation research. Evaluation practice, Vol. 15, No. 3, 283-290. (1994)
44. Yin, R.K.: Case study methods. (2012)
45. Göknil, A., Kurtev, I., van den Berg, K.: Change impact analysis based on formalization of trace relations for requirements. In Proceedings of the Traceability Workshop (ECMDA-TW). (2008)
46. Basirati, M. R., Femmer, H., Eder, S., Fritzsche, M., Widera, A.: Understanding Changes in Use Cases: A Case Study. In Proceedings of the International Symposium on the Requirements Engineering. (2015)
47. Buckley, J., Mens, T., Zenger, M., Rashid, A., Kniesel, G.: Towards a taxonomy of software change. Journal of Software Maintenance and Evolution: Research and Practice, Vol. 17, No. 5, 309-332. (2005)
48. Ghosh, S., Ramaswamy, S., Jetley, R. P.: Towards requirements change decision support. In Proceedings of the 20th Asia-Pacific Software Engineering Conference (APSEC), Vol. 1, 148-155. (2013)
49. Harker, S. D. P., Eason, K. D., Dobson, J. E.: The change and evolution of requirements as a challenge to the practice of software engineering. In Proceedings of the IEEE International Symposium on Requirements Engineering. (1993)
50. Nurmuliani, N., Zowghi, D., Fowell, S.: Analysis of Requirements Volatility during Software Development Life Cycle. In Proceedings of the Australian Software Engineering Conference, 28. (2004)
51. Nurmuliani, N., Zowghi, D., and Williams, S. P.: Using card sorting technique to classify requirements change. In Proceedings of the Requirements Engineering Conference, 240-248. (2004)

52. Hua, X., Jin, Q., Ying, Z.: Supporting Change Impact Analysis for Service Oriented Business Applications. In Proceedings of the Workshop on Systems Development in SOA Environments, 6-6. (2007)
53. Gupta, C., Singh, Y., Chauhan, D.: A dynamic approach to estimate change impact using type of change propagation. Journal of Information Processing, Vol. 6, No. 4. (2010)
54. Stark, G. E., Oman, P., Skillicorn, A., Ameele, A.: An examination of the effects of requirements changes on software maintenance releases. Journal of Software Maintenance, Vol. 11, No. 5, 293-309. (1999)
55. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland. (2000)
56. Lam, W., Loomes, M.: Requirements evolution in the midst of environmental change: a managed approach.Iin Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering, 121-127. (1998)

Shalinka Jayatilleke holds a BSc (Hons) from Institute of Technological Studies (Affiliated to Troy University, USA), an MSc from Sri Lanka Institute of Information Technology and present reading for a PhD (in computer science) at La Trobe University, Australia. She is currently an Associate lecturer at La Trobe University with an academic career of 12 years. His current research interests are requirements engineering and change management.

Richard Lai holds a BE (Hons) and a MEngSc from the University of New South Wales and a PhD from La Trobe University, Australia. He has spent about 10 years in the computer industry prior to joining La Trobe University in1989. His current research interests include component-based software system, software measurement, requirements engineering, and global software development.

Karl Reed holds an Assc. Dip. in Communications Engineering from the RMIT and an MSc in Computer Architecture from Monash University. After 12 years in industry he joined Monash University in 1976. He moved to La Trobe University in 1988. His research interests include software testing, compilable restricted natural languages, web-page design, high-level software re-use, design isomorphism, organisational resilience, cloud computing and industry policy.

Received: November 30, 2016; Accepted: July 2, 2017.