Using Card Sorting Technique to Classify Requirements Change

Nur Nurmuliani, Didar Zowghi,
Faculty of Information Technology
University of Technology, Sydney
{nur, didar}@it.uts.edu.au

Susan P. Williams
School of Business
Faculty of Economics & Business
The University of Sydney
s.williams@econ.usyd.edu.au

Abstract

Requirements Volatility is considered to be a major source of risk to the management of large and complex software projects. The ability to characterise the nature and origins of requirements change during software development is important and can lead organisations towards more effective management of changing requirements.

This paper focuses on a study to establish how practitioners classify requirements change requests. We used the Card Sorting method to identify categories of change requests that software developers use in practice. Card sorting is a knowledge elicitation method that is commonly used for capturing information about different ways of representing domain knowledge. This study has allowed us to get valuable insights into the way practitioners classify change requests and to understand their perspectives on classification. This classification is a valuable source of information in prioritizing change requests and assessing their impact. Our findings from the card sorting exercise further reveal that the criteria used for categorization are related to the role the practitioner plays in the software development team and the nature and extent of their responsibilities.

Keywords: requirements change, classification, card sorting.

1. Introduction

It is clearly established in the extant literature that requirements change while systems/software is being developed. These changes are often brought about by aggressive market competition and rapidly changing technology. Changing requirements have been recognised as a difficult problem for most large and complex software projects [1, 2, 3]. Software engineering researchers in the past have focused on identifying more effective strategies and methods for handling changing requirements.

Managing requirements change is one of the key process areas that organisations with low maturity level have to focus on when engaged in improving their software processes. Identifying and characterising the nature of requirements changes could lead to more effective management of changing requirements [3, 4].

While previous research has focused on managing and classifying requirements change, investigations of how software developers classify such changes are lacking. The work described in this paper aims to investigate the way practitioners classify requirements change requests.

In this paper we present the card-sorting technique as an effective method for classifying requirements change. The objective of this method is to gain insights into the ways that practitioners categorise change request data, and to identify the different perspectives on classifying requirements changes. This method also allows us to gain more information about how the practitioners' roles in software development team may contribute to their categorisations.

The main contributions of this study are: (a) the method (based on card sorting technique) that we used to construct the classification of requirements changes, (b) the elicitation of the categories that software developers use in practice when classifying change requests. The method is described in sufficient details to make it repeatable by researchers and practitioners alike. The results of this exercise further informed us of practitioners' thoughts about the value of the classification itself.

The paper is organised as follows. Section 2 describes previous studies related to the classification of requirements change. Section 3 discusses the card

Proceedings of the 12th IEEE International Requirements Engineering Conference (RE'04)
1090-705X/04 $ 20.00 IEEE

sorting method used in our case study. Section 4 presents the results of card sorting exercise. Section 5 discusses some issues associated with the classification of changes and the use of the card sorting method. Finally we end this paper with conclusions and recommendations for future work.

2. Requirements Change Classification

Classifying requirements changes has been identified as one of the ways to improve practitioners' understanding of the nature of these changes [3, 4]. Harker and Eason [3] distinguish between stable and volatile requirements. They classify volatile requirements into five categories such as *emergent*, *consequential*, *mutable*, *adaptive*, and *migration requirements*. These volatile types of requirements are mostly driven by changes in user needs, environmental factors, technology, developers' knowledge, and policies. Categorizing changes will help software developers analyse each type of change according to its origin and assess its impact on software development product and process.

Lam and Shankharaman [4] adopted a process improvement approach in developing a framework to manage changing requirements during software development. They define 'classifying change' as one of the five best practices in managing change. The classification they propose is domain-specific change (i.e. *Screen change*, *Report change*, and *Data change*). Other classifications of requirements changes have been defined in the literature both during software development and maintenance [5, 6].

The classifications described in most of the previous studies are in the form of high-level abstraction of changes, except for the classification by Lam and Shankharaman [4], which is more specific and domain oriented. The classifications defined by other researchers [3, 5, 6] are very useful when information about the nature of changes (such as reasons for change and its origin) are included in the change request forms. However, problems arise if this kind of information is not available or only partly recorded in the change request forms [7].

Managing requirements change is a multifaceted problem. The solution may be to consider adopting a more multidimensional approach. For example, Harker and Eason [3] suggest that we should consider the nature of changes, the characteristics of the participants, and design context as these are all important factors in the strategy for managing requirements change.

While the classification of requirements change has been identified in many studies, these efforts have not investigated how software engineering practitioners would classify changes. The work described in this paper intends to establish the ways practitioners classify requirements changes. Gaining insights into the ways practitioners classify requirements change will help us to understand the multifaceted nature of the problem.

3. Research Approach

In our previous work [7, 8, 9,10], we have analysed the various aspects of requirements volatility during software development. In a recent study [7], we have conducted an industrial case study on requirements volatility. Using historical information from a change request database we investigated several important dimensions of requirements change, such as the rate of change requests, the rationale/reason of the proposed requirement changes, the types of requirements change (i.e. addition, deletion, and modification), and the source/origin of requested change. Based on this information we developed a preliminary classification of requirements change [7].

During this preliminary analysis we discovered that most of the change requests that were used by the organisation in this case study had little information about the reason for change and had inadequate information to analyse the importance of the change to be made. To overcome the limitation of insufficient change request data and in order to triangulate our findings, we used the Card Sorting method to capture and elicitate practitioners' perspectives. Furthermore, the practitioners' perspectives are used as a method of validating our preliminary classification.

3.1. The Card Sorting Method

Card Sorting is an established method for knowledge elicitation [11] and has been widely used in various fields such as Psychology, Knowledge Engineering, Software Engineering, and Web Site Design. In the field of Requirements Engineering, card sorting is described as the most effective method for eliciting requirements engineering problem domains [12].

Card sorts typically consist of the researcher creating approximately 60 index cards (3"x5"), on which a description of domain entities is printed. Respondents sort the cards into groups/categories and explain the criteria they use for sorting, and the names they assign to groups [13, 14, 15].

It has been demonstrated that card sorts have many positive aspects that make them a useful elicitation tool. First, card sorts can be used to investigate respondent's recall knowledge of the domain entity [16]. Second, card sorts are a useful technique to

distinguish between high and low level problems [11]. Third, card sorts offer more insights into the target population's views of the topic [17]. Fourth, card sort results can provide an input for another technique and further analysis [15]. The results can also be used as input to a hierarchy or further classification (e.g. information architectures for web sites) [17]. Finally, the card sorting process can be done relatively quickly, at nominal cost, and is flexible and easy for the researcher to handle [17, 18]

In general, most researchers have suggested that the Card-sorting method is an excellent approach to help develop classifications and that it delivers the classification that people actually use.

The following section describes our card sorting approach in an industrial setting.

3.2. Setting and Participants

The Card Sorting exercise was conducted at the Global Development System (hereafter GDS[1]) organisation located in Sydney, Australia. GDS is an engineering lab that develops product line software. The software produced is characterized by the delivery of a series of releases. Each release is around 8000KLOC, development time between 12-18 months, with approximately 180 full time developers involved. The product is an enterprise software application, of which customers are themselves developers using the system for developing software. Requirements for new releases are requests for enhancements to the product and they are gathered from multiple sources:

- Market needs (representing current customers needs and market directions representing potential for future customers)
- Product strategy requirements (representing technology and engineering direction of the product in line with the organizational strategy)

At GDS, key stakeholder groups are scattered across several continents. The product strategy is directed from the US, where the Product and Program Management group is located across four sites. The development group is located in three Australian and one New Zealand sites, and customers are grouped in five large market segments across five continents. In addressing the geographical distribution of customers worldwide, the organization maintains on-site field support centres, to provide services to the diverse market segments.

[1] The company and product names are fictitious to preserve confidentiality

For the card sorting exercise, we obtained GDS management permission to involve their software engineers and management in our card sorting exercise. Management then selected 12 out of 20 senior software engineers in the organisation. The management sent a notification letter to all selected participants requesting their contribution to the study. We contacted all the engineers to introduce our study goals, and sought their commitment to the project. Only two engineers could not participate. The 10 participants involved in this study represented a number of the organisation functional areas such as senior management, project management, engineering management, systems architect, and technical leads. These participants agreed to take part in the Card Sorting exercises, which were scheduled during a one-week period and the time slot for each participant was arranged according to the participant's availability.

3.3. Card Sorting Materials

Before the actual card sorting commenced, lists of candidate entities were prepared (i.e. requirements change problems and issues). These were extracted from the company's change request database. This preparation stage was the most challenging and time consuming. A brief description of each change request was extracted from change request forms.

As a result, we produced a set of 52 cards, each with a brief descriptions of requirements changes. The descriptions include the type of changes, the rationale or reasons for a requested change, and change activities involved. The description of each item was printed on a 3" x 5" card. All the cards are the same size and are numbered with a unique number (random numbers) for recording the results after each session.

3.4. Card Sorting Procedures

The researcher coordinating the card sort activity conducted a one-on-one session with each participant. Since the practitioners have very limited time available within their daily tight schedule, this exercise involved only 'single-criterion sorts', (i.e. sorting the same set of cards, using a single criterion).

The card sorting proceeded in the following five steps:

1) At the beginning of the exercise, a brief explanation of the sorting exercise, and verbal instructions were given to the participant. The main purpose of the sorting was to classify the change requests related to changing requirements.
2) The participant was given the cards. Before the sort began, the participant was given time to read

through all the cards to familiarise himself with the contents of cards,

3) The participant was instructed to sort the cards into groups of similarity according to his own criteria. The cards were placed on the table and arranged into groups or piles. The participant was free to form as many groups of cards as they felt necessary,
4) After the sorting was completed, the participants' chosen criteria and categories were recorded. The unique numbers of each card are used to record which cards were placed in which categories. The participants were also given a pad of blank Post-It stickers to write a label or the name of each of the categories. An example of participant's set of cards is illustrated in Figure 1.
5) At the end of the sorting session, a simple questionnaire (see Appendix) was administered to obtain further information about the participants and their views on classification and the Card sorting method. The interview was recorded on audiotape and transcribed.

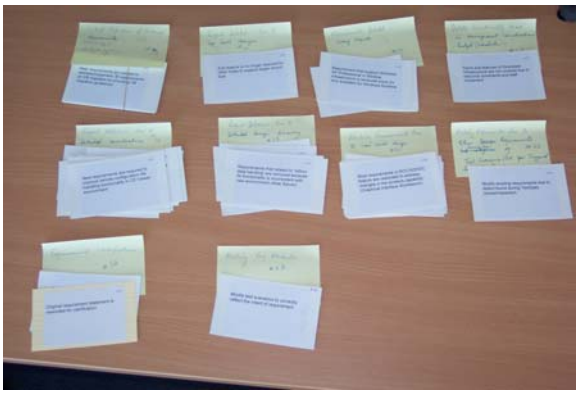

Figure 1. Participant's set of cards after sorting

3.5. Analysis

Since the purpose of this study was to establish the way software practitioners categorise changes (change requests) and to gather more information about the kind of classification they produce, the analysis of the results is primarily qualitative. Since we used a small sample of participants from one company, statistical analysis is not appropriate.

The data from this study was analysed in terms of: lists of the participants' criteria, number of categories, lists of named categories, and the commonality (agreement) between the participants.

Content analysis was used to assess the commonality between respondents classification. There are two forms of agreement, *verbatim* and *gist*. Verbatim agreement takes place when different participants use exactly the same words. Gist agreement takes place when different participants use different words for the same ultimate meaning [16]. For gist agreement, *independent judges* were asked to identify which criteria and categories are forms of gist agreement. The respondents' categories are grouped into high level constructs. The number of respondents who used the construct is identified. Furthermore, the transcripts of the interviews with the participants were analysed to triangulate and verify the findings.

4. Results

All the participants completed the entire process. Each session took on average 40 minutes, though some took 60 minutes. The average number of categories given by each participant was 6.6 with the minimum number of categories 4 and the maximum number was 10.

4.1. Criteria and Categories Used

The criteria used by the practitioners in the card sorting sessions include: *reasons for changing requirements, general changes, schedule impacted, and the magnitude of effort involved.* As illustrated in Figure 2 the most common criterion used by the participants (60%) was '*Reason for changing requirements'*. This is not surprising since the entities are related to the issues of changing requirements.

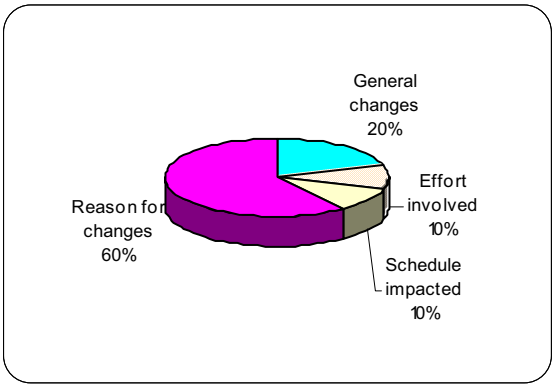

Figure 2. Card Sorting Criteria

A total of 66 categories were generated by the 10 participants. The distribution of categories for all criteria is presented in Figure 3.

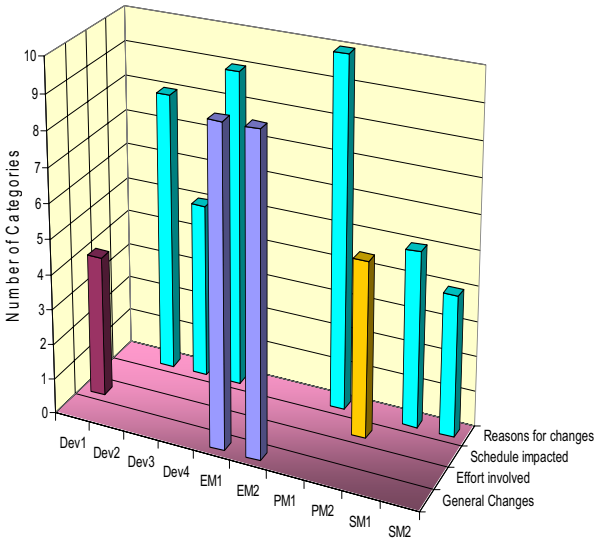

Figure 3. The distribution of categories by sorting criteria for all participants

(Note: SM=senior manager, PM=project manager, EM=Engineering manager, and Dev=Developer)

After *verbatim* and *gist* agreement has been analysed, the similarities between respondents' criteria and categories can be identified. There are two examples of verbatim agreement for criteria in this finding: '*General changes*' and '*Reason for changing requirements*'.

The following is the results of our analysis on the participants' categories within the criteria involved in the card sorts.

First, we have two participants who used two different criteria in the card sorts, e.g. '*the magnitude of effort involved'* and '*schedule impacted'*. These participants produced completely different categories. The first participant grouped the cards into four categories based on 'the *magnitude of effort involved'*. The four categories of requirements change requests include High Effort (for functionality changes in known complex area), Medium Effort (for requirements changes that have reasonable impact on effort), Low Effort (for minor changes to requirements or functionality), and No Effort (for rework/rewording requirements). This participant whose role is Developer (System Architect) was mostly concerned about the effort required to implement the change.

The second participant whose role is a Project Manager had categorised the cards into five categories based on *'schedule impacted'* criterion. His categories are:

- New requirements with high impact on schedule,
- New requirements with low impact on schedule,
- Modified requirements with significant impact on schedule,
- Modified requirements with no impact on schedule,
- Removal requirements that have no impact on schedule (i.e. less work).

This participant was mostly concerned about the impact of adding new requirements or modifying existing requirements on project schedule.

Second, the first verbatim agreement for criteria in this finding is '*General changes*'. Two participants used this criterion when they sorted the cards. We found very little overlap among the categories identified by these participants. The overlapping categories were: addition of new requirements, drop/removal and modification of existing requirements for clarification purposes, and functionality upgrade. It is interesting to note that these two participants have the same role as Engineering Manager in this organisation.

Table 1. List of constructs based on common categories within participants

List of Superordinate Constructs	Total
Product Strategy, changes to media packaging/ licensing/ branding	5
Hardware/Software Environment changes	7
Scope reduction, due to technical reasons and lack of resources	5
Discovered Requirements as results of design improvement	7
Missing requirements	3
Clarification changes, related to rewording requirements text, redundant requirements, and resolving interdependencies	8
Testability, changes that triggered by test scenario changes	3
Functionality enhancement/upgrade	5

Finally, the second verbatim agreement for criteria is '*Reasons for changes'* that were used by the remaining six participants. The numbers of categories were drawn from the cards varied significantly (see

Figure 3). The verbatim and gist agreements for categories were identified and grouped together into a superordinate construct (a higher abstraction level). Then, each superordinate construct was given a name that represents the identified grouped categories. Table 1 lists the superordinate constructs based on common categories generated by all participants.

The common categories or constructs listed in Table 1 represent a substantial part of the participants' categories for criteria 'reasons *for changes'*. These categories will serve as additional information to refine our preliminary classification of requirements changes developed in the previous study [7]

From the card sorting findings we identified some categories that do not belong to any group or construct because they are completely different to each other. These categories are *bug fix*, *architectural incompleteness*, *changing requirements attributes*, *wrong requirements*. However, we believe these categories are as important as the others and are potential categories requiring further investigation.

4.2. Participant Feedback

At the end of each card sorting session, the participant was asked to give his opinion about the requirements change classification and card sorting technique.

The majority of participants responded very positively to the card sorting exercise. They viewed the card sorts as a good method to sort out change requests quickly. The following comments reflect their positive reaction:

"…It is good and forces you to make decision"

"…It not only forces you to make decision, it also gives you an ability to make decision"

"It is useful technique because you can separate or sort them (change requests) easily".

Another participant explained that the card sorting is useful "for me to start thinking about categories". He continued to say "It forces me to put cards into some sort of categories that you never thought before".

Regarding the classification of change requests that reflected changing requirements, the majority of participants thought that the change classification may provide substantial benefits in improving their current change process and in the way they analyse changes. The following lists some of the benefits of change classification according to the participants of this study:

1. Classifying change requests could be used as a means of controlling and managing changes.
2. It can help in assessing the impact of requirements changes in a reliable way.
3. It can promote a common understanding within the software development team of what the changes actually mean.
4. It can be used to identify risk associated with each change request or the group of changes.
5. It can help in determining the change acceptability (i.e. reject or approve changes), hence supporting crucial decision making throughout software development lifecycle.
6. The requirements change categories and the subsequent constructs could be used to develop a multi dimensional matrix of all change requests. The particularly useful dimensions for more effective decision making are schedule, effort and reasons for changes. The matrix could be populated with all the change requests after categorisation has been performed. For example, when project managers need to prioritise the change requests for implementation purposes, they can consult the matrix to identify the effort, schedule and the reasons for each change request for their assessment.

5. Discussion

The use of the Card Sorting exercise was found to be a very effective method for exploring the practitioner's view of requirements change problems. Although this Card Sorting exercise has never been conducted in GDS and the software development team had never really had to think about change classifications previously, it received a good response from the practitioners who were involved in the sorting exercise. They found the technique easy to use and felt that it encouraged them to start thinking about change classification.

The most important issue for effective Card Sorting is that the participants and the researcher should sufficiently understand the domain entities [12]. In this study, the participants were familiar with and understood the domain entities being elicited (i.e. requirements change requests). Furthermore, the researchers' understanding of the requirements change has improved considerably through the long-term case study at the GDS.

Software developers at GDS involved in the cards sorting sessions exhibited a broad range of knowledge about the issues/problems related to changing requirements. During our Card Sorting sessions, we noted that each participant viewed the change problems, which are expressed in the change request forms, differently. For instance, the project manager was mostly concerned about the types of change that have impact on project schedule. Other participants

such as a system architect and an engineering manager were mostly concerned about changes that may need more or less work and general changes (i.e. additions, deletions, and modifications to requirements)

It is understandable that the GDS practitioners used their knowledge or expertise according to their roles to sort requirements change. Understanding the different views and the interplay between functional roles and problem domains is very useful for categorising requirements engineering problem domains [14]. The common categories resulting from our Card Sorting exercises contribute significantly to the refinement of our preliminary classification of requirements changes. Since this study was conducted in the GDS environment, the classification will be specifically applied to the work in this organisation. However, the method we described here could easily be adopted by any organisation that has an established change control process and a repository for storing and tracking change requests. This is because our analysis was based on information acquired from a change request database.

It is interesting to note that although GDS has an established change control process, our study revealed that the information recorded in the change request form is not sufficient for effective decision making. For example, the reason for change is not explicitly stated and hence it is open to interpretation. The other piece of valuable information missing in the form was an estimate of effort required for implementing the change. This limits the impact analysis. As a result of this study, suggestions have been made to GDS to include the missing information slots in the change request forms.

The common categories also reflect the types of changes to requirements during software development. For instance, there was high commonality between participants in the grouped categories of "C*larification changes"*. This type of changes involved rewording requirements text, removing redundant requirements, and modifying requirements to resolve interdependencies. The category *"Discovered requirements"* is related to adding new requirements that are discovered during product development or as a result of design improvement. The high commonality was also found in the category *"Hardware/Software environment changes"*, which is a group of changes related to requirements modification caused by changes in 3rd party software and supported platforms.

When we conducted the interview after each card sorting session, we found that most participants had a positive reaction to the idea of requirements change classification. Our results reveal that the majority of the participants believed that the classification might help them in analysing requested change to requirements more effectively in terms of effort estimation and assessing the impact of changes on schedule. In addition, the classification will assist in prioritising change requests. This is because currently they tend to treat all change requests equally whereas in fact some change requests are more important than others in terms of impact, cost and effort required to implement them.

Although the classification of requirements change described in this study emerged from professionals of one organisation who develops a specific type of software, many categories are generic and indicative of categories in use within other organisations. We encourage organisations to identify and validate their own classification according to their need. In other words, the classification developed should be meaningful to the organisation.

Finally, the results of this card sort exercise have significantly benefited our long term study of requirements volatility and will be a valuable source of information for the practitioners at GDS to improve their change management process.

6. Conclusions and Future Work

In this paper we have presented the use of the card sorting technique to classify requirements changes (change requests) in an industrial setting. The main contributions of this study are twofold. Firstly, the analysis of card sorting results has led us to get insights into the ways practitioners categorise change requests and obtain more information about their classification. Secondly, the Card Sorting method for eliciting knowledge of practitioners in classifying requirements changes has been very useful. This method was described in detail and therefore could potentially be used by other researchers and software practitioners in their own environment to identify and classify requirements changes.

This study represents a preliminary investigation to identify practitioners' classifications of change requests. There are two limitations to this study. First, we have adopted a single-case study methodology. Future work will be undertaken to establish the effectiveness of the card sort to identify practitioners' classifications of requirements change in other companies. Secondly, our preliminary findings and feedback from respondents indicate that there are many potential benefits in developing a classification of requirements change. These benefits are yet to be realised. Subsequent stages of this work will investigate the extent of these benefits in practice.

This study is the second phase in a long-term investigation of the phenomenon of requirements volatility and is one of a number of longitudinal

investigations currently being undertaken. It helps to set the scene for future stages of this research project. The findings of this case study have provided valuable insight about the practitioners' classification of requirements changes. The next stage of the research involves refining the classification of requirements changes by incorporating the results from these card sorts. This will allow us to develop a set of strategies to manage the impacts of requirements volatility during software development life cycle and will enable us to identify and manage risks associated with requirements volatility.

Acknowledgements

We would like to thank all the participants from the GDS organisation and the management for their continued support of our research.

References

[1] B. Curtis, H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems", *Communications of the ACM*, vol. 31, pp. 1268-1287, 1988.

[2] The Standish Group, "CHAOS: A Recipe for Success", 1998. http://www.standishgroup.com/sample_research/chaos1998.pdf

[3] S. D. P. Harker and K. D. Eason, "The Change and Evolution of Requirements as a Challenge to the Practice of Software Engineering". *In the Proceedings of IEEE International Symposium on Requirements Engineering*, (RE'93), San Diego, California, 1993

[4] W. Lam and V. Shankararaman, "Managing Change in Software Development Using a Process Improvement Approach", i*n the Proceedings of IEEE Euromicro Conference*, 1998.

[5] I. Sommerville, *Software Engineering*, 5th ed: Addison Wesley, 1996.

[6] R. S. Pressman and D. Ince, *Software Engineering: A Practitioner's Approach*, Fifth Edition Ed: McGrawHill, 2000.

[7] N. Nurmuliani, D. Zowghi, and S. Fowell, "Analysis of Requirements Volatility during Software Development Lifecycle", i*n the Proceedings of Australian Software Engineering Conference*, Melbourne, 2004.

[8] D. Zowghi and N. Nurmuliani, "Investigating Requirements Volatility during Software Development: Research in Progress". *In the Proceedings of the Third Australian Conference on Requirements Engineering*, Geelong, 1998.

[9] D. Zowghi and N. Nurmuliani, "A Study of the Impact of Requirements Volatility on Software Project Performance", in the proceeding of the 9th Asia-Pacific Software Engineering Conference, Gold Coast, Australia, 2002.

[10] D. Zowghi, R. Offen, and N. Nurmuliani, "The Impact of Requirements Volatility on Software Development Lifecycle", in the Proceeding of the International Conference on Software, Theory and Practice (ICS2000), Beijing, China, 2000.

[11] A. R. Barrett and J. S. Edwards, "Knowledge Elicitation and Knowledge Representation in a Large Domain with Multiple Experts", *Expert Systems with Applications*, vol. 8, pp. 169-176, 1995.

[12] N. A. M. Maiden and G. Rugg, "ACRE: Selecting Methods for Requirements Acquisition", *Software Engineering Journal*, vol. 11, pp. 183-192, 1996.

[13] J. Palmer, T. Duffy, K. Gomoll, T. Gomoll, J. Richards-Palmquist, and J. A. Trumble, "The Design and Evaluation of Online Help for Unix EMACS: Capturing the User in Menu Design", *IEEE Transactions on Professional Communication*, vol. 31, pp. 44-51, 1988.

[14] N. A. M. Maiden and M. Hare, "Problem Domain Categories in Requirements Engineering", *International Journal Human-Computer Studies*, vol. 49, pp. 281-304, 1998.

[15] L. Upchurch, G. Rugg, and B. Kitchenham, "Using Card Sorts to Elicit Web Page Quality Attributes", in *IEEE Software*, 2001, pp. 84 - 89.

[16] G. Rugg and P. McGeorge, "The Sorting Techniques: ATutorial Paper on Card Sorts, Picture Sorts and Item sorts", *Expert Systems*, vol. 14, pp. 80 - 93, 1997.

[17] D. E. Zimmerman and C. Akerelrea, "A Group Card Sorting Methodology for Developing Informational Web Sites", i*n the Proceedings of International Professional Communication Conference (IPCC)*. 2002.

[18] G. Rugg, C. Corbridge, N. P. Major, A. M. Burton, and N. R. Shadbolt, "A comparison of Sorting Techniques in Knowledge Elicitation", *Knowledge Acquisition*, vol. 4, pp. 279-291, 1992.

Appendix

Questionnaire on Card Sorting and Classification

A. Your Role

1. General Information.
   Name:
   Position:
   Date:

1. How many years have you acted in this role?
2. How many years have you been working in the software development industry?

B. Classification of Changes

1. What benefits might a classification of requirements change request provide?
2. Do you think the classification of changes will be useful to you?
3. Do you have any suggestions in developing the classification of changes?

C. Card Sorting Method

1. What do you think of this Card sorting method in gathering information?
2. Do you have any other suggestions to improve this method?