# Towards an understanding of the causes and effects of software requirements change: two case studies

McGee, S., & Greer, D. (2012). Towards an understanding of the causes and effects of software requirements change: two case studies. Requirements Engineering, 17(2), 133-155. DOI: 10.1007/s00766-012-0149-0

**Published in:**
Requirements Engineering

**Document Version:**
Publisher's PDF, also known as Version of record

**Queen's University Belfast - Research Portal:**
Link to publication record in Queen's University Belfast Research Portal

Requirements Eng (2012) 17:133–155
DOI 10.1007/s00766-012-0149-0

RE'11 BEST PAPERS

# Towards an understanding of the causes and effects of software requirements change: two case studies

Sharon McGee · Des Greer

Received: 29 October 2011 / Accepted: 18 March 2012 / Published online: 9 May 2012

**Abstract** Changes to software requirements not only pose a risk to the successful delivery of software applications but also provide opportunity for improved usability and value. Increased understanding of the causes and consequences of change can support requirements management and also make progress towards the goal of change anticipation. This paper presents the results of two case studies that address objectives arising from that ultimate goal. The first case study evaluated the potential of a change source taxonomy containing the elements 'market', 'organisation', 'vision', 'specification', and 'solution' to provide a meaningful basis for change classification and measurement. The second case study investigated whether the requirements attributes of novelty, complexity, and dependency correlated with requirements volatility. While insufficiency of data in the first case study precluded an investigation of changes arising due to the change source of 'market', for the remainder of the change sources, results indicate a significant difference in cost, value to the customer and management considerations. Findings show that higher cost and value changes arose more often from 'organisation' and 'vision' sources; these changes also generally involved the co-operation of more stakeholder groups and were considered to be less controllable than changes arising from the 'specification' or 'solution' sources. Results from the second case study indicate that only 'requirements dependency' is consistently correlated with volatility and that changes coming from each change source affect different groups of requirements. We conclude that the taxonomy can provide a meaningful means of change classification, but that a single requirement attribute is insufficient for change prediction. A theoretical causal account of requirements change is drawn from the implications of the combined results of the two case studies.

**Keywords** Requirements change · Requirements evolution · Requirements volatility · Collaborative case study

## 1 Introduction

Software requirements continue to evolve during software development and maintenance, and the associated risk to cost, project schedule, and quality appeals to the need for increased understanding of the phenomena. The case studies presented in this paper are part of a family of collaborative empirical initiatives, each of which makes progress towards the ultimate goal of requirements change anticipation. Loconsole [1] provides a model to predict requirements change which utilises the observed correlation between the size of a requirements use case document and volatility. While correlation is significant, there is no claim made with respect to causality. Our approach is to use case studies to increase our understanding of the causes and effects of requirements change in order to provide an empirically founded theory upon which to base causal models. To this end, it is necessary to firstly determine an informative means by which to classify and measure change and secondly formulate causal relationships. Two temporally contiguous case studies address these objectives and considered together, the results inform the derivation of a model-based summary of the causes and effects of software requirements change.

S. McGee (✉) · D. Greer
School of Electronics, Electrical Engineering and Computer Science, Queens University Belfast, Belfast, UK
e-mail: smcgee08@qub.ac.uk

D. Greer
e-mail: des.greer@qub.ac.uk

A traditional measure of requirements volatility is a count of all amendments, deletions, and additions that occur during a specified time period. This has been used in the prediction of requirements change [1], as the measure of the health of a project [2], and to support process technique selection [3]. However, requirements changes can vary greatly in terms of their cost and value; the metric 'requirements changes = 2' that results from the addition of one change costing £100 to a second change at a cost of £1,000 is not that informative. One solution is to measure the size of the change by function points [4, 5]. However, there may be other qualities, such as value or stakeholder involvement, which are important considerations for change management. The first step, therefore, is to establish a means by which changes can be classified in order to provide an informative basis for measurement and management. Drawing upon the literature, a previous study [6] established standardised constructs, such as 'new stakeholder', or 'novelty of application', each of which was considered to be a cause of requirements change. Using card sorting, a number of project managers derived a taxonomy of change source constructs comprising the five change domains illustrated in Table 1. The objective of the first case study presented in this paper is to discover whether this classification can provide an informative basis for understanding and analysing requirements changes that occur during software development. With our industrial partner, we firstly identify those properties of change, such as cost and value, most pertinent to the characterisation and management of requirements change. We then analyse change data to determine whether these change domains can be said to represent distinct groups of changes, with significantly different average values for each change property.

**Table 1** Requirements change source domains

| Change domain | Description |
|---|---|
| Market | Differing needs of many customers, government regulations |
| Organisation | Changing strategic direction of a single customer, customer organisation considerations, political climate |
| Vision | Change to the problem to be solved, product direction and priorities, stakeholder involvement, process change |
| Specification | Change to the specification of the requirements of the established problem, resolution of ambiguity, inconsistency, increased understanding |
| Solution | Change accommodating new technical requirements, design improvement, solution elegance |

While the constructs within this taxonomy reflect situations and events that may give rise to changing requirements, it may be the case that some parts of a software application are more prone to change than others. Nuseibeh [7] maintains that attempts should be made to answer some difficult questions, among them "What classes of requirements are more stable than others, and how do we identify them?" While there have been studies undertaken to investigate the correlation between requirements volatility and process factors, such as maturity level and elicitation technique [8, 9], there are no attempts to determine whether there are attributes of the requirements themselves that predispose them to change.

Mindful of this objective, our second case study examines the requirements involved in the first case study and investigates the correlation between requirements volatility and some generic attributes of requirements (dependency complexity and novelty) selected by our industrial partner. We also consider whether this correlation is replicated across the change taxonomy.

The following research questions are addressed:

RQ1 Is the taxonomy an informative basis for change measurement?

RQ2 Do attributes of requirements predispose them to change and are they similarly prone to changes in all domains?

The software application development project under investigation was within the government sector and lasted 16 months. Inclusive of requirements changes, the project delivered 240 requirements at a total cost of 4,222 days effort. Overall, 282 requirements changes were recorded at a cost of 2,405.5 days effort.

This paper is in an extension of a previous publication [10] wherein the first case study was presented. The paper is organised as follows. Following a review of related research in Sect. 2, the research approach, including detailed research questions and data specification, is explained in Sect. 3. Section 4 presents the results of both case studies, and this is followed by a discussion in Sect. 5. Section 6 concludes and outlines the future direction of this work.

## 2 Related work

### 2.1 Requirements change classification

As far as the authors are aware, there is no existing study that uses an empirical basis for the evaluation of requirements change classifications. This is substantiated in a comprehensive literature review of change-based studies undertaken by Banested [11]. In that review, three primary objectives for empirical studies of requirements change are identified, among them the characterisation of evolution. A traditional classification of change during software development includes the categories add, modify, and delete. A number of alternative classifications have been proposed, focused upon software development, maintenance, or both, which often have the intention of meeting different objectives.

Much empirical and theoretical work focused upon software maintenance reuses or builds upon Swanson's classification [12] which includes corrective, adaptive, and perfective changes. Chapin et al. [13] provide a thorough review of literature referring to maintenance change types and propose a comprehensive classification combining a focus upon activity (such as enhancement, performance improvement, and consideration of future maintenance) with what is being changed (such as interface, properties, and business rules). This dual approach to classification echoes that of Kemerer and Slaughter [14] who intend their observations to contribute to a theory of software evolution and derive their change classification from empirical data of 25,000 requirements changes. With the objective of providing a list of change types to support impact analysis, regression testing, estimation, or software re-use, a number of classifications have been proposed which primarily concern what is being changed. These include domain-specific business features [15], object-oriented elements (such as classes, interfaces, methods) [16], and a distinction between deep structure and semantic changes [17].

By contrast, a classification that will support the needs of change anticipation must give consideration to the cause of the change. A categorisation of change types thus focused is presented by Harker [18] who divides empirically gathered requirements changes occurring during software development into five categories depending upon the source of the change—fluctuations in the organisation and product market environment, increased understanding of requirements, consequences of system usage, and changes necessary due to customer migratory or adaptation issues. Based on Harker's study, an appraisal by Sommerville [19] includes compatibility requirements relating to business process change in place of migratory and adaptation issues. Working from data held in a change control database within an industrial setting, Nurmuliani [20] catalogues software development volatility by type (addition, subtraction, deletion), origin, and reason for change. Noting that most change requests used in the study had little information about the reason for change, a further study was undertaken to classify the recorded changes [21]. This resulted in a list of 'super-ordinate constructs' classified by reason for change—product strategy, hardware/software environment changes, scope reduction, design improvement, missing requirements, clarification changes, testability, and functionality enhancement. A later study [22] added change sources of internal and external. Clearly, there is dissimilarity in the terminology used in these cause-focused classifications, and it would seem at first sight that studies to date have little commonality. This may be due to the different contextual basis of the studies, or perhaps that classification was established at different levels. It is possible, for example, that Nurmuliani's change reason of 'missing requirement' is included within Harker's change source of 'increased understanding'.

The derivation of the taxonomy, which is the focus for the case studies presented in this paper, used the expert knowledge of experienced project managers to consolidate and classify 73 change source constructs elicited from the cause-focused classifications described above [18, 19, 21] in addition to other empirical studies including [2, 9]. Using individual card sorting and workshops, a classification of change sources was derived comprising the five change domains illustrated in Table 1. In addition, an important distinction was made between constructs relating to a situation such as 'insufficient stakeholder involvement' and those relating to an event such as 'business process change'. A full taxonomy relating the domains in Table 1 to uncertainties (situational constructs) and triggers (event constructs) can be found in the "Appendix". With the initial focus on software development, the taxonomy was extended to include the maintenance phase of a project [23].

The constructs in the requirements change source taxonomy bear little synergy with change reasons derived by Nurmuliani [21] as many of these reasons such as 'missing requirement' and 'new functional feature' were considered to be consequences of other events, rather than sources of change. However, Nurmulianis 'external' change source [22] bears semblance to Harker's 'mutable' class defined as "changes that arise in response to demands outside the system" and is comparable to the combined *market* and *customer organisation* domain sources. By making the distinction between changes that occur in response to market demands and those answering to customers' organisational considerations, the taxonomy reflects the difference between customer-driven and market-driven software development. Harker's 'emergent' requirements, "direct outcomes of the process of engagement in the development activities", correspond to constructs in both the *project vision* and *requirements specification* domain. Differentiating between the domains of *project vision* and *requirements specification* reflects the difference between variation in the product to be developed and change due to better understanding of the problem. The *solution* domain has no direct parallel in any classification but reflects the reality that changes to the technical solution, though perhaps less visible, pose a risk to timely development. The means of taxonomy derivation was empirical in contrast to the theoretical approach taken by Perry [24], who states that the needs of a software system must address a number of realms. However, it is possible to draw sensible comparison between Perry's 'real world' and the domains of '*market*' and '*organisation*', his 'model of the real world' with the domain of '*vision*', his 'system requirements' with '*requirements specification*', and his 'technical theory' with '*solution*'.

### 2.2 Empirical studies of requirements volatility during software development

While there are many studies focused upon software evolution (see [11]), these concentrate upon changes made between consecutive software product releases. Further to those change classification studies discussed in the previous section, there are few experience reports of requirements change during software development. An investigative survey [9] concluded that requirements volatility has a significant impact on schedule and cost overrun, but can be constrained by effective communication, definable methodology, and requirements inspections. These findings are echoed by Ferriera et al. [8] in a survey completed by over 300 software development practitioners, who also note that higher levels of project maturity appear to reduce volatility. Jones [4] recommends elicitation and inspection techniques for mitigating the negative effects of volatility. However, Weiss [25] observes that more than 75 % of changes took a day or less to make and that relatively few changes resulted in errors. In a case study using change request data, Nurmuliani [22] reports a significant positive correlation between changes coming from external sources and the effort required to make those changes. Weiss [25] further discerns that while changes are non-localised with respect to individual components, they are localised with respect to subsystems, with the majority of changes being made in one or two subsystems. From a different perspective, Nakatani et al. [26] consider that different types of requirements mature at different times in the development process, and recommend the categorisation of groups of requirements according to maturation.

### 2.3 Requirements change prediction

From case study data collected in retrospect on fourteen Use Case Models (UCM) comprising 39 use cases, Loconsole [1] observed a significant correlation between the number of lines in a UCM and requirements volatility, and recommended modularization of larger UCMs. In contrast to the predictive approach by correlation, a number of researchers are seeking to develop and evaluate more complex causal relationships [27–29]. In particular, Fenton [29, 30], whose causal models were engineered to predict defect rates, argues that many models using small numbers of prediction variables ignore 'causal' factors such as programmer ability, or design quality. Further, models that reflect true causal mechanisms facilitate understanding and explanation as well as prediction. Most illuminating about Fenton's work is the observation that results of studies to examine the relationship between requirement complexity and defect rate disagree entirely and argues that without including some measure of 'testing effort', the relationship between complexity and defect rate is arbitrary [30].

## 3 Research approach

The two case studies presented here are part of a sequential family of studies [31] in which research questions are formulated at each stage to deepen our understanding and support our overall goal. The first case study was designed prior to the commencement of the software development project, and conducted during the entire lifecycle. Once data analysis was underway, we began the process of determining research questions for the next sequential case study that would support progress towards our longer-term goal of requirements change anticipation. Both case studies were designed in accordance with the case study guidelines outlined by Runeson and Host [32], which reviews terminology from other sources such as Wohlin et al. [33] and Yin [34] and adapts them for Software Engineering research. They provide recommended practices and an extensive empirically derived and evaluated 50-point checklist to support case study research, which includes items relating to the design of the case study, data collection, analysis, and reporting. Both case studies are single-unit studies, in which the unit of analysis in the first is the change and in the second is the requirement. Although they were based upon the same project, they were carried out sequentially and had differing data collection protocols that are detailed below.

### 3.1 Case study context

#### *3.1.1 Organisation*

Our industrial partner in this research employs 300 staff, has offices in England and Ireland, and delivers IT solutions to clients across both the public and private sectors. Most of their contracts involve a single customer, and roughly 80 % of these relate to governmental work. Of importance to collaborative research, their involvement is supported by both upper and middle management and reflects their stated initiative to become a centre of project management excellence.

#### 3.1.2 *Project*

The project of interest in this study is in the government sector, has an estimated cost in excess of a million pounds, comprises on average 15 software developers and analysts, and follows a traditional waterfall lifecycle. Beginning in April 2009, the project was completed in August 2010 and data for the first case study were collected during the entire development lifecycle. Since the software development work was the result of a successful tender, at the commencement of the project, the requirements made available to the software provider during that tendering process became the basis of the initial requirements specification effort. There were four main stakeholder groups involved, comprising the software provider and three departments on the customer side.

### 3.2 Identification of detailed research questions and data specification

#### 3.2.1 *Case study 1: Is the taxonomy an informative basis for change measurement?*

As well as supporting the needs of the academic objective, the data to be collected will also replace the company's existing change control database and be used for project retrospective analysis. Effort was therefore required to clearly identify the research questions and define mutually expedient case study data. With our industrial partner, the goal question metric (GQM) approach [35] was largely adhered to in order to firstly articulate research questions and secondly identify the case study data. Past change data were used as the basis of discussion, and this was supported by UML modelling of project processes and work products, which enabled the identification of the possible values of the variables under study. A researcher and two project managers were present at these meetings.

In addition to examining the cost and value of change, our industrial partner was interested in other issues that were considered to be important for change management. As well as discovering when change was happening, and whether it represented an opportunity to add functionality or attend to a defect in the requirements specification, they wanted to determine whether a greater number of involved stakeholders influenced the number of changes seen. Also of concern was the level of control that the project manager had of change discovery. In other words, 'with hindsight could/should this change have been discovered earlier' perhaps by the use of alternative techniques or additional resources?

To answer RQ1 "Is the taxonomy an informative basis for change measurement?" the following detailed questions are addressed:

Across change domains, is there a significant difference in:

RQ 1.1 change cost;
RQ 1.2 change value;
RQ 1.3 proportion of opportunity versus defect-related change;
RQ 1.4 the number of stakeholders involved in agreeing the change;
RQ 1.5 the activities during which changes are found; and
RQ 1.6 the level of project management control?

The selection and practical implementation of metrics to answer the research questions was not straightforward. In the main, a pragmatic approach was taken, which often required compromise between research and practice. It was considered too labour intensive to include metrics for change KLOC, or function points. However, it was also noted that 'change cost' was not always reflective of change size, since average change costs may increase as the project progresses due to supportive documentation and architectural rework as well as increased functionality. The addition of the data item 'phase' (Requirements Specification, Design and Code, System Test, User Acceptance Test) was included to allow temporally staged cost comparison. Cost was measured in days and defined as the difference between any unused portion of the previous estimate (if it existed) and the effort required to implement the change. This was agreed between the customer and the software development organisation. Expressing value in monetary terms was impossible in most situations. However, the customer had used the consideration of business value previously to prioritise requirements, so a Likert scale, subjectively assessed by the customer, was employed instead. Perhaps not surprisingly, particularly towards the latter phases of the project, the customer and software provider experienced increased difficulty in coming to agreement about whether the change represented an opportunity to add functionality or to address a defect in the requirements specification. This was evidenced in cases where the customer was expecting functionality not stated explicitly within the agreed documentation. Therefore, an allowable value—'Undefined'—was added to the original allowable values of the 'opportunity?' data item. Values for 'project management control' were provided by the project manager. The additional data items 'trigger' and 'domain' were added to relate changes to the change domains in Table 1. No classification scheme had been previously used by the company, though ad-hoc reasons for change were included in descriptive text.

Given the need to define an agreed and standardised list of activities to facilitate analysis, UML modelling sessions led by the researcher and involving project managers as available gave rise to the production of an activity diagram and a domain model that captured the development processes and work products. These were used during data review as a basis for communication and understanding. The data specified (excluding those relevant only to practice such as originator, dates) are illustrated in Table 2.

It will be noted that many of the data items are subjective measures. While appreciating the limitations imposed by non-objective measurement upon the analytical significance of results, the collection of subjective measures is becoming more widely accepted and advocated [36, 37]. In this case, some of these items, such as cost and opportunity/defect, constituted a contractual agreement between the customer and the software development organisation and were the basis for customer invoicing.

#### 3.2.2 *Case study 2: Do attributes of requirements predispose them to change and are they similarly prone to changes in all domains?*

Rather than software application–specific attributes such as 'financial system interface', the intention was to identify generic qualities that could be attributed to any set of software requirements, so that results may support the formulation of general predictive models. Once again using the GQM approach [35], a project manager and senior analyst identified requirement qualities that, in their opinion, were most likely to give rise to change. A potential list of six attributes were identified, and reduced to requirements dependency, complexity, and novelty, to reduce data collection effort. Those not included in this study are framework (COTs) usage, business criticality, and number of involved stakeholders. It was felt that both novelty and complexity had a technical as well as a business faculty. To answer our question "Do attributes of requirements predispose them to change and are they similarly prone to changes in all domains?" the following detailed questions are addressed:

RQ 2.1 Do requirements that have a high volatility in one domain also experience high levels of volatility in others?
Does requirement volatility correlate with
RQ 2.2 requirement dependency;
RQ 2.3 requirement business complexity;
RQ 2.4 requirement technical complexity;
RQ 2.5 requirement business novelty;
RQ 2.6 requirement technical novelty?

The data were to be collected solely for the purpose of research, and as such ease of collection was an important

**Table 2** Data specification for requirements changes (case study 1)

| Name/research question | Description | Allowable values |
|---|---|---|
| ID | Unique identifier | |
| Trigger (RQ 1.1–1.6) | Change source trigger e.g. Change to business case, Increased customer understanding, new technology available (nominal, objective) | A complete list can be found in the "Appendix" |
| Domain (RQ 1.1–1.6) | Change source classification. This was derived where possible from the trigger using the taxonomy in "Appendix" and reviewed (nominal, objective) | Market, organisation, vision, specification, solution |
| Cost (RQ 1.1) | Change cost expressed in days (ratio, subjective) | |
| Phase (RQ 1.1) | Project phase when change identified (nominal, objective) | Requirements (Req), design and code (D&C), system test (SysTest), User Acceptance Testing (UAT) |
| Value (RQ 1.2) | Business value to the customer (ordinal, subjective) | Very low, low, medium, high, very high |
| Opportunity? (RQ 1.3) | Opportunity or defect (nominal, subjective) | Opportunity, defect, undefined |
| Stakeholders (RQ 1.4) | Number of stakeholder roles involved agreeing the change (ordinal, objective) | 1, 2, >2 |
| Discovery_activity (RQ 1.5) | Activity during which change was identified (nominal, objective) | Provide business case, define goals, define vision, derive initial requirements, define functional requirements, define technical requirements, define quality requirements, balance requirements, approve requirements, define manual processes, derive system requirements, specify scenarios, define architecture, build and unit test, system test, specify UAT, perform UAT, implement solution |
| Project_manager_control (RQ 1.6) | Project manager's control of change identification (ordinal, subjective) | Very low, low, medium, high, very high |
| Description | Free text—qualitative | |

consideration during data specification. Rather than a Likert scale for each subjective data item, a short description for each allowable value 1–5 was identified by our industrial collaborators (see Table 3) enabling agreement between individual data collectors. The changes recorded in case study one were attributed to the appropriate requirements, and change frequency and cost for each requirement were calculated. For simplicity, the cost of a change was divided equally among the involved requirements.

Following Barry, who advocates that a measure of volatility should include dimensions of amplitude as well as frequency [38], we calculate relative percentage volatility by dividing the change cost by the original cost estimate. Change frequency can be calculated for each requirement by counting the number of changes. However, the focus of our analysis is upon change cost, since this better reflects the size of the change and is considered by our industrial partner to be a greater risk to timely and cost-effective software development. Since change cost was defined in case study one as (revised cost—unused original estimate), our definition of volatility for each requirement is as follows:

$$\text{Relative volatility} = \frac{\sum(\text{revised cost} - \text{unused previous estimate})}{\text{original estimate}} \times 100$$

This yields a volatility for each requirement, which is reflective of the extent of change to that requirement during the entire development lifecycle. In order to compare requirements volatility in each change domain, there are five measures of volatility as follows:

Vol(tot) = Total volatility for each requirement
Vol(domain 1–4) = Volatility for all changes where ‘domain’ is one of those specified in the change source taxonomy evaluated in case study one (Organisation, Vision, Specification, Solution).

## 3.3 Data collection protocol

Data for the first case study were collected for the duration of the development lifecycle. As changes were discovered, data were collected on a spreadsheet, by either the project manager or the senior analyst. Initially, bi-monthly meetings took place wherein a researcher and one member of the project team reviewed the changes gathered, though these became less frequent due in part to the urgency of project delivery. Data were made available for research at the end of each project phase, though the analysis did not begin until the end of the project. The data was owned by the company until project sign-off, whereupon the company removed any company-confidential data before the transference of the data spreadsheet to the researchers.

Upon project completion, the data for case study two were gathered. Requirements data were extracted for all 240 requirements from the requirements specification documentation by the project manager and technical architect, who each worked independently on a subset of requirements. These were entered into a spreadsheet, and in situations where there was uncertainty about individual data items, it was agreed by all parties that they would be left blank, and reconsidered during data validation.

## 3.4 Data validation

Since change data in the first case study were collected as it occurred, not only were the data fresh in the minds of the practitioners, but there were opportunities for data review by a researcher during the project lifecycle. We believe these factors contributed significantly to the quality of the data. Effort was made to ensure that correct values had been entered against each change record collected in case study one. Data triangulation is a process by which an item of data is verified from two or more sources, either by two participants (observer triangulation) or by two means of data collection (methodological triangulation) [32]. This increases the precision of empirical research especially in cases of subjective measurement [32]. Observer triangulation was applied in the case of ‘cost’, and ‘opportunity’ by the customer and project manager and remaining data items by project manager and senior analyst. Methodological triangulation between the qualitative ‘change description’ and the quantitative factors was achieved during the change review meetings with a researcher and project manager. A number of changes, randomly selected, were reviewed at these meetings. Roughly 60 % of changes were re-examined, though data quality was high and only a small percentage of changes were amended, usually due to the completion of missing data items.

Once the spreadsheet of data from case study two was made available for analysis, data were checked visually for completeness. Since both practitioners were familiar with all of the requirements, data items left blank by one participant were requested from the other participant, and a small sample of data was also sent for cross-validation.

## 3.5 Data review process

During the data review meetings in the first case study, in addition to data validation, the trigger placement within each change domain was reviewed and the taxonomy amended as required. For example, the change trigger ‘New Market Technology’ was added to the domain of *market* to differentiate it from the trigger ‘New technology’

**Table 3** Data specification for requirements (case study 2)

| Name/research question | Description | Allowable values |
|---|---|---|
| ID | Unique identifier | |
| Change ID (s) (RQ 2.1–2.6) | Change identifiers | |
| Cost (RQ 2.1–2.6) | Effort estimate of original requirement expressed in days (ratio, subjective) | |
| Dependency (RQ 2.2) | Pervasiveness of requirement (ordinal, subjective) | 1. Requirement has no dependency to any others<br>2. Requirement has dependencies with non-functional requirements, for example, security, performance<br>3. Requirement may have dependencies upon non-functional requirements and also dependencies to other requirements within a single business functional area, e.g. maintain address, manage messages<br>4. Requirement had dependencies to other requirements within a single business function and consumes services from generic components (those without business rules)<br>5. Requirement is a consumer of many other requirements throughout the solution |
| Business complexity (RQ 2.3) | Difficulty of requirement expression by the customer (ordinal, subjective) | 1. Simple—can be captured with a checklist<br>2. Some thought needed of documented process<br>3. Multiple pages captured via use cases with only a few variant flows<br>4. Requires an experienced user. Multiple variant business flows that need supporting documentation styles<br>5. Very challenging requiring an experienced user. Multiple media and techniques such as animation, wire frames, prototypes |
| Technical complexity (RQ 2.4) | Difficulty of technical solution (ordinal, subjective) | 1. Simple—set of parameters to proven code<br>2. Some thought needed—linear script met by one single language/tool/product<br>3. Either uses many technologies, new/untried technologies or has many interfaces<br>4. Requires two of: many technologies, new/untried technologies, many interfaces<br>5. All three of: many technologies, new/untried technologies, many interfaces |
| Business novelty (RQ 2.5) | How novel is the requirement to existing business processes? (ordinal, subjective) | 1. Part of day job<br>2. Occasionally used<br>3. Occasionally used but not in this job<br>4. Familiar with concept<br>5. Entirely new idea |
| Technical novelty (RQ 2.6) | How novel is the technical requirement to the design, build and test teams? (ordinal, subjective) | 1. Very similar to something done before by many of the team<br>2. Some of the team have done something similar<br>3. We have examples but no direct experience<br>4. We are aware that something similar has been done before but we have no examples<br>5. We believe we may be able to do this but we aren't sure how |
| Cost total changes (RQ 2.2–2.6) | Total cost of all changes made to that requirement (ratio, calculated from 'cost') | |
| Domain change cost (RQ 2.2–2.6) | Cost of changes in each domain made to that requirement (one data item for each domain) (ratio, calculated from 'cost') | |

**Table 3** continued

| Name/research question | Description | Allowable values |
|---|---|---|
| Volatility (RQ 2.2–2.6) | $\frac{\sum(\text{change cost})}{\text{original estimate}} \times 100$ (ratio, calculated from 'cost') | |
| Domain volatility Vol(Org), Vol(Vis), Vol(Req Spec), Vol (Sol) (RQ 2.2–2.6) | Volatility for each domain (four data items) (ratio, calculated from 'cost') | |

residing in the *solution* domain. Also, some overlap between triggers in the *specification* domain and those in the *vision* domain were identified. For example, 'Increased Customer Understanding' could change both the *vision* and the *specification*. A revised taxonomy can be found in the "Appendix".

### 3.6 Analysis procedures

Descriptive tables and graphs are complemented by statistical methods to test data relationships in order to answer the research questions. These procedures were selected on the basis of underlying distribution and variable scale assumptions. Data pertaining to change cost did not follow a normal distribution (see results Sect. 4.2 'The Cost of Change'), and many of the data items have a nominal (categorical) scale as indicated in Tables 2 and 3. What follows are short descriptions of the appropriate statistical test, and the research questions that they are employed to address.

### 3.7 Statistical methods

The Kruskal–Wallis test allows the comparison of groups of data scores (ordinal or scale type) and tests whether the scores could be thought to come from different groups, that is, that there is a significantly different central tendency for each group. Simply put, when data do not conform to a normal distribution (as is the case with the change costs recorded in case study one and the volatility used in case study two), using this test is one of the ways groups can be compared without reference to mean values. This test uses score rankings in place of actual scores to perform the statistical test. Post hoc procedures include the examination of pairs of groups to determine where the main differences lie (Mann–Whitney test). These tests will be used to examine the change costs observed for changes within each change domain (research question 1.1). The chi-squared test looks for relationships between two categorical variables, by comparing the observed frequencies in certain categories with expected frequencies. This test is appropriate for examining the ordinal scale for value as well as the nominal variables selected to represent managerial considerations (research questions 1.2–1.6). Spearman's rho is also based upon ranks rather than scores, and tests for correlation between two sets of data that do not display a normal distribution. We can test for positive or negative correlation (two-tailed), or positive correlation only (one-tailed). A one-tailed test is used to determine whether requirements that have a high degree of change in one domain also have a similarly high level of change in other domains (research question 2.1). A two-tailed test is used to explore the correlation between requirement dependency, complexity and novelty, and volatility (research questions 2.2–2.6). The reader is referred to [39] for details of these tests.

### 3.8 Limitations of study validity

Following Yin [34], Runeson [32] identifies four aspects of validity which may limit the trustworthiness of the results—construct, internal, external, and reliability. Construct validity reflects the extent to which the data specification is similarly understood by all parties involved, and accurately meets the needs of the research question. Since the data were discussed and specified through meetings with researcher and practitioners, there is a shared understanding of the meaning of the data items. They were defined specifically to answer the questions in these studies. In the first case study, two of the four subjective measures ('cost' and 'opportunity') were agreed by two roles (customer and software development project manager or analyst), ensuring agreement of both interpretation and collection. Similarly, in the second case study, all subjective measures involving cost (those collected and those derived) were agreed between two practitioner roles. The remainder of the subjective measures in the second case study was collected by only one participant. However, a subset of data collected by each practitioner was sent to another practitioner for validation, which resulted in no changes.

Internal validity is of concern when causal relations are examined, given that a third invisible (confounding) factor that is not included in the study may affect the results. The first case study is not concerned with causal relations. However, the second begins our investigation of requirements change causality through the examination of correlation. In this instance, care has been taken with regard to causal claims, and awareness of the presence of confounding factors, especially in the light of other research, enriches our understanding of results.

No claims to external validity can be made, in that without study replication, it is impossible to draw conclusions about the application of results to software development environments that are different to the study context described here. Ideally, this study should be replicated firstly within a similar context for results comparison before applying it in alternative software development environments. Subsequently, widening software development context reflects Sjoberg's recommendation [40] to "formulate [research] scope relatively narrowly to begin with and then extend it gradually".

Attempt has been made to ensure study reliability which is defined as "the extent to which the data and analysis are dependent upon specific researchers" [32]. Threats to this aspect of validity are, for example, if it is not clear how data were specified and collected. Details of data specification and collection protocol are provided for use by other researchers. Efforts were made to encourage a high level of data quality through data reviews with a practitioner and a researcher and cross-validation (where a second industrial participant checks the data collected). While the results rely upon subjective measures, the participants in this study can be considered experts since their knowledge of both the business and the software development domain is extensive. All participants have at least 10 years' experience in their respective software development roles and at least 5 years' experience of software application development within the government sector.

## 4 Results

### 4.1 Overall look at changes during the developmental lifecycle

From project inception to delivery, over a period of 16 months, a total of 282 requirements changes were recorded, at a cost of 2,405.5 days effort which represents more than 50 % of the final project cost of 4,222 days. Table 4 illustrates the phase during which these changes were discovered, and the change source domain. Since the project followed a strict traditional waterfall process, the phases are temporally contiguous.

As can be seen, a high proportion of these changes occurred during the User Acceptance Testing and Design and Code phases of the project. Since this was a project intended for a particular customer- rather than a market-based initiative, it is not surprising that there is only one market change (which related to following market trends in COTS usage). This change (costing 30 days effort) was removed for all subsequent analysis and means that this study is limited to the examination of the remaining four change domains. In addition, changes involving only requirement deletions (12) at zero cost are excluded from future analysis, reducing the total number of changes considered from 282 to 269.

**Table 4** Number of changes per phase per domain

| | Req. | D&C | SysTest | U.A.T. | Total |
|---|---|---|---|---|---|
| Market | 0 | 1 | 0 | 0 | 1 |
| Organisation | 30 | 4 | 0 | 0 | 34 |
| Vision | 15 | 1 | 1 | 7 | 24 |
| Specification | 22 | 58 | 5 | 102 | 187 |
| Solution | 0 | 33 | 3 | 0 | 36 |
| Total | 67 (24 %) | 97 (34 %) | 9 (3 %) | 109 (39 %) | 282 |

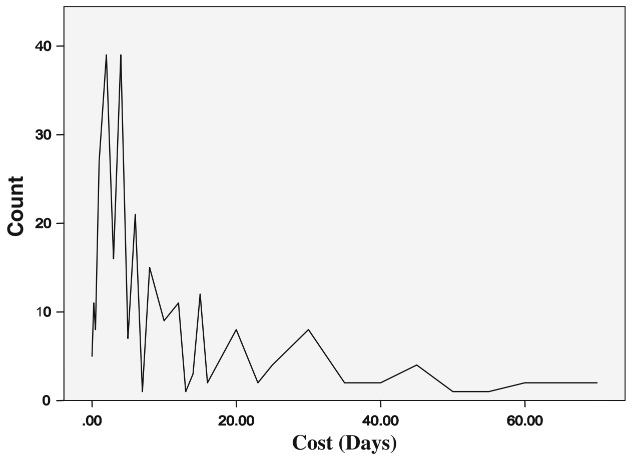


**Fig. 1** Frequencies of change costs across all change domains

### 4.2 The cost of change

The analysis of change cost discounts the 12 deleted requirements. Figure 1 illustrates the frequencies of change costs for the entire project across all change domains. Change cost is not normally distributed (Shapiro–Wilk $W = 0.669$, $p < 0.001$) and is highly positively skewed due to the lower limit of zero cost being fixed. Since the examination of mean values for cost is therefore less meaningful, future analysis uses the median values as representation of central tendency.

Table 5 shows a breakdown of these costs as they pertain to project phase and change domain. Although the most significant cost was experienced during the initial phase of requirements confirmation, a high percentage of change cost occurred during User Acceptance Testing

**Table 5** Change cost per phase per domain

| | Req | D&C | SysTest | UAT | Total |
|---|---|---|---|---|---|
| Organisation | 638.0 | 64.0 | 0.0 | 0.0 | 702.0 |
| Vision | 266.0 | 5.0 | 2.0 | 163.0 | 436.0 |
| Specification | 193.9 | 222.0 | 4.5 | 737.0 | 1,156.5 |
| Solution | 0.0 | 78.0 | 2.5 | 0.0 | 81.0 |
| Total | 1,097.0 (46 %) | 369.5 (16 %) | 9.0 (0.4 %) | 900.0 (38 %) | 2,375.5[a] |

[a] Excludes market change at a cost of 30 days

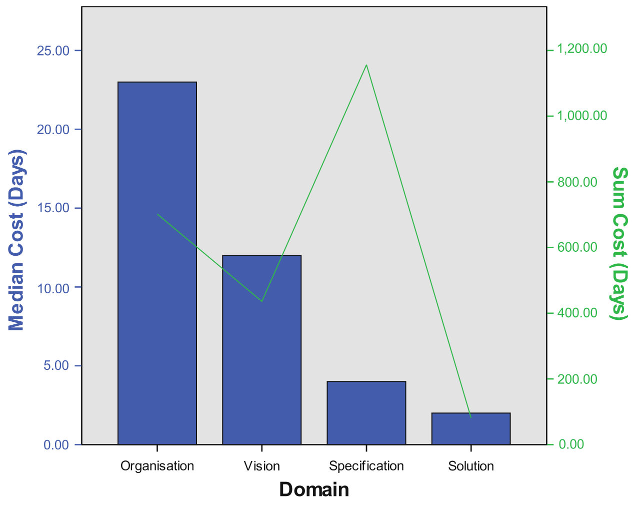


**Fig. 2** Median and total cost for each change domain

(38 %). By far, the largest percentage of cost came from the *specification* domain (46 %).

Figure 2 illustrates a comparison between median and total cost for each change domain. While total costs are significantly higher in the *specification* domain, the medians of these costs illustrate that on average changes due to *organisation* changes are the most expensive, followed by changes to the *vision,* then *specification*, and with the lowest average cost being in the *solution* domain. The Kruskal–Wallis test indicated that the differences are significant ($H(3) = 75.038, p < 0.001$) to the extent that these changes could be thought of as coming from different groups. While this indicates that there is a difference overall, it does not inform us of where the major differences lie. Performing selected Mann–Whitney tests to test for differences between adjacent domains reveals that the median of the domain of *organisation* does not vary significantly to that of *vision* ($U = 229.5$, $z = -1.787$, $p > 0.05$), but that vision differs from specification ($U = 851.500$, $z = -4.879$, $p < 0.001$) and similarly *specification* differs from *solution* ($U = 1{,}901$, $z = -4.006$, $p < 0.001$). Since costs change over time, it is

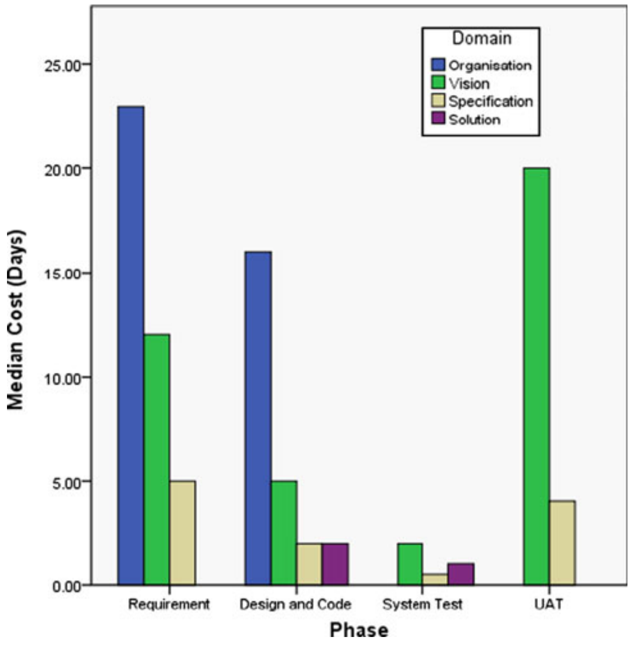


**Fig. 3** Median change costs per domain per phase

useful to explore the differences in domain costs for each phase of the project.

Figure 3 shows median change costs for each domain in each phase. It can be seen that the trend in all domains is generally reflective of the results we saw when all phases were included, with the most expensive changes occurring in the *organisation* domain and the least expensive in the *solution*. Costs tend to fall in the second and third phases and in the case of the *vision* and *specification* domain rise sharply during User Acceptance Testing. From quantitative data alone, it is impossible to assess whether this rise in cost is due to increased change size (more function points per change) or rework of existing code and architecture. Performing the Kruskal–Wallis test on phase one data alone indicates that there is significant difference in the costs in the three domains of *organisation, vision*, and *specification* ($H(2) = 15.239$, $p < 0.001$).

Similarly, overall cost medians are significantly different in phase two ($H(3) = 10.692$, $p < 0.05$). As can be seen though, there is no difference in this phase between costs in the domains of *specification* and *solution*. It is not possible to do median comparisons for phases three and four due to insufficient data.

The results support an affirmative answer to *RQ1.1*, "Across change domains, is there a significant difference in change cost?" Change costs are not consistent across change domains; the most expensive changes come from the domain of *organisation* and costs decrease through the domains of *vision, specification*, and *solution*.

Table 6 illustrates the proportion of subjectively assessed value accruing from these changes across the change

**Table 6** Change value per domain

| | Value | | | | | |
|---|---|---|---|---|---|---|
| | Very low | Low | Medium | High | Very high | Total |
| Organisation | 0 | 7 | 9 | 5 | 6 | 27 |
| Vision | 2 | 9 | 11 | 0 | 2 | 24 |
| Specification | 102 | 64 | 13 | 3 | 0 | 182 |
| Solution | 33 | 2 | 1 | 0 | 0 | 36 |
| Total | 137 (51 %) | 82 (30 %) | 34 (13 %) | 8 (3 %) | 8 (3 %) | 269[a] |

[a] Market change and changes representing requirements deletions removed

**Table 7** Numbers of changes by domain categorised as opportunity, defect, or undefined

| | Opportunity | Defect | Un-defined | Total |
|---|---|---|---|---|
| Organisation | 24 | 2 | 1 | 27 |
| Vision | 18 | 5 | 1 | 24 |
| Specification | 104 | 62 | 16 | 182 |
| Solution | 13 | 20 | 3 | 36 |
| Total | 159 (59 %) | 89 (33 %) | 21 (8 %) | 269[a] |

[a] Market change and changes representing requirements deletions removed

domains. Just over half of all changes are of a very low value (51 %), and of those, the greatest proportion (74 %) was in the *specification* domain.

The highest value changes are only in the domains of *organisation* and *vision*. By contrast, most of the changes in the solution domain are of very low value (91 %) A chi-squared test (performed with value 'high' and 'very high' changes added together due to low frequencies in these groups) reveals that there is an uneven distribution of values across the four domains ($X^2(9) = 144.354$, $p < 0.001$).

These results indicate a positive answer for *RQ1.2*, "Across change domains, is there a significant difference in change value?" From the perspective of value, these requirements changes could be thought of as coming from different groups according to the change domains specified. The highest value changes come from the domain of *organisation* and the lowest from *solution.*

## 4.3 Opportunity/defect

Changes can represent opportunities to enhance functionality as well as correction of previous errors. Table 7 illustrates how these are spread across the change domains. Changes representing an opportunity comprise the majority of changes in the domains of *organisation* (89 %), *vision* (75 %), and *specification* (57 %) and represent a total cost of 1,677 days effort.

Defects costing a total of 559 days effort are more often the cause of change in the *solution* domain (56 %). When the customer and software provider have not been able to arrive at an agreement about whether the change represents an opportunity or a defect, it has been referred to as 'Undefined'. In this case, most of these changes were related to assumptions regarding functionality implementation methods. They represent a small proportion of all changes (<10 %), have a cost of 139.5 days effort, and are mostly in the *specification* domain. The chi-squared test is significant ($X^2(6) = 21.662$, $p = 0.001$), confirming that there is an uneven distribution of opportunity change across these change domains.

*In answer to RQ 1.3*, "Across change domains, is there a significant difference in the proportion of opportunity-versus defect-related change?" these results show that the proportion of changes representing an opportunity as opposed to a defect are not evenly spread across domains. Opportunity change is more often seen in the domains of *organisation* and *vision*, while defects predominate the domains of *specification* and *solution.*

## 4.4 Number of stakeholders

As the software provider was considered a stakeholder, changes involving only one stakeholder were either those that required decisions to be made without customer involvement or those where a single customer stakeholder group was able to make changes that required only agreement rather than negotiation with the software provider. A stakeholder number of '3' means three or more stakeholders groups involved in agreeing the change. Table 8 illustrates stakeholder groups involvement in each change domain. In all domains, there is greater proportion of changes requiring more than one stakeholder group. (89 % of *organisation* changes, 96 % of *vision* changes, 90 % of *specification* changes, and 56 % of *solution* changes).

However, in the domains of *organisation* and *vision*, there are proportionally more changes requiring the involvement of three or more stakeholders (22 and 33 %,

**Table 8** Changes categorised as numbers of stakeholder groups involved in agreeing change per domain

| | Stakeholder groups | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | Total |
| Organisation | 3 | 18 | 6 | 27 |
| Vision | 1 | 15 | 8 | 24 |
| Specification | 19 | 149 | 14 | 182 |
| Solution | 16 | 20 | 0 | 36 |
| Total | 39 (14 %) | 202 (75 %) | 28 (10 %) | 269[a] |

[a] Market change and changes representing requirements deletions removed

respectively) compared with the *specification* domain (8 %) and *solution* (0 %). In the *solution* domain, we see a greater proportion of single stakeholder changes (44 %) than in any other domain. A chi-squared test indicates that there is dissimilarity in these domains when considering the numbers of stakeholder groups usually involved in the change ($X^2(6) = 50.795$, $p < 0.001$). Interestingly, median costs also rise as the number of involved stakeholders increases. The median cost when one stakeholder group is involved is 2 days effort, compared with 4 days for 2 stakeholders and rising sharply to 10 days for 3 or more stakeholder groups.

These results indicate a positive answer to *RQ 1.4*, "Across change domains, is there a significant difference in the number of stakeholders involved?" More often, a higher number of stakeholders are involved with *organisation* and *vision* change.

## 4.5 Discovery activity

As shown in Table 9, a high proportion of *specification* changes were discovered during UAT (55 %), though many of the *organisation* changes (63 %) and *vision* changes (46 %) were discovered during the 'define functional requirements' activity. *Solution* changes in the main were discovered during build and test (64 %).

A visual analysis of these changes, presented in Table 9, would suggest therefore that changes in different domains are discovered during different activities in the developmental lifecycle. However, there are insufficient data to perform a chi-squared test for inequality of change discovery activity spread among domains. Subsequent to this case study and publication [10], this question was revisited and the activities were grouped in order to reduce their number. This resulted in the four activities contained in Table 10. Demo (demonstration) was understood to be any communication or review by any stakeholder of a part of the application, either by presentation or by prototype demonstration to any number of people. 'Translate' described the process of using a previously created work product to create a new one, for example creating code from specification. 'Test' is any activity related to testing, and 'external' is an event outside of the remit of the project manager. A high percentage (59 %) of *organisation* change was discovered through external activity, whereas all *solution* changes were discovered through translation activities. *Vision* changes are more frequently discovered through demonstration activities (50 %). A high percentage of *specification* changes (63 %) were discovered through activities related to testing. A chi-squared test confirms that there is a significant difference between activities that

**Table 9** Change discovery activity per change domain

| | Org. | Vis. | Spec. | Sol. | Total |
|---|---|---|---|---|---|
| Define vision | 1 | 1 | 0 | 0 | 2 |
| Define functional reqs | 17 | 11 | 14 | 1 | 43 |
| Define technical reqs | 1 | 1 | 3 | 0 | 5 |
| Balance Reqs | 0 | 0 | 1 | 0 | 1 |
| Approve bus reqs | 1 | 0 | 0 | 0 | 1 |
| Derive system reqs | 4 | 2 | 5 | 2 | 13 |
| Specify scenarios | 0 | 0 | 2 | 0 | 2 |
| Define architecture | 1 | 0 | 0 | 2 | 3 |
| Build and unit test | 0 | 1 | 25 | 23 | 49 |
| System test | 0 | 1 | 5 | 8 | 14 |
| Specify UATs | 0 | 0 | 26 | 0 | 26 |
| Perform UAT | 0 | 7 | 101 | 0 | 108 |
| No activity | 2 | 0 | 0 | 0 | 2 |
| Total | 27 (10 %) | 24 (9 %) | 182 (68 %) | 36 (13 %) | 269[a] |

[a] Market change and changes representing requirements deletions removed

**Table 10** Change discovery event per change domain

| | Event | | | | Total |
|---|---|---|---|---|---|
| | Demo | Translate | Test | External | |
| Domain | | | | | |
| Organisation | 7 | 4 | 0 | 16 | 27 |
| Vision | 12 | 7 | 1 | 4 | 24 |
| Specification | 7 | 61 | 114 | 0 | 182 |
| Solution | 0 | 36 | 0 | 0 | 36 |
| Total | 26 (10 %) | 108 (40 %) | 115 (43 %) | 20 (7 %) | 269[a] |

[a] Market change and changes representing requirements deletions removed

**Table 11** Extent of management control over changes per domain

| | Project management control | | | | | Total |
|---|---|---|---|---|---|---|
| | Very low | Low | Medium | High | Very high | |
| Organisation | 7 | 4 | 16 | 0 | 0 | 27 |
| Vision | 2 | 3 | 18 | 1 | 0 | 24 |
| Specification | 3 | 13 | 126 | 31 | 9 | 182 |
| Solution | 1 | 1 | 5 | 18 | 11 | 36 |
| Total | 13 (5 %) | 21 (8 %) | 165 (61 %) | 50 (19 %) | 20 (7 %) | 269[a] |

[a] Market change and changes representing requirements deletions removed

discover change in each of the domains ($X^2(9) = 265.4$, $p < 0.01$).

These results indicate a positive answer to *RQ 1.5*, "Across change domains, is there a significant difference in the activities during which changes are found?"

## 4.6 Project management control

As stated, the process followed in this project adhered to a waterfall approach wherein attempts are made to define all requirements at the beginning of the project. 'Project manager control' captures a subjective assessment by the project manager regarding the ease by which these changes may have been discovered earlier. It was felt that some changes would have been impossible to find (project manager control = 'very low') even with improved techniques. An example of a change such as this is changing the list of Internet browsers that the system was intended to be compatible with, following an organisational study of browser usage. By contrast, those that the project manager believed may have been uncovered with more time, or different techniques (project manager control = 'very high') would include changes such as screen layout modification.

These results, illustrated in Table 11, indicate that all of the changes over which the project manager has the most control lie within the domains of *specification* and *solution*. There is a proportionally greater volume of 'very low' control change in the domain of *organisation* (26 %) than in *vision* (4 %), *specification* (2 %), and *solution* (3 %). As it stands, the data are insufficient to perform a chi-squared test. However, when project manager control = 'very low' and 'low' and project manager control = 'high' and 'very high' are compressed into single categories, the data meet the criteria necessary for this test and are significant ($X^2(6) = 85.113$, $p < 0.001$), indicating that in general the level of project management control differs according to the domain from which the change arises.

These results indicate a positive answer to *RQ 1.6*, "Across change domains, is there a significant difference in the level of project management control?" It was considered that a higher proportion of *solution* and *specification*

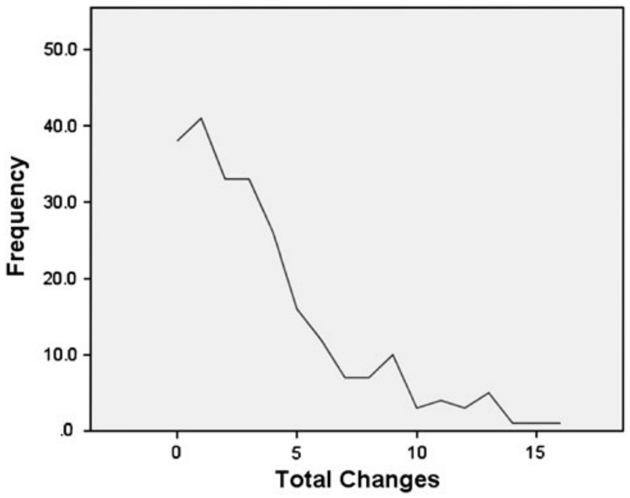


**Fig. 4** Frequency of total changes per requirement

changes could have been discovered earlier by the use of alternative approaches or techniques, while much *organisation* and *vision* change would have occurred regardless of the analysis effort.

## 4.7 Overall look at changing requirements

The developed application had a total of 240 requirements, of which only 40 were change free. Per requirement, change counts range from 0 to 16 with the mean being 3.5 changes per requirement for the entire development lifecycle. However, on average, a requirement changes 2.5 times from *specification* changes, in contrast to an average of less than one in each of the other change domains. Figure 4 shows the frequency of changes made to requirements. Thirty requirements were subject to volatility (measured by cost, see Sect. 3.2.2) greater than 500 %, and the highest volatility recorded was 1,831 %. Figure 5 shows the frequency of total volatility values per requirement.

A much higher percentage of requirements (75 %) underwent changes coming from the domain of *specification* compared to other domains where between 23 and 31 % requirements were affected. Relating to Boehm's observation

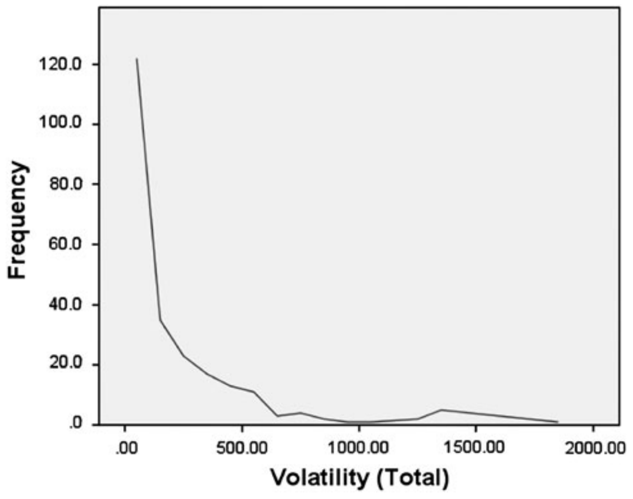


**Fig. 5** Frequency of volatility per requirement

[41] that 80 % of maintenance effort was absorbed by 20 % software modules, a Pareto analysis reveals that 80 % change cost actually involves 31 % of requirements. Interestingly, 80 % of *organisation* and *vision*-related rework were attributed to less than 11 % of the requirements. This is in contrast to the 36 % of requirements contributing to 80 % of the *specification* domain change cost.

### 4.8 Patterns of requirements volatility across change domains

Further scrutiny of this Pareto analysis reveals that there is no single requirement contributing to 80 % of the change cost in all change domains. However, a one-tailed Spearman's rho test[1] reveals that there is a weak but significant correlation between volatility in the domain of *organisation* and all other change domains. {(Vol(org) and Vol(vision) = 0.38, $p < 0.05$; $r$ (Vol(org) with Vol(req Spec)) = 0.21, $p < 0.05$; $r$ (Vol(org) with Vol(sol)) = 0.14, $p < 0.05$}. There is no significant correlation between volatility in the remaining domains. For example, a requirement experiencing *vision* volatility is not necessarily also subject to *specification* domain changes.

The results suggest a negative answer to *RQ 2.1*, "Do requirements that have a high volatility in one domain also experience high levels of volatility in others?" There is a weak correlation between volatility in the domain of *organisation* and all other domains.

### 4.9 Requirement dependency

Figure 6 shows the distribution of median volatility by requirement dependency for all changes. In this graph, and

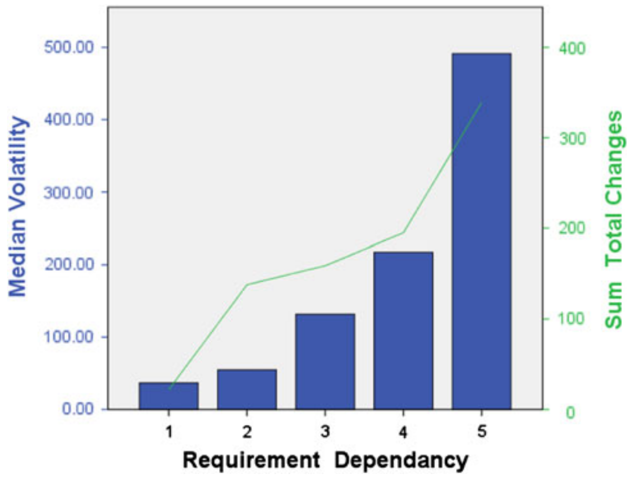


**Fig. 6** Median volatility for requirement dependency

all the graphs to follow, there are two measures illustrated. The bars indicate the median volatility, and the line shows a sum of the changes that occurred during the development lifecycle. It is clear from this that a higher requirements dependency gives rise to both increased change frequency and higher volatility. This is reflected in the Spearman's rho test that identifies significant positive correlation between dependency and Vol(tot) [$r = 0.587, p < 0.01$]. A further exploration of the domain-specific volatility reveals a similar pattern (see Fig. 7), and Spearman's rho confirms that requirements dependency is positively correlated with volatility in all domains [Vol(Org) = 0.563, $p < 0.01$; Vol(Vis) = 0.380, $p < 0.01$; Vol(Req Spec) = 0.342, $p < 0.01$; Vol(Sol) = 0.203, $p < 0.01$]. The strongest correlations are with total volatility and *organisation* volatility.

In answer to *RQ 2.2*, "Does requirement volatility correlate with requirement dependency?" the results indicate a positive correlation in all domains between requirements dependency and requirement volatility.

### 4.10 Business complexity

There is a less clear relationship between business complexity and volatility. Indeed, as can be seen in Fig. 8, frequency of changes is highest when business complexity is at level 2 (Value 2: 'some thought needed …' in Table 3). A Spearman's rho test confirms that there is no significant correlation between business complexity and any of our measures of volatility. [Vol(Tot) = −0.76, $p > 0.05$; Vol(Org) = 0.07, $p > 0.05$; Vol(Vis) = 0.06, $p > 0.05$; Vol(Req Spec) = −0.71, $p > 0.05$; Vol(Sol) = 0.07, $p > 0.05$].

The results indicate a negative answer to *RQ 2.3*, "Does requirement volatility correlate with requirement business complexity?"

[1] For simplicity, the results of the Spearman's Rho test replace '*r*' with the associated volatility description.

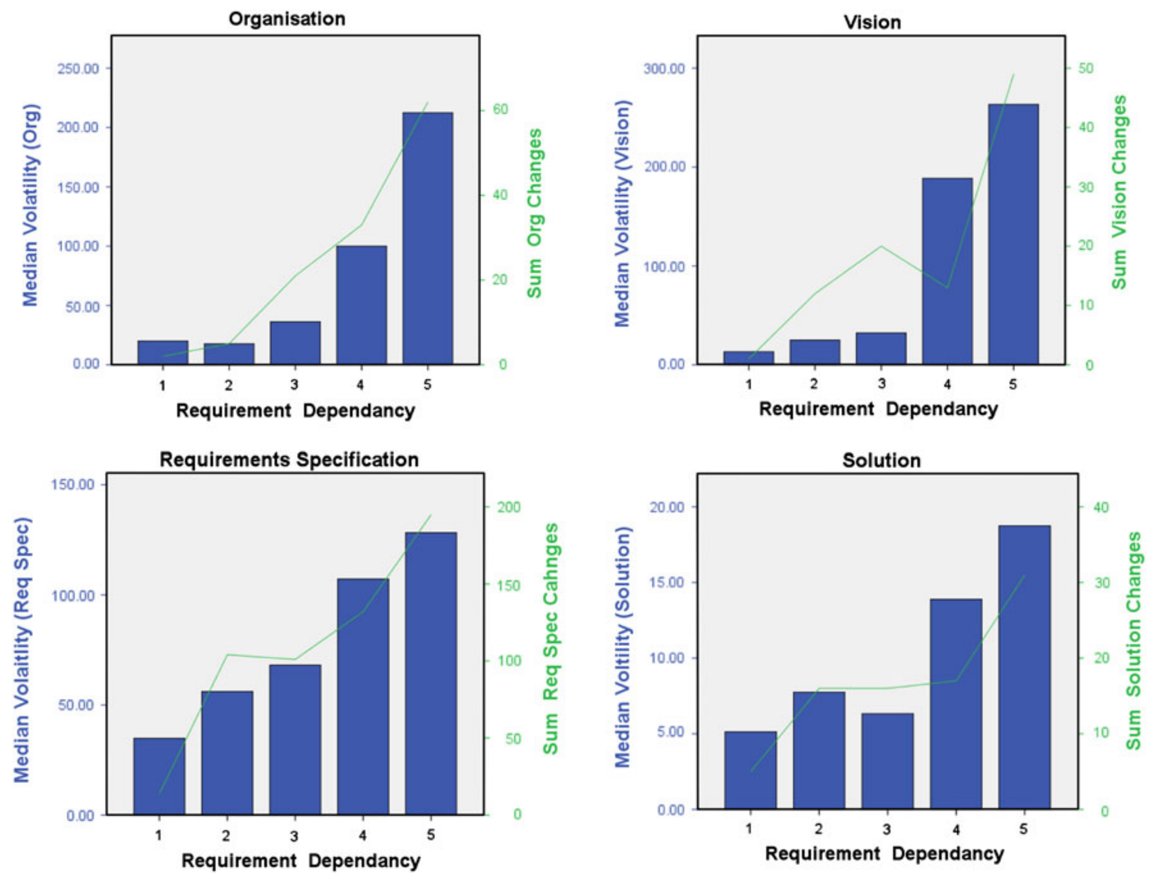


**Fig. 7** Median domain volatility for requirement dependency

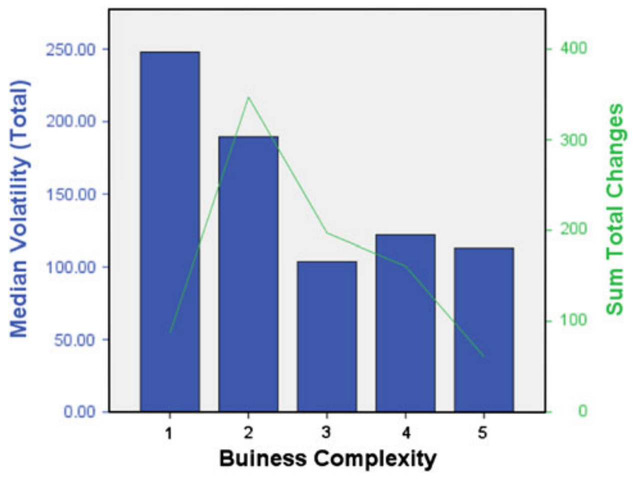


**Fig. 8** Median volatility for business complexity

## 4.11 Technical complexity

Once again, the distribution of volatility for technical complexity (Fig. 9) is not straightforward and would seem to suggest that less technically complex requirements are changing more frequently and have higher rates of volatility. The Spearman's rho test confirms that both total volatility and *specification* volatility are weakly negatively correlated with technical complexity. There is no significant correlation in the remainder of the domain volatilities. [Vol(Tot) $= -0.17$, $p < 0.05$; Vol(Org) $= 0.1$, $p > 0.05$; Vol(Vis) $= 0.10$, $p > 0.05$; Vol(Req Spec) $= -0.21$, $p < 0.01$; Vol(Sol) $= 0.144$, $p < 0.05$].

In answer to *RQ 2.4*, "Does requirement volatility correlate with requirement technical complexity?" the results indicate that technical complexity negatively correlates with requirements volatility, but this correlation is not observed in all change domains.

## 4.12 Business novelty

The somewhat confusing distribution illustrated in Fig. 10 not only reveals that though, in general, volatility is lower for less novel requirements, the frequency of change is higher. Further exploration of the distributions for change domain volatilities reveals a contrasting distribution shape between *organisation* volatility and *specification* volatility (se Fig. 11). A Spearman's rho confirms a moderately strong positive correlation between *organisation* volatility and business novelty and a weaker negative correlation with *specification* volatility. [Vol(Tot) $= 0.08$, $p > 0.05$;

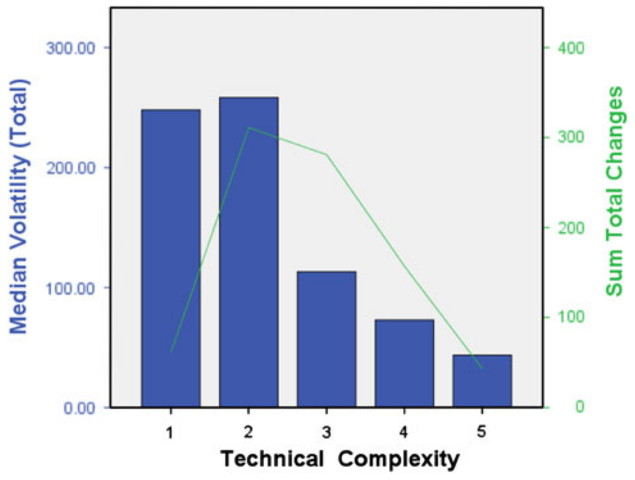


**Fig. 9** Median volatility for technical complexity

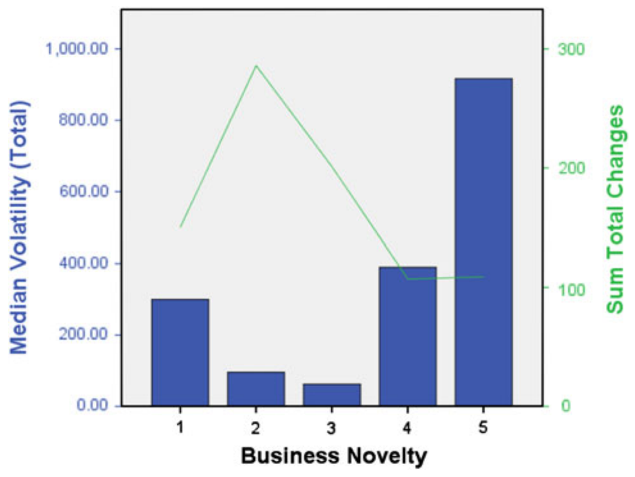


**Fig. 10** Median volatility for business novelty

Vol(Org) $= 0.423$, $p < 0.01$; Vol(Vis) $= 0.39$, $p < 0.01$; Vol(Req Spec) $= -0.25$, $p < 0.01$; Vol(Sol) $= 0.27$, $p < 0.01$]. There is no significant correlation with total volatility, and the same is true for the remainder of change domain volatility.

*In answer to RQ 2.5*, "Does requirement volatility correlate with business novelty?" the results provide no evidence that there is a correlation between total volatility and business novelty. However, *organisation* volatility is positively correlated, while *specification* volatility is negatively correlated.

### 4.13 Technical novelty

A more straightforward distribution of volatility is observed when we consider the attribute technical novelty. It can be seen in Fig. 12 that requirements with a higher technical novelty have higher overall volatility. However, a Spearman's rho test is illuminating in that the only change domain where volatility even weakly positively correlates with technical novelty is the requirements specification domain. [Vol(Tot) $= 0.28$, $p < 0.01$; Vol(Org) $= 0.07$, $p > 0.05$; Vol(Vis) $= 0.07$, $p > 0.05$; Vol(Req Spec) $=$ $0.362$, $p < 0.01$; Vol(Sol) $= 0.06$, $p > 0.05$]. There is no significant correlation in any other domains.

*In answer to RQ 2.6*, "Does requirement volatility correlate with technical novelty?" the results indicate a weak correlation which is only evident in the domain of *specification*.

## 5 Discussion of results

The analysis of data collected in the first case study data has allowed us to assess whether there is any correlation between the change taxonomy groups and change attributes reflecting change size, value, stakeholder involvement, and project management control. Results indicate that there is a distinction between changes falling into the classifications in this taxonomy. Not only do changes arising due to customer *organisation* changes cost more on average and accrue more value but they were also considered by the participants to be more difficult to uncover, and generally involve the agreement of a higher number of stakeholder roles. This is in stark contrast to *solution* changes which are in the main controllable and less costly than changes from other sources. These results are in accordance with those of Nurmuliani [22] who observed that changes coming from sources external to the project require more effort to implement. The implication of this analysis is that the management approach and assessment of risk to project schedule, cost, or quality should be reflective of different type of changes, and that change measurement and monitoring would be more informative if classified in this way. For example, to reduce the uncertainty associated with higher risk of customer organisational change, it would be necessary for project analysts to broaden the scope of application analysis to wider organisational concerns. As well as differences in cost and value, there are also differences in management considerations between changes due to *vision* changes and those coming from *specification.* While it was possible to uncover changes from *specification* issues during build and test, any vision changes not already discovered were not found at this stage and remained until User Acceptance Testing.

Maintaining change data in this way across multiple projects would allow software providers to assess the efficacy of analysis techniques and guide future process selection decisions. For example, the high number of *vision* changes discovered during hands-on system usage during User Acceptance Testing may provide empirical support

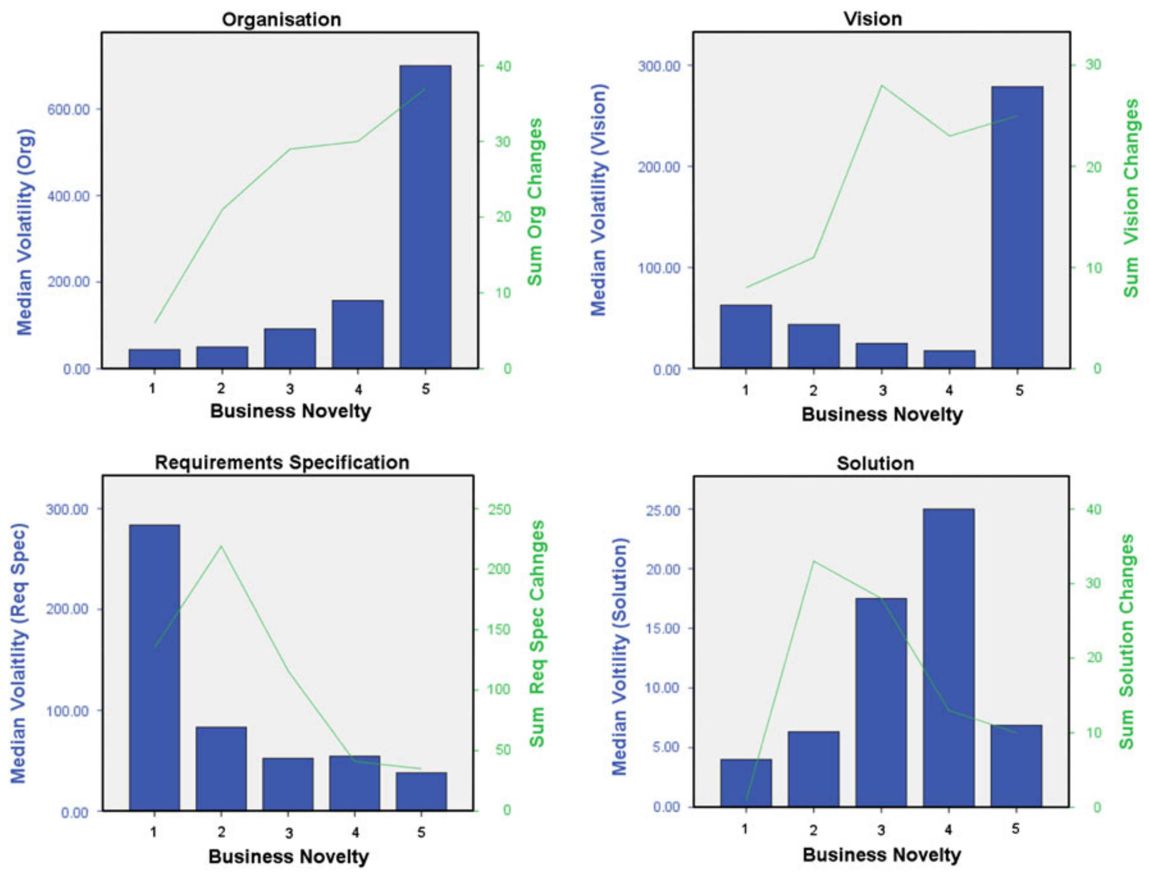


**Fig. 11** Median domain volatility for business novelty

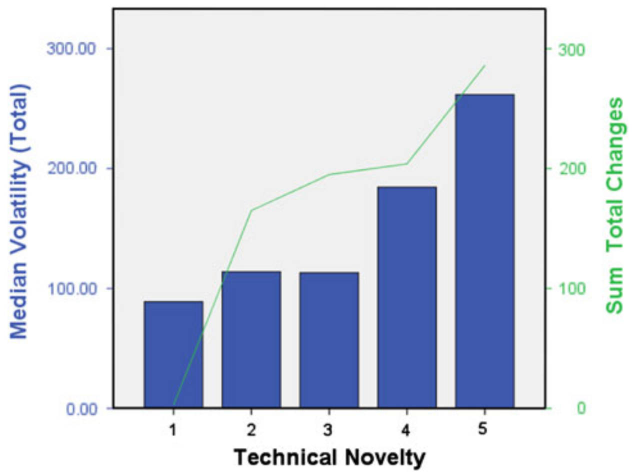


**Fig. 12** Median volatility for technical novelty for all changes

for the use of more agile techniques such as early prototyping or iterative development. Indeed, while it may be the case that agile techniques assuage late vision change, the observation that many *specification* changes were discovered during build and test may imply that the onus is upon analysis techniques as well as process procedures to reduce the types of changes that arise from *specification* issues.

Since a higher proportion of *organisation* and *vision* changes represent an opportunity to enhance previously agreed functionality as opposed to the correction of defects, the taxonomy also captures the notion that some change should be encouraged and some types of change avoided. Despite the concerning fact that this project increased in size by over 50 % due to requirement changes, over 70 % of these changes represented an opportunity to enhance previously agreed functionality rather than correct errors.

The results are visualised in Fig. 13. The arrow indicates the tendency for increasing cost, value, and opportunity change from the *solution* domain through the *specification*, *vision* to the *organisation* domain. At the same, the level of project management control is decreasing. While this study did not investigate the changes arising from the domain of *market*, it has been included here for completeness in lighter shading. There is no direct mapping between a requirement and an element in this taxonomy. A single requirement can be thought to comprise a slice consisting of elements of all 5 domains in differing proportions depending upon the developmental phase and position within the requirements hierarchy. Any requirement is

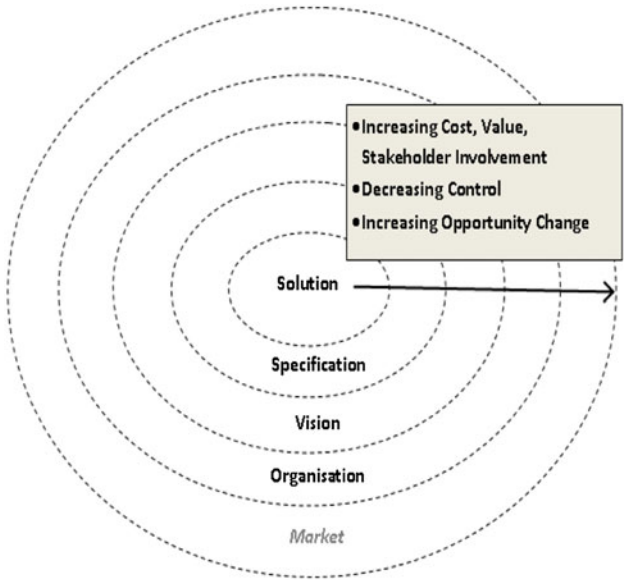

•Increasing Cost, Value, Stakeholder Involvement
•Decreasing Control
•Increasing Opportunity Change
Solution
Specification
Vision
Organisation
Market

**Fig. 13** Requirements change taxonomy

therefore subject to change arising from any change source domain.

The second case study set out to determine whether there are identifiable attributes of requirements that render them prone to change, and also investigated whether requirements exhibited the same pattern of volatility in the change domains evaluated in case study one. Representatives of our industrial partner identified these requirement attributes specifically because they expected that they would give rise to changes. However, apart from requirement dependency, which positively correlated with all measures of volatility, some of the results are counterintuitive. That business complexity bears no correlation with requirements changes raises the possibility of a confounding factor, and mirrors the observation by Fenton [29] that complexity alone has an arbitrary relationship to software defects. Most interesting, however, is the result that technical complexity is negatively correlated with volatility. Less technically complex requirements are changing with more frequency and cost with respect to their original cost estimation. Taken together, these results indicate that a prediction of changing requirements will not be achieved based solely upon the requirement attributes examined in this study. Further consideration of more complex causal factors, such as the process factors and analysis techniques that have been found to correlate with requirements volatility [8, 9], and also levels of effort and ability thought to influence the likelihood of defects [30].

The results of this case study also deepen our understanding of the distinction between the change domains contained within the requirements change source taxonomy. The implication from the results is that changes coming from sources of *organisation*, *vision*, *specification,* and *solution* are affecting different groups of requirements. Requirements with a higher level of business novelty change more often and with a higher change cost *only* from changes coming from the domain of *organisation*. We may infer that in addressing *organisation* change, there is increased certainty that novel requirements meet business needs, and as a result are less prone to other types of changes. This inference is supported by the contrasting observation that requirements with a lower level of business novelty are changing more frequently from *specification* changes. No such inference can be made about requirements with a higher level of technical complexity since there are no instances of a positive correlation with volatility. However, the only domain in which there was a negative correlation was *specification*, which, as we discovered in the first case study, has a higher instance of change from specification defects rather than opportunity. Fenton's argument [29] that more complex requirements may be afforded more resource, and therefore are likely to contain fewer (code) defects, is one possible explanation for these results.

Figure 14 illustrates a simplified cause and effect diagram that captures the salient implications of the results of these two case studies combined with conclusions drawn by other researchers. Clearly evident in the diagram is the two-sided causal nature of requirements change, which is reflective of the two types of constructs in the change source taxonomy (see the "Appendix"). These are 'uncertainties', which are situations that affect the disposition of a requirement to change, and 'triggers', which are events that promote change discovery. In the first case study, we attributed the event constructs to four activities that are contained here within the box labelled "Trigger". The curve over the arrows from the Change Triggers to the 'Changes Found' node indicates that only one trigger is required in order to discover change. As discovered in the second case study, apart from requirements dependency, no single attribute correlates with requirements volatility in all change domains. Previous studies have noted an influence of environmental factors, software development process and techniques, and effective communications upon requirements change [8, 9]. Therefore, all factors are causally relevant and must work in combination to affect requirements uncertainty, as illustrated in the box labelled "Uncertainty". In a fully developed causal model, each of the factors would consist of a number of attributes, or variables, and there may be requirements attributes in addition to dependency, novelty, and complexity that affect requirements uncertainty. Those such as effort and skill would affect both the uncertainty of a requirement and the likelihood of change discovery. Since a requirement continues to change subsequent to software delivery, there is an amount of uncertainty remaining even after changes

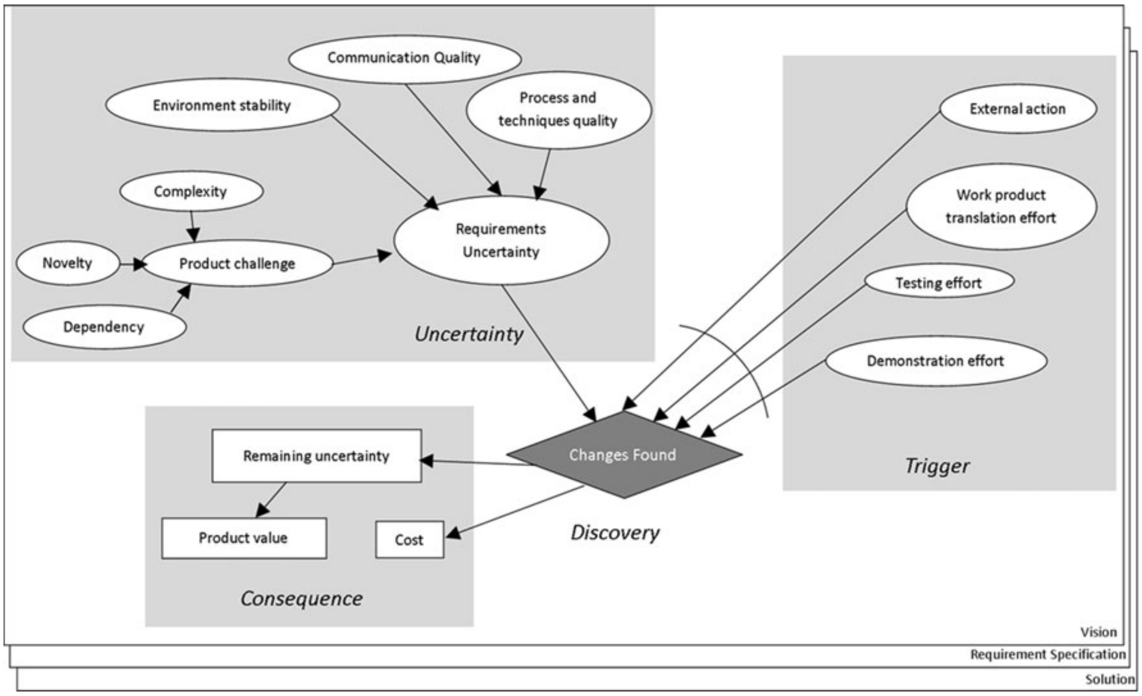


**Fig. 14** Layered causal account of software requirements change

have been discovered during development. This is included here, alongside cost and value, as a consequence of changes found.

As depicted in Fig. 13, the change domains within the taxonomy are characterised by not only by cost and value, but also by stakeholder involvement and change controllability. In addition, in case study two, we determined that there were different relationships between requirements attributes and volatility depending upon the change domain source. Further, environment stability attributes will differ according to change domain. For example, *solution* domain changes are not directly brought about by a change to customer's business process. Instead, this would result in project *vision* volatility. This distinction of character and variable influence is captured here by layering seen in Fig. 14.

## 6 Conclusions and further work

In an on-going empirical endeavour to better understand the phenomena of requirements change, two related case studies were undertaken in collaboration with our industrial partner. The first evaluated whether the requirements change source taxonomy could provide an informative means to measure change. The second examined attributes of the requirements for correlation with volatility and determined whether this pattern was replicated across the change domains investigated in the first study.

The software requirements change source taxonomy contains the change domains of *Market*, *Organisation*, *Vision*, *Requirements Specification*, and *Solution*. Informally, the question asked here is "How does this classification help me understand the consequences of change and why and when it is happening, so that I may be able to monitor and manage better". Researchers worked closely with an industrial partner to identify, collect, and validate suitable data to facilitate this investigation. While no results are available for the domain of *Market*, findings indicate the following:

- There are significant differences in cost, value, control, and stakeholder involvement between changes arising from each of the non-market sources.
- Generally, changes from the *organisation* domain are more costly, have a higher value, more often represent an opportunity rather than a defect, but also have increased stakeholder group involvement, considered less easy to control.
- From *Vision* to *Specification* to *Solution*, change costs fall, stakeholder involvement decreases, and there is an increased level of control.
- Changes coming from activity considered external to the project are all from the *Organisation* and *Vision* domains.

The implication is that the assessment of risk and management of changes should be tailored according to the characteristics of these change domains. As a means of monitoring and measurement, use of the requirements change taxonomy is feasibly practical and will aid understanding of software evolution during development as well as providing opportunities for retrospective project analysis to aid future process and technique tailoring.

To discern whether there were generic requirements qualities that lead to increased volatility, the changes from case study one were attributed to the requirements involved. Our industrial partner identified and collected three attributes that they felt were most influential to changing requirements—dependency, complexity, and novelty. The results can be summarised as follows:

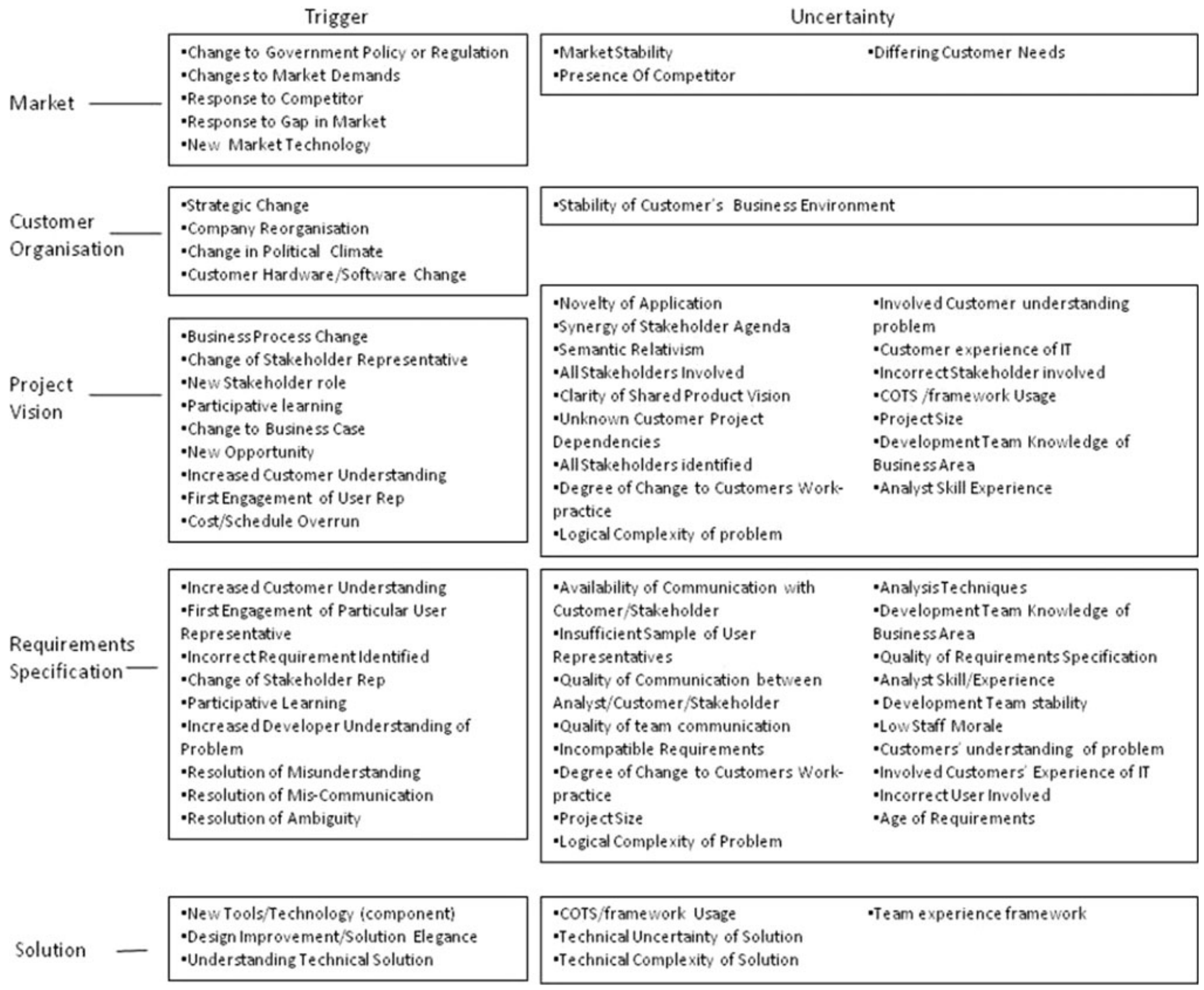
- Some requirements are more change-prone than others given that 80 % changes involved 31 % requirements.
- In the main, changes coming from the domains sources effected different groups of requirements.
- The only clear correlation with requirements volatility was dependence, which was positively correlated in all domains.
- Requirements business complexity showed no correlation with volatility, while technical complexity was negatively correlated but only in the domain of *Specification*.
- Requirement business novelty exhibited different relationships with volatility in each of the domains. Notable were the positive correlation between business novelty and volatility in the domain of *organisation* and the contrasting negative correlation in the domain of *requirements specification.*

Clearly, some requirements change more than others, and the results corroborate the distinction between categories in the taxonomy. However, although some influence can be observed, there is an implication that these attributes alone are not sufficient for predictive models. Instead, we have complex causal relationships involving attributes that influence the change—proneness of an individual requirement as well as factors related to project process and techniques that effect the software development as a whole. Importantly, we also must consider the effort expended upon the activities that promote change discovery.

These results have significant implications for the feasibility of change anticipation. Not only does requirements volatility arise in response to changes in the immediate ‘small world’ of the development environment, but more challengingly, the ‘larger world’ of the *organisation* and *market* whose needs must be met by the software. Changes discovered through ‘external action’ (16 % change cost in case study one) are going to be very difficult if not impossible to predict. However, our intention at this juncture is to build and test a set of predictive causal models, using a combination of expert judgment, previously published studies, and software development change data. These will be founded upon implications derived from the results of the case studies presented here. We will begin by concentrating upon the more controllable change domains of *solution, specification*, and *vision.*

**Acknowledgments** We would like to thank the project manager and analysts who gave their valuable time to data specification and the collection for the purposes of this investigation.

## Appendix: Software requirements change source taxonomy



| | Trigger | Uncertainty |
|---|---|---|
| Market | •Change to Government Policy or Regulation<br>•Changes to Market Demands<br>•Response to Competitor<br>•Response to Gap in Market<br>•New Market Technology | •Market Stability<br>•Presence Of Competitor<br>•Differing Customer Needs |
| Customer Organisation | •Strategic Change<br>•Company Reorganisation<br>•Change in Political Climate<br>•Customer Hardware/Software Change | •Stability of Customer's Business Environment |
| Project Vision | •Business Process Change<br>•Change of Stakeholder Representative<br>•New Stakeholder role<br>•Participative learning<br>•Change to Business Case<br>•New Opportunity<br>•Increased Customer Understanding<br>•First Engagement of User Rep<br>•Cost/Schedule Overrun | •Novelty of Application<br>•Synergy of Stakeholder Agenda<br>•Semantic Relativism<br>•All Stakeholders Involved<br>•Clarity of Shared Product Vision<br>•Unknown Customer Project Dependencies<br>•All Stakeholders identified<br>•Degree of Change to Customers Work-practice<br>•Logical Complexity of problem<br>•Involved Customer understanding problem<br>•Customer experience of IT<br>•Incorrect Stakeholder involved<br>•COTS /framework Usage<br>•Project Size<br>•Development Team Knowledge of Business Area<br>•Analyst Skill Experience |
| Requirements Specification | •Increased Customer Understanding<br>•First Engagement of Particular User Representative<br>•Incorrect Requirement Identified<br>•Change of Stakeholder Rep<br>•Participative Learning<br>•Increased Developer Understanding of Problem<br>•Resolution of Misunderstanding<br>•Resolution of Mis-Communication<br>•Resolution of Ambiguity | •Availability of Communication with Customer/Stakeholder<br>•Insufficient Sample of User Representatives<br>•Quality of Communication between Analyst/Customer/Stakeholder<br>•Quality of team communication<br>•Incompatible Requirements<br>•Degree of Change to Customers Work-practice<br>•Project Size<br>•Logical Complexity of Problem<br>•Analysis Techniques<br>•Development Team Knowledge of Business Area<br>•Quality of Requirements Specification<br>•Analyst Skill/Experience<br>• Development Team stability<br>•Low Staff Morale<br>•Customers' understanding of problem<br>•Involved Customers' Experience of IT<br>•Incorrect User Involved<br>•Age of Requirements |
| Solution | •New Tools/Technology (component)<br>•Design Improvement/Solution Elegance<br>•Understanding Technical Solution | •COTS/framework Usage<br>•Technical Uncertainty of Solution<br>•Technical Complexity of Solution<br>•Team experience framework |

## References

1. Loconsole A, Borstler J (2005) An industrial case study on requirements volatility measures, In: Proceedings 12th Asia-Pacific software engineering conference (APSEC'05), pp 249–256
2. Costello R, Liu D (1995) Metrics for requirements engineering. J Syst Softw 29(1):39–63
3. Boehm B, Turner R (2004) Balancing agility and discipline: evaluating and integrating agile and plan-driven methods, In: Proceedings 26th international conference on software engineering, pp 718–719
4. Jones C (1996) Strategies for managing requirements creep. Computer 29(6):92–94
5. Kulk GP, Verhoef C (2008) Quantifying requirements volatility effects. Sci Comput Program 72(3):136–175
6. McGee S, Greer D (2009) A software requirements change source taxonomy. In: Proceedings 4th international conference on software engineering advances, pp 51–58
7. Nuseibeh B (2001) Weaving together requirements and architectures. IEEE Comput 34(3):115–117
8. Ferreira S, Shunk D, Collofello J, Mackulak G, Dueck A (2011) Reducing the risk of requirements volatility: findings from an empirical survey. J Softw Maint Evol Res Pract 23(5):375–393
9. Zowghi D, Nurmuliani N (2002) A study of the impact of requirements volatility on software project performance. In: Proceedings 9th Asia-Pacific software engineering conference, pp 3–11
10. McGee S, Greer D (2011) Software requirements change taxonomy: evaluation by case study. In: Proceedings 19th IEEE international requirements engineering conference, pp 25–34
11. Benestad H, Anda B, Arisholm E (2009) Understanding software maintenance and evolution by analyzing individual changes: a literature review. J Softw Maint Evol Res Pract 21(6):349–378
12. Swanson E (1976) The dimensions of maintenance. In: Proceedings 2nd international conference software engineering, pp 492–497
13. Chapin N, Hale J, Khan K, Ramil J, Tan W (2001) Types of software evolution and software maintenance. J Softw Maint Evol Res Pract 13(1):3–30

14. Kemerer C, Slaughter S (1999) An empirical approach to studying software evolution. IEEE Trans Softw Eng 25(4):493–509
15. Stark G, Skillicorn A, Ameele R (1999) An examination of the effects of requirements changes on software maintenance releases. J Softw Maint Res Pract 11(5):293–309
16. Xing Z, Stroulia E (2004) Understanding class evolution in object-oriented software. In: Proceedings 12th IEEE international workshop on program comprehension, pp 34–43
17. Heales J (2000) Factors affecting information system volatility. In: Proceedings 21st international conference information systems, Brisbane, Australia, pp 70–83
18. Harker SDP, Eason KD, Dobson JE (1993) The change and evolution of requirements as a challenge to the practice of software engineering, In: Proceedings IEEE international symposium on requirements engineering, pp 266–272
19. Sommerville I (2010) Software engineering, 9th edn. Addison Wesley, Reading
20. Nurmuliani N, Zowghi D, Powell S (2004) Analysis of requirements volatility during software development life cycle, In: Proceedings Australian software engineering conference, pp 28–37
21. Nurmuliani N, Zowghi D, Williams SP (2004) Using card sorting technique to classify requirements change. In: Proceedings 12th IEEE international conference on requirements engineering, pp 240–248
22. Nurmuliani N, Zowghi D, Williams SP (2006) Requirements volatility and its impact on change effort: evidence-based research in software development projects. In: Proceeding 11th Australian workshop on requirements engineering. http://awre2006.cis.unisa.edu.au/proceedings/paper%207%20Nurmuliani.pdf
23. McGee S, Greer D (2010) Sources of software requirements change from the perspectives of development and maintenance. Int J Adv Softw 118–200
24. Perry D (1994) Dimensions of software evolution. In: Proceedings of international conference on software maintenance, pp 296–303
25. Weiss DM, Basili VR (1985) Evaluating software development by analysis of changes—some data from the software engineering laboratory. IEEE Trans Software Eng 11(2):157–168
26. Nakatani T, Hori S, Tsuda M, Inoki M, Katamine K, Hashimoto M (2009) A proprosal for the prince model. In: Proceedings of international conference software and data technologies, pp 145–150
27. Card DN (2006) Myths and strategies of defect causal analysis. In: Proceedings of the Pacific northwest software quality conference
28. Höfer A, Tichy WF (2007) Status of empirical research in software engineering. In: Basili et al. (eds) Experimental software engineering issues: assessment and future directions, Springer, Berlin, pp 10–19
29. Fenton NE, Neil M (1999) A critique of software defect prediction models. IEEE Trans Software Eng 25(5):675–689
30. Fenton N, Neil M, Marsh W, Hearty P, Radliński L, Krause P (2007) Project data incorporating qualitative factors for improved software defect prediction. In: Proceedings of the 3rd international workshop on predictor models in software engineering
31. Perry DE, Porter A, Votta L (2000) Empirical studies of software engineering: a roadmap. In: Proceedings ICSE '00: proceedings of conference on the future of software engineering, pp 345–355
32. Runeson P, Host M (2009) Guidelines for conducting and reporting case study research in software engineering. Emp Softw Eng 14(2):131–164
33. Wohlin C, Host M, Henningsson K (2003) Empirical research methods in software engineering. Lect Notes Comput Sci 2765:145–165
34. Yin RK (2003) Case study research. Design and methods, 3rd edn. Sage, London
35. Basili VR (1985) Quantitative evaluation of software methodology, University of Maryland, TR-1519
36. Fenton N, Neil M (2000) Software metrics: roadmap. In: Proceedings of conference on future of software engineering, pp 359–370
37. Pfleeger SL (2008) Software metrics: progress after 25 years? IEEE Softw 25(6):32–34
38. Barry EJ, Kemerer CF, Slaughter SA (2006) Environmental volatility, development decisions, and software volatility: a longitudinal analysis. Manage Sci 52(3):448–464
39. Field A (2009) Discovering statistics using SSPS, 3rd edn. Sage Publications ltd, London
40. Sjoberg DIK, Dyba T, Jorgensen M (2007) The future of empirical methods in software engineering research. In: Proceedings future of software engineering, pp 358–378
41. Boehm B, Basili VR (2005) Software defect reduction top 10 list. In: Boehm B, Rombach HD, Zelkowitz MV (eds) Foundations of empirical software engineering. Springer, Berlin, pp 426–431