Application of Hazard Analysis to Quality Modelling

Hong Zhu, Yanlong Zhang, Qingning Huo and Sue Greenwood
School of Computing and Mathematical Sciences
Oxford Brookes University, Wheatley, Oxford, OX33 1HX, UK
Emails: (hzhu | yzhang | qhuo | sgreenwood) @brookes.ac.uk

Abstract

Quality is a fundamental concept in software and information system development, but it is also a complex and elusive concept. A large number of quality models have been developed for understanding, measuring and predicting qualities of software and information systems. The rapid expansion of the application areas of computer technology in recent years with the growth of the Internet and WWW has imposed great challenges to the existing theories and techniques of quality management in the development of information systems. It has been recognised that quality models should be constructed in accordance to the specific features of the application domain. This paper proposes a systematic method for constructing quality models of information systems. A diagrammatic notation is devised to represent quality models that enclose application specific features. Techniques of hazard analysis for the development and deployment of safety related systems are adapted for deriving quality models from system architectural designs. The method is illustrated by a part of business-to-business e-commerce information systems.

Keywords: Information systems, Quality modelling, Construction of quality models, Business to business e-commerce, Hazard analysis.

1. Introduction

Software quality is a complex and elusive concept [1]. There are dozens, even hundreds, of attributes of software systems that are associated with the quality of software and information systems. In the past few decades, researchers have built models to understand, measure and predict the quality of software and information systems [2]. These models can play a significant role in software and information system development. In particular, a quality model can provide general guidelines for the elicitation of users' requirements on software and information systems' quality. It can provide the insight information for software designers to seek technical solutions to achieve required quality. It is the foundation for organising quality assurance activities. For example, it can provide detailed information about a system's sensitivity to a quality issue so that adequate testing can be targeted directly to the issue. It is the foundation for the development and deployment of software quality evaluation methods and quantitative measurement metrics. Recently, in accordance with the rapid growth of the Internet and the World Wide Web, a wide range of information systems in new application areas have emerged, such as e-commerce, e-learning, e-science, and so on. Since web-based information systems are distributed, multimedia, co-operative, and even intelligent, they have imposed great challenges to current theories and methods of software quality management [3]. In [4], the differences between web-based information systems and conventional information systems are discussed from the perspective of software quality. It is realised that new quality models for web-based information systems are necessary. The theme of this paper is how to construct quality models for such information systems.

The paper is organised as follows. Section 2 reviews the related work and discusses the open problems in quality modelling. Section 3 proposes a diagrammatic notation for the representation of quality models. Section 4 adapts a hazard analysis method for the derivation of quality models from system architectural designs. Section 5 gives an example to illustrate the method. Section 6 is the conclusion of the paper.

2. Related work

Existing research work in the literature falls into two categories: quality models and methods for
constructing quality models. The following briefly reviews each of the categories and outlines our previous work.

2.1 Existing quality models

The past several decades has seen a great amount of work on quality models of software and information systems. Among the best known are hierarchical models, such as the McCall model [5], the Boehm model [6], and the quality model of ISO 9126 [7]. A hierarchical quality model divides the abstract concept of quality into a number of quality factors. Each quality factors is associated with a number of quality criteria that contribute to the quality factor. Each quality criterion is further associated with a number of quality measures. Such models provide the information about the positive influences between quality attributes. However, they failed to represent more complicated relationships between quality attributes. Relational models, such as the Perry model [8] and the Gillies model [2, 9], characterise the relationships between quality attributes by a number of stereotypes of relations, such positive, negative and neutral impacts of one attribute on another. Quantitative models of software quality usually appear in the form of software quality metrics that measurement of a software system's quality on an attribute is calculated from measurements of other attributes; see, for example, [10, 11].

Regarding information systems as a much wider concept than software systems, quality models for information systems have also been proposed and investigated in the literature. For example, the SOLE model [12] and its variants are hierarchical models. They organise the hierarchic structure according to three different views from three different groups of stakeholders: users, technical staffs and managers. This approach to quality modelling is consistent with the general theory of quality and quality management proposed by Garvin [13]. According to the theory, quality is multifaceted and highly context dependent because it means different things to different people. It characterises the quality of a product by five different perspectives according to the viewer's relationship with the product. Recently, quality models for web-based applications have also been investigated. In [14], a quality model for websites of universities, called Website QEM, was proposed based on the users' view. It breaks down the quality of websites into more than a hundred attributes. A great number of guidelines for web design have also been published [15, 16, 17, 18]. However, no common accepted standards have been set for defining and assessing the web site quality.

Existing quality models were intended to be comprehensive and applicable to all software development. However, as pointed out in [1], there can be no single and simple measure of software quality acceptable to everybody. It has also recognised that every software system may have its own quality concerns [19]. Special requirements of the software or the information system must be considered in the application of quality models [19, 20]. In particular, with the ever-growing range of information systems that are developed with different functions using various software and hardware techniques, software engineers are seeking for quality models that can provide useful insight information not only for quality management, but also for other development activities. How to develop such quality models for an information system remains an open problem.

2.2 Methods for constructing quality models

Existing quality models are mostly constructed based on many years' experience in the development and maintenance of software and information systems. The validation of such models is mostly by empirical studies, such as by analysing data collected from questionnaires and interviews [8]. In [21, 22], Dromey pointed out that the difficulty of quality modelling is due to the perceived scale of the problem, the diversity of quality defects in software, and the difficulty of factoring high-level quality attributes down to tangible properties. A systematic method is necessary to construct testable, assessable, and refineable quality models for different software products and key products of software development. Dromey proposed a generic quality model and a process to systematically develop software quality models for different software products. The generic quality model consists of three principal elements: product properties that influence quality, a set of high-level quality attributes, and a means of linking them. The generic model is instantiated and refined for a particular software product through model construction, which consists of the following five steps.
• Identify a set of high-level quality attributes for the product.
• Identify the product components.
• Identify and classify the most significant, tangible, quality-carrying properties for each component.
• Propose the set of axioms for linking product properties to quality attributes.
• Evaluate the model, identify its weakness, and either refine it or scrap it and start again.

This process seems straightforward, but still relies heavily on the empirical knowledge of software systems. Recently, Bansiya and Davis [23] applied the method to build a hierarchical model of object-oriented design quality. Although the links between the lower-level and higher-level attributes were established and evaluated based on statistical data, the identification of quality attributes at various levels is still based on empirical knowledge.

Dromey demonstrated the application of the method to software requirements definition, design and implementation. Although Dromey correctly recognised that a quality model must be built through quality-carrying properties of the components of the software product, the applications of the method to requirements definition, design and implementation only produced universal quality models for such software artefacts. The specific features of the application area and system design and implementation were not considered in the quality models. The card sort method proposed in [24] provided a partial solution to the problem.

The card sort method [24], originated by Kelly’s Personal Construct Theory [25], was applied to elicit the quality attributes of web-based applications. The method starts with the selection of a number of pictures of user interfaces, such as the downloaded screens of web pages, as the entity to sort. These pictures were shown to different groups of people, such as farmers, web designers and laypeople. Respondents were required to describe the quality attributes using subjective or objective criteria. The responses from the subjects were collected, and then, processed. There are several ways to process the result. One way is to sort these criteria and count the number of categories within each criterion. The results from the card sort method are promising as the technique is easy to use, and the technique is suitable for the investigation of the visual natural of web-based applications. A shortcoming of the approach is that it can only be applied after the completion of the development of the web site.

2.3 Our previous work

In [26, 27], we proposed a method to systematically derive quality models from architectural designs of information systems. It adapted hazard analysis methods, which are widely used in the development of safety critical systems, to enable software engineers to systematically identify certain types of quality attributes and the quality carrying properties of each component and connector, and to establish the links between them. Hazard analysis provides the context of the links and forms the logic that provides a bridge between the gap from components' quality carrying properties to system's quality attributes. Case studies of the methods have been carried out for business to customer and business to business e-commerce systems. These case studies have demonstrated the applicability and a number of advantages of the quality modelling method. First, it enables the practitioners and researchers to develop their own quality models for individual systems. Second, as many applications in a specific application domain often share a common architectural structure, the method can also provide a quality model for those systems of the same architecture and in the same application domain. Moreover, the method is applicable at a relatively early stage of system development process. Finally, the method can provide more insight information of the system than existing quality models and quality modelling methods. Such information can be directly used to help allocating development resources and managing quality assurance and other development activities. In particular, it is useful to identify the risk of each component and functionality so that design solutions can be put forward to resolve risks and to improve quality. Moreover, test cases can be derived directly targeting to the quality sensitive functions and components.

This paper further develops the method by proposing a new representation of quality models and a process for the derivation of such quality models.
3. Representation of quality models

Existing software quality models have been represented using simple and intuitive notations. For example, hierarchical models are represented in the form of a tree graph with nodes as quality attributes and arcs as positive relations between the attributes. Relational models use matrices that each row and column represents a quality attribute and the values of the matrix on a row and a column represents the stereotype relations between the corresponding attributes. Such representations do not refer to any elements of the software product whose quality is under investigation. Hence, they are independent of the software product. These representations are suitable for universal quality models that are intended to be applicable to all software systems. Although such models do play significant roles in software and information system development, quality models that make no explicit references to the product specific features have limited usefulness. We believe in Dromey’s principle that abstract quality attributes must be linked to the tangible software properties through the quality-carrying properties of each component. However, in Dromey’s method, the model of a software product is only used as a tool. The result quality models make no explicit reference to the components of the product. Here, we argue that how a quality-carrying property of a component is related to a quality attribute of the system is important because it provides the sort of insight information that can significantly improve the usability of quality models.

For example, safety is an important quality attribute of safety critical systems. It is of extreme importance for engineers to understand how faults and failures of the components are related to the safety of the system. Only when such information is available, can design solutions be put forward to eliminate the specific types of faults of the component and to prevent the occurrences of the specific types of failure modes that may contribute to safety. Moreover, testing of the system can then directly target the safety-related components and events to ensure system safety.

A quality attributes / quality-carrying properties of a component, such as usability and maintainability, are usually abstract. Consequently, the links between two abstract properties cannot be easily established or validated. However, abstract properties usually demonstrate themselves through various concrete events and observable phenomena, which are tangible and observable. For example, the poor usability of a web page is clearly demonstrated if the user cannot find the required information through the hyperlinks. While relationships between abstract properties are difficult to establish and validate, the relationships between observable phenomena are often self-evidence in the context of the system. For example, when the incorrectness of an HTML file of a Web page is demonstrated by the fact of containing a large number of broken links, the usability of the system will be poor because the user would not be able to find the information through the hyperlinks. This example shows that if the observation of one phenomenon implies the occurrence of another phenomenon, the corresponding abstract properties must have an implication relationship. Many authors have used such rationale in the construction of quality models. Unfortunately, such rationale has never been included in existing quality models. We believe that such information is useful. For example, in the design and analysis of safety critical systems, we not only need to know if the system is safe, we also need to know how the system will behave if certain event happens. This provides the crucial information for software testers to develop test cases to check if the system correctly implements the safety as designed.

In summary, we identify the following requirements on the representation of quality models.

Requirements 1: A quality model should explicitly associate quality attributes / quality carrying properties to the components of the system.

Requirements 2: A quality model should associate abstract properties with observable and verifiable phenomena of the components / system.

Requirements 3: A quality model should present the rationale of the relationships between the properties. Such rationales can be system specific and should be able to be verified and validated in the context of the system.

Therefore, as shown in Figure 1, our proposed diagrammatic representation of quality models is directed
graphs, which consists of two principal elements: the nodes and links. Each node contains three basic elements: (1) the component of the system; (2) the quality-carrying properties of the component; and (3) the observable phenomena of the property. The links are directed arcs between the nodes. A link from node A to node B means that the observation of the phenomenon on node A implies the occurrence of the phenomenon on node B. Each link can contain an optional annotation for the reasons why the two nodes are related.

![Proposed notation for representation of quality models](image)

Figure 1 Proposed notation for representation of quality models

Figure 2 gives a fragment of quality model of Web-based information systems. This fragment of quality model only shows that the usability of a web-based system is related to the correctness, responsiveness, structuredness of HTML files, compatibility, and the usability of the online help subsystem. It also indicates in detail how these properties are related to whether the user can find the required information. For example, if a file is of large size, it will increase the response time. If the response time is longer than the time-out setting, the browser will regard the requested file as unavailable. It also shows that the compatibility of the code on the client side will affect the usability, while the compatibility of the server side does not. The designer can use this quality model to ensure that each page is in a reasonable size to that short response time can be achieved. The testers can use this model to check if there are any broken links in the Web pages, and so. It should be noticed that, the links between the nodes must be understood as the implications of one phenomenon to another, rather than simply the relationship between two quality attributes. For example, large sized HTML files may contain less hyperlinks between them than smaller sized files. This makes the navigation between the files easier. Consequently, the user may find it is easier to use. Therefore, it is positively related to the usability of the system. On the hand, large sized HTML files will increase the response time and in extreme cases may cause poor usability. Such complexity cannot be represented in a quality model that only relates two abstract quality attributes.

![A fragment of quality model of Web-based information systems](image)

Figure 2 A fragment of quality model of Web-based information systems
4. Derivation of quality models

Our method for the construction of quality models takes structural models of information systems as input and applies hazard analysis methods to derive the quality sensitive observable phenomena of the behaviour of the components or the system and the causal relationships between the phenomena. The quality carrying property / quality attribute that a phenomenon demonstrates is then identified according to the nature of the phenomenon. These elements are then assembled together and represented in the diagrammatic notation given above.

4.1 Adaptation of Hazard analysis

Originally, hazard analysis intends systematically identifying, assessing and controlling hazards before a new work process, piece of equipment, or other activity is initiated. Hazard analysis techniques have been widely used in the development and deployment of safety critical systems that involve computer software or not. In such a context, a hazard is a situation in which there is actual or potential danger to people or to the environment. Associated with each hazard is a risk, which is related to the likelihood of the event occurring and its consequences. Once the hazards are identified and analysed, safety as well as other quality requirements can be specified for each component. Risks can be avoided or reduced ultimately through technical design, management and organisational means. Consequently, the quality and reliability of the system are improved [28, 29, 30, 31].

In [26, 27], we adapted the methods of hazard analysis and extended the concept of hazard to construct quality models of information systems. In our context, the word hazard has its widest meaning, which means any situation that may cause harm. The more likely a hazard occurs and more serious the consequences of the hazard, the more important the corresponding quality attribute, and vice versa.

There are a number of hazard analysis techniques available in the literature. We are particularly interested in the FMEA (failure modes and effects analysis) technique. FMEA progressively selects the individual components or functions within a system and investigates their possible modes of failure. It then considers possible causes for each failure mode and assesses their likely consequences. In the original FMEA, the effects of the failure are determined for the unit itself and for the complete system. Possible remedial actions are also suggested. A simple example of FMEA is given in Figure 3.

<table>
<thead>
<tr>
<th>Ref No.</th>
<th>Unit</th>
<th>Failure mode</th>
<th>Possible cause</th>
<th>Local effects</th>
<th>System effects</th>
<th>Remedial action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tool guard switch</td>
<td>Open-circuit</td>
<td>Faulty component</td>
<td>Failure to detect tool guard in place</td>
<td>Prevents use of machine - system fails safe</td>
<td>Select switch for high reliability and low probability of dangerous failure Rigid quality control on switch procurement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>contacts</td>
<td>Excessive current</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Extreme temperature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Short-circuit contact</td>
<td>Faulty component</td>
<td>System incorrectly senses guard to be closed</td>
<td>Allows machine to be used when guard is absent - dangerous failure</td>
<td>Modify software to detect switch failure and take appropriate action</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Excessive current</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Excessive switch-bounce</td>
<td>Ageing effects</td>
<td>Slight delay in sensing state of guard</td>
<td>Negligible</td>
<td>Ensure hardware design prevents excessive current through switch</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Prolonged high currents</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3 An example of FMEA chart [31]
FMEA enables us to identify a system’s potential failure modes, their possible causes and the consequences. Each cause of a failure indicates what quality attribute that the system is sensitive from developer’s point of view. The corresponding consequences of the failure indicate what quality attributes the system is sensitive from the users’ point of view. Both causes and their consequences are observable phenomena of the system. Therefore, the relationships between the quality attributes or quality-carrying properties can be established. However, the original FMEA chart is ambiguous about which component causes the failure. As discussed in the previous sub-section, which component causes the failure is important for quality models of information systems. Therefore, to adapt FMEA for analysing software quality, we modify the FMEA chart to the following format so that the component that causes a failure becomes clear. Another modification to FMEA is that the effects of a failure mode are not charted. There are two reasons for this. First, we found that for a complicated software system, the indirect effects such as those at system level may not be so clear when a component fails. Second, because indirect effects will be analysed subsequently as the effect of other failures, the system level effects of a component failure will eventual emerge from such a chain of cause-effect. The direct effects of a failure mode should have been charted if the direct causes of all failure modes are charted because the 'effect' relation is the inverse of the relation of 'cause'. Finally, we also included an explanation column in the SFMEA chart so that the reasons why a failure mode is caused by another can be given.

For example, Figure 4 shows a failure mode that the user cannot find the required information on the Web page. A cause of the failure mode is charted as that the Web page is unable to obtain a file through a hyperlink. Two further causes of the failure are charted: (1) the hyperlink is broken; (2) the Web server is down.

<table>
<thead>
<tr>
<th>Ref. No.</th>
<th>Failure mode</th>
<th>Possible cause</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Component Phenomena</td>
<td>Component Fault/failure mode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>The user Cannot find required information</td>
<td>Web page Unable to obtain a file through a hyperlink</td>
<td>When the user searches for information by browsing through hyperlinks</td>
</tr>
<tr>
<td>2</td>
<td>Web page Unable to obtain a file through a hyperlink</td>
<td>HTML files The link is broken</td>
<td>The file cannot be found due to the broken of the link.</td>
</tr>
<tr>
<td>3</td>
<td>Web server Server is down</td>
<td></td>
<td>The file cannot be retrieved and transmitted.</td>
</tr>
</tbody>
</table>

Figure 4 The format of Software FMEA chart

In the application of SFMEA, each of the causes and consequences of a failure mode become a new entry to the chart. These causes and consequences are further investigated until the cause is primitive and the consequences are terminal. A failure mode is primitive if it is caused by a fault of a component and its causes cannot be further identified without additional knowledge about the system. A failure mode is terminal if it does not effect any other component of the system or does not cause any other failures. In the example shown in Figure 4, the consequence of 'user cannot find required information' can be considered as terminal. The failure 'broken link' can be considered as primitive. The failure mode that the server is down is neither terminal nor primitive. It should be further investigated for its causes, which might be hackers' attack, maintenance shutdown, system crash due to software failure, etc. The consequences of the

---

1 Notice that, there are more causes of the failure than what have been charted in the example.
failure also need further investigation, which may be more than just that the user cannot find the information. In next section, we will see how the adapted FMEA can be used to analyse information systems.

4.2 Construction of quality model

In our method, the construction of a quality model takes the information charted in the SFMEA. Each failure mode in the chart forms a node with the component and phenomenon as specified in the SFMEA chart. Each row in the chart forms a link from the node that represents the cause to the node that represents the failure mode. The explanation column of the row forms the reason of the link. For example, for the first row in Figure 4, the following nodes and link are generated.

![Figure 5: The fragment of diagram derived from row 1 of Figure 4.](image)

Similarly, from the second and third rows in Figure 4, we can derive the following nodes and links.

![Figure 6: The fragment of diagram derived from row 2 and 3 of Figure 4.](image)

Assembling these nodes and links together, we can obtain the following diagram.

![Figure 7: The diagram derived from row 1 and 2 of Figure 4.](image)

However, the diagrams generated from SFMEA charts as above are incomplete. The property slots need to be filled. Therefore, for each node in the diagram, the observable phenomenon is compared with the definitions of a set of quality attributes and quality-carrying properties of the components. The quality attribute or quality carrying property that the phenomenon demonstrates is, then, identified, or a new attribute or property is recognised. This property is filled into the slot of each node. For example, 'a hyperlink is broken' demonstrates the quality attribute correctness. 'Server is down' is related to the reliability of the system. 'User cannot find required information' is associated to the usability of the system. Therefore, we can derive the following quality model from Figure 4.
5. Example: B2B e-commerce systems

This section applies the method proposed above to a part of business to business e-commerce systems to illustrate the use of the method.

5.1 Architectural description of B2B e-commerce systems

Business to business (B2B) e-commerce systems are different from the business to customer (B2C) or customer to customer (C2C) e-commerce systems, as they have a number of special characteristics including high volume of goods traded, high value of goods traded, multiple electronic payment methods, and involvement of business bidding, contracts and agreements [32]. These features make the B2B systems more complex to manage than B2C or C2C systems. In this section, we develop a quality model of B2C e-business systems to illustrate the method proposed in the previous section.

At the highest level of abstraction, a B2B e-commerce system consists of three parties, the buy-side, the virtual market and the sell-side. Information is exchanged among these parties in certain orders. A cycle of activities must be completed before a deal can be made. The flow of information is depicted in the UML sequence diagram of Figure 9, which is adapted from [32].

![Figure 9 Buy-side B2B systems information flow diagram](image)

The information exchange process within the systems can be decomposed into four phases: information searching, purchasing requisition, signing contract, and receiving goods and make payment. Therefore, the
The basic architectural structure of the virtual market part of an e-commerce system consists of four components: product information manager, purchasing requisition processor, contract manager and payment manager.

5.2 Analysis of hazards

Now, we apply the software hazard analysis method SFMEA by investigating the components and connectors of the system as described in Figure 10. For the sake of space, we only give the details of some failure modes of the component Product Information Manager.

In the information searching phase, a user, i.e., a buyer, tries to find the information of required products. The user may use search engine, hyperlinks, and other means provided by the component Product Information Manager to find as much useful information as possible. However, problems or failures are common with such information search methods. We can identify three types of failures of Product Information Manager: (A) No information found; (B) The information found is irrelevant, including the situation that too much is presented but only a little is relevant; (C) Information is inaccurate. For each type of failure, there may have one or more possible causes of the failure. For example, suppose that two information searching facilities are provided by the component Product Information Manager: browsing through hyperlinks, query through keywords. The following can be the causes of 'no information found'.

1. **Fail to browse**: The user cannot find required information by browsing through the hyperlinks.

2. **Fail to make queries**: The user cannot find required information by querying using keywords.

For the first failure mode 'fail to browse', we identified the following possible causes.

(a) **Broken links**: The HTML files of the Web pages that belong to this component contains some broken hyperlinks, i.e. the file that a hyperlink points to does not exist.

(b) **Excessive response time**: If the requested information cannot be transmitted to the buy-side to display
on the screen, the information will be regarded as unavailable even if it exists and is actually transmitted. An error message ‘Page not found’ will be displayed when the response takes a time longer than a certain value of time-out set by the user.

(c) Difficulty in use of the facilities: The end-user does not know which path of hyperlinks to follow, for example, the end-user gets lost in the network of hyperlinks, etc.

(d) Incompatible platforms: The platform or system that the user is using does not support the buy-side of the application or to display the information provided by the system.

For each of the failure mode, the causes and consequences are then identified. For example, the following causes of broken links can be identified.

- The software contains faults;
- The server is down;
- The file that the hyperlink points to has been moved, but the link in the HTML file has not been updated timely, etc.

The causes of time-out can be any of the following.

- Heavy traffic on the network;
- Excessive load on the server;
- Large volume of information transmitted, etc.

The causes of difficulty in the use of the facilities can be any of the following.

- Poor interface design, especially the hyperlinks are not indicative or informative;
- Unavailability of online help, such as navigation maps;
- The user is lack of background knowledge, etc.

The failure mode of incompatible platforms is a primitive failure, whose causes cannot be identified without further knowledge about the system such as how it is implemented and what is the platform that the user is using.

Similar analysis is applied to the failure mode 'failed to make queries'. The result of the analysis is then charted in the form shown in Table 1. Filling the form drives the analyst to think of the possible failure modes, their causes and consequences. While the completed form itself forms a detailed quality model, diagrammatic representation of quality models can be derived from the form by further identification of the quality attributes or quality-carrying properties of each failure mode.

<table>
<thead>
<tr>
<th>Ref No.</th>
<th>Failure mode</th>
<th>Possible cause</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Product Information Manager</td>
<td>Failed to provide any information</td>
<td>When user browses the Web page</td>
</tr>
<tr>
<td>2</td>
<td>Search Engine</td>
<td>Failed to provide any information</td>
<td>When user searches information by making queries</td>
</tr>
</tbody>
</table>

Table 1. A Part of SFMEA chart for B2B e-commerce systems
<table>
<thead>
<tr>
<th></th>
<th>HTML files</th>
<th>Failed to provide any information</th>
<th>HTML files</th>
<th>Broken links</th>
<th>The file cannot be located</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Product Information Manager</td>
<td>Excessive response time</td>
<td>The file will be considered as unavailable when response time exceeds time-out setting.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Web Server</td>
<td>The server is down</td>
<td>The files cannot be retrieved and transmitted.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Buy-side</td>
<td>Incompatible platform</td>
<td>The web page cannot be displayed on user's screen.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>HTML files</td>
<td>Difficult to use</td>
<td>HTML files</td>
<td>The hyperlinks are not indicative or informative</td>
<td>The user does not know which hyperlink to click on.</td>
</tr>
<tr>
<td>8</td>
<td>Online help</td>
<td>Unable to provide useful helps</td>
<td>The user cannot find help to overcome the difficulty.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>The user</td>
<td>Lack of background knowledge</td>
<td>E.g. no navigation maps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Search engine</td>
<td>Failed to provide any information</td>
<td>Buy-side</td>
<td>Query is not submitted to the virtual machine</td>
<td>Search engine does not know the request</td>
</tr>
<tr>
<td>11</td>
<td>Database server</td>
<td>The server is down</td>
<td>Cannot search for and transmit the information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Database</td>
<td>No information found</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Product Information Manager</td>
<td>Excessive response time</td>
<td>When response time exceeds the time-out setting.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Buy-side</td>
<td>Query is not submitted to the virtual machine</td>
<td>Buy-side</td>
<td>Incompatible platform</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Buy-side</td>
<td>Software failure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Network</td>
<td>Broken connection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Database</td>
<td>No information found</td>
<td>User</td>
<td>Difficult to make correct query</td>
<td>Query must be in certain format</td>
</tr>
<tr>
<td>18</td>
<td>Database</td>
<td>No information stored</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>User</td>
<td>Difficult to make correct query</td>
<td>User</td>
<td>Lack of background knowledge</td>
<td>Does not know the format</td>
</tr>
<tr>
<td>20</td>
<td>HTML files</td>
<td>The interface is not informative</td>
<td>No prompt for making query</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Online help</td>
<td>Unable to provide useful helps</td>
<td>No adequate help information about the format of queries</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Product Information Manager</td>
<td>Excessive response time</td>
<td>Network</td>
<td>Heavy traffic w.r.t. capacity</td>
<td>Transmissions are queued.</td>
</tr>
<tr>
<td>23</td>
<td>HTML files</td>
<td>Large size</td>
<td>Large volume of data transmitted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Web Server</td>
<td>Heavy load</td>
<td>Requests are queued.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>HTML files</td>
<td>Small number of nodes and links</td>
<td>HTML files</td>
<td>Large size</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>HTML files</td>
<td>Easy to navigate</td>
<td>HTML files</td>
<td>Small number of nodes and links</td>
<td>Simple network of hyperlinks and nodes</td>
</tr>
<tr>
<td>27</td>
<td>User</td>
<td>Easy to find information</td>
<td>HTML files</td>
<td>Easy to navigate</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Web server / Database server</td>
<td>Server is down</td>
<td>Server</td>
<td>Hardware failure</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>Server</td>
<td>Software failure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>Server</td>
<td>Attacked by hackers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>Server</td>
<td>Under maintenance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Database</td>
<td>No information stored</td>
<td>Server</td>
<td>Attacked by hacker</td>
<td>Information damaged by the hacker</td>
</tr>
<tr>
<td>33</td>
<td>Database</td>
<td>Information not updated timely</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The quality model derived from this part of SFMEA chart is given in Figure 11, where for the sake of space, the reasons of the links are not presented.
Figure 11 A part of the quality model for B2B e-commerce

As shown in Figure 11, the quality of the Product Information Manager part of B2B e-commerce systems involves a large number of quality attributes and quality carrying properties of various components of the system.
6. Conclusion

In this paper, we further developed the method of hazard analysis based approach to quality modelling of information systems proposed in [26, 27]. The main contribution of the paper is two-fold. First, a diagrammatic notation is proposed to represent quality models of information systems. It enables the explicit references to the components of the information system whose quality-carrying properties effect the system quality attribute. It also enables the explicit annotation of the reasons why two properties or attributes are related. Containing such information in quality models can significantly improve the usability of quality models in information system development. Second, the failure mode and effect analysis method originally developed for hazard analysis of safety critical systems is adapted for the analysis of software and information systems. The result of this adapted method can be directly used to construct quality models of information systems. It provides a logic that bridges the gap between abstract system quality attributes and the tangible quality-carrying properties of components and the observable behaviour of the system and their components.

There are also a number of other advantages of the proposed method. First, it enables software engineers to derive quality models at an earlier stage of software development in comparison with similar methods such as the card sorting method [24]. This is particularly important because the awareness of a sensitive quality attribute at early stage such as at design stage enables software engineers to seek for technique solutions to achieve the required quality standard.

Second, derive quality models at architectural level enable software engineers to understand the quality of a type of software systems in a particular application domain and of the same architectural features. The results of such quality modelling have a wide applicability. Yet, when more details of the structure and functions of the components are provided, more details of the quality model can be obtained, and thus, provide more insight information for follow up development activity.

The preliminary study of the method seems very promising. There are a number of directions for further work. For example, how to identify failure modes systematically for each component needs further investigation. It seems that the HAZOP hazard analysis technique can be adapted for this purpose. Moreover, quality attributes are of different importance in information system development. How to assign weights to quality attributes needs further investigation. We can also learn from the methods and techniques of hazard analysis where the safety and risks of a system are quantitatively analysed according to the consequences of a hazard and the probability of the hazard occurs.

References

1992, pp175-191