ABSTRACT
Software products are a critical and strategic asset in an organization's business. They are becoming larger, more sophisticated and more complex. The challenge is to develop more complicated software products within the constraints of time and resources without the sacrifice of quality. Quality standards, methodologies and techniques have been continually proposed by researchers and used by software engineers in the industry. However, studies on quality have mainly focused on the final software product. This paper looks at the requirements document. It develops cognitive structures for the user and developer stakeholder groups from a qualitative study of requirement document quality. The paper compares these results with the cognitive structures of the finished software product.

Keywords
Requirements Engineering, Requirements Documentation, Quality, Cognitive Structures

1. Introduction
One finds almost as many definitions of quality as writers on the subject. Writers have been remarkably few in number considering the obvious importance of quality concepts and the frequent appearance of the term quality in our everyday language.

Though people have been discussing software quality for decades, software quality research is still relatively immature, and it is difficult for a user to compare software quality across products. Researchers are still not clear as to what a good measure of software quality is because of the variety of interpretations of the meaning of quality, of the meanings of terms to describe its aspects, of criteria for including or excluding aspects in a model of software, and of the degree to which software development procedures should be included in the definition. A particularly important distinction is between what represents quality for the user and what represents quality for the developer of a software product.

In a recent article titled 'New Years Resolution for Software Quality', ten distinguished individuals in the software quality field were asked to share the resolution they wished software organizations would make and keep in order to improve software quality [1]. The article showed that quality is complex, that there are many views of quality, and many views of what actions to take in order to improve software quality. Today, organizations in search of competitive advantages, have invested heavily in automating their business processes. Greater reliance is placed on the software products, to the point where software has assumed a critical and strategic role in organizations' business. With this level of importance and the reliance placed on software products, it has become a necessity to improve the quality of our software products.

We also need to improve the efficiency and productivity of the development and maintenance processes. As such, researchers and practitioners have been paying increasing attention in understanding quality and improving the quality of the software being developed. Some studies have focused on techniques and approaches to assure the quality of software products, whilst others have focused on the software development process, how to define it, evaluate it and improve it.

2. The meaning of "QUALITY"
The term "quality" is applied to virtually all products, businesses, professions and processes. It has many definitions, each with a different perspective. Researchers in the software engineering area have tried different ways of defining quality. They have adopted the product-based view ([2], [3], [4]), the manufacturing-based view ([5], [6], [7]), the user-based view ([8], [9], [10], [11]) and even combinations of the views ([12]).

There have been many software quality studies. More recent empirical studies have addressed the rationale behind the inclusion or exclusion of a particular quality factor or criterion, describing influences on different perceptions of quality ([13], [14], [15]). In addition, several organizations are recognizing the importance of an integrated view of quality and customer satisfaction ([16], [17]).

Robert Pirsig [18] comments "that quality is easy to see and is immediately apparent when encountered, but when you try to pin it down or define what it is, you find the concept is elusive and slips away". He states that "it is not the complexity, but the simplicity of quality that defies explanation" [18]. He sees quality as an abstract description. Garvin[19] describes this view of Pirsig as a
transcendental view, that is, "You know it when you see it, but you can’t describe what it is - it is a gut feeling". He describes this view as the one often taken by philosophers, and that if you were an engineer, or an economist, your view would be different. Garvin[19] describes other views of quality - the product-based view, the manufacturing-based view, the economics-based view and the user-based view. He shows how each of these views can be used to define product quality. Engineers who believe a product has set characteristics often adopt the product-based view. These characteristics are used as the measure of quality. The manufacturing-based view is adopted when one believes the quality development process determines a quality product. Recently, many organizations have obtained ISO9000 certification [20]. Certification assesses the manufacturing process and is awarded on successfully having a quality management system in place. The economics-based view is adopted by economist who believe that price has a correlation with quality. And lastly, the user-based view is the one which emphasizes that each individual will have their own perception of quality. Garvin[19] states that most existing definitions of quality fall into one of these categories be it conformance to specification, or meeting user requirements or best practice.

Pfleeger [21] supports this, stating that software quality is determined by the person analyzing the software and that people have different views of quality. The reasons behind why ones view and perception differ can be found in the studies performed in applied psychology. For example, the psychologist Kelly, quoted in Weiner[22], asserted that "meaning" is subject to change and depends on the eye of the beholder. Kelly believed that each person is an individual with his or her own views. A person’s perception is therefore guided by how he or she understands and interprets the world at large. Since individuals perceive the same situation and experiences in different ways, it follows that their perceptions of quality will also differ. Users, for example may judge software to be of high quality if it does what they want in a way that is easy to learn and easy to use. Software developers may judge software on how well the program is written, the choice of algorithms used, the efficiency and speed of the program. Software can be judged by those who are designing and writing code, by those who maintain the programs after they are written, by those who use the software and by the managers who pay for the software to be developed. Therefore, software quality, contain factors and characteristics which address the needs of users, developers, maintainers and managers, and these may differ from one person to the next ([14], [15], [21]).

Researchers in the software engineering area have tried different ways to define quality. They have adopted the product-based view ([11], [2], [3]), the manufacturing-based view ([4], [5], [6], [20], [23], [24]), the user-based view ([7], [8], [9], [10]) and even combinations of the views ([30]).

The models, which follow the product-based view, define quality by a collection of characteristics and factors ([11], [2], [3]). Though these models have the same view of quality and have similar characteristics, they differ in the number of factors and criterias. For example, Boehm’s model has seven characteristics [1], McCall’s has eleven characteristics [2], Bowen’s has thirteen characteristics [3], and the recent international standard for Information Technology software product evaluation, ISO9126 [25], has six characteristics [26], [27]. Kitchenham [3] made a number of observations about these product-based models. She stated that there seemed to be little supporting rationale for including or excluding any particular quality factor or criterion. Definitions of quality factors and criteria were not always consistent when comparing one model to the next. It appears that though the product-based view of quality has been widely used and accepted, the differences between them, highlights how ones view can dictate the perception of quality. These models, when used by people with different backgrounds and assumptions, can result in various interpretations for a particular characteristic.

The manufacturing-based approach has been another focus in the software engineering area. Its inception had been largely encouraged by the introduction of software process improvement practices ([4], [5], [6], [20], [23], [24]). The approach follows the manufacturing industry where the basic belief is that the quality of the finished product is determined and measured by the process used to create it ([6], [23], [24]). It is expected that by improving the development process, improvements in product quality will reduce re-work during development, and reduce maintenance effort after the products have been delivered, consequently reducing overall life cycle costs.

This manufacturing-based approach was initially promoted during the early 1980s by a small group of industrial and academic software engineers ([6]). Though this approach has much merit and has been applied successfully in a number of cases ([6], [23], [24]), it is not guaranteed that by trying to introduce new practices to improve the software development process, that one will obtain a quality product ([27]). Constantine [29] states that there is a danger of only focusing on the development processes and the technical practices, and losing sight of the people who must master and carry out the development processes. Even with such systematic approaches as the Software Engineering Institute’s Capability Maturity Model, unless new practices are fitting to the people who will put them into practice, real improvement may remain elusive [27].
Recent research has adopted the user-based view of quality ([7], [8], [9], [10], [11]). The view recognizes that each person is different and takes on a different perception of quality.

Vidgen ([9], [10]) proposed a framework to define quality based on the Multi-view development method ([28], [29], [30]). Vidgen believed that multiple perspectives of software quality are required if one is to assess product quality properly. The framework was based on customer satisfaction, relating the product with its use and the services provided to support it. These three views provide the basis for the multiple perspectives. It was not merely an exercise in looking at the same object from different angles but entailed different assumptions about what quality was.

Though the topic of software quality has been around for decades, software product quality research is still relatively immature, and today it is still difficult for a user to compare software quality across products. Researchers are still not clear as to what is a good measure of software quality because of the variety of their interpretations of the meaning of quality, of the meanings of terms to describe its aspects, of criteria for including or excluding aspects in a model of software, and of the degree to which software development procedures should be included in the definition ([31]).

Recent studies of Wong showed through empirical studies that different groups of people view quality in different ways, and that it is possible to group people with similar definitions of quality and similar choices of characteristics in their quality assessment process. More recently, the studies address the question of what influences the choice of characteristics used in quality evaluation, and what influences the different views of quality. These studies focused on whether the desired values sought by the evaluator determines their choice of characteristics for the quality evaluation, and whether people with the same desired values can be grouped together as having similar definitions of quality and similar sets of characteristics. Diagrams called cognitive structures were introduced to represent these relationships, and to show the differences in the views of quality.

Whilst the studies support the many views of quality and give empirical evidence for why the different views, the studies were all conducted on the finished software product. This paper addresses the requirements document and its quality. Are there different views of quality for the requirement document? How do the cognitive structures of software product quality compare with the cognitive structures of the requirement document quality?

3. Data Collection and Analysis
Eight subjects were interviewed singly and in-depth at a distribution organization during late 2002 and early 2003. All respondents were involved with the e-commerce project being evaluated, either as a user of the software or as a developer supporting the software. Four of the subjects were users and four were developers. The organization is an international distribution company of wireless voice and data products and a premier supplier of outsourced services with over 20,000 customers worldwide. The organization recently installed a new financial system, and aims to implement an e-commerce solution to improve and automate their supply-chain management and to introduce added value to customer service, like on-line customer order tracking, on-line product catalogs, and in-house management reporting. The study was conducted after the first phase of development was released. This first phase included a catalog system, and an automatic pricing system, which would calculate in real time appropriate pricing of products for each customer.

The people surveyed came from different jobs and backgrounds in this organization. All the respondents were familiar with the business and the objectives of the e-commerce project. We surveyed two programmers, a technical support leader, and an operations coordinator from the development side. The development team all had a minimum of 4 years experience in the I.T industry. We also surveyed 2 sales managers, financial controller, who "owns" the system, and the marketing manager from the user side.

The interviews aimed to focus on the respondent's perception of the quality of the requirements document, which consisted of both the requirements description and specification, and later the quality of the first release of the e-commerce system. There were no hints, nor guidelines used during the interview, which would influence the subjects to give any particular result. Most subjects were involved with both the evaluation of the requirements document and the e-commerce system. However, one sales manager resigned at the beginning of the year, and so another person became involved in the project and contributed to the second survey.

To identify the full set of linkages connecting means to ends, users were given a laddering task ([32], [33], [34]). The laddering procedure consists of a series of directed questions based on mentioned distinctions the individual has with respect to the quality of the software being evaluated. The purpose of the laddering is to force the user or developer up the "ladder of abstraction" to uncover the structural aspects of user knowledge as modeled by the means-end chain. The questioning procedure was designed around the unique demands of the laddering procedure. They were based on prior answers, the interpretation of the answers, and focused on "pushing the participant up" the characteristic-consequence-value hierarchy. Very often, further clarification of the answers was sought before introducing
another question. The laddering method has been widely used in the consumer research and psychology disciplines. It has become an accepted method to gain insights regarding sources of value that are perceived to be, or could become, motivationally important ([13], [35], [36], [37]).

In both sets of interviews, those conducted in late 2002 and those conducted in early 2003, warm-up questions were used to set the tone for the interviews. Subjects were asked about their interest in computers, in particular software, the web, e-commerce. Different types questions were used between the two sessions, but the aim was the same, that is, to relax the subject in order to obtain more detailed and spontaneous answers.

During the first interview, questions were asked regarding their desired e-commerce project. How much had they been involved in the preparation of the requirements document? How do they rate the finished document? These questions allowed for the "laddering" approach to then take over. The questions, which followed, then asked why they rated the document in that way, and what contributed to that assessment. The interview would then focus on each reply, with continual questions based on each answer given. The questions and answers created a chain starting from the characteristic used in the quality evaluation, with links to the desired consequences, until finally ending with the value sought. This process would be repeated continuously for each identified characteristic, resulting in a number of "ladders" being created for each respondent.

During the second interview, questions were asked about how much they had used e-commerce systems. Respondents were asked to explain how they could tell if the e-commerce site and the software was of high quality or not. Next, subjects were asked to rate the quality of their e-commerce system, that is, whether the system was high quality or not. The subjects were asked to describe why the quality rating and what were the influences which led them to that score. As with the first interview, the laddering approach was used and a number of "ladders" were created for each respondent.

An example of the use of this laddering approach is given in an earlier study of Wong & Jeffery [13], where an excerpt of one of the respondents is given and described. It must emphasized, that each question is carefully formulated to not bias the results. At times throughout the interview, clarifications of the replies are made to ensure correct interpretation of the dialogue. Whilst this may appear to bias the study, it should be pointed out that what is clarified, is as a result of what had already been said. Unfortunately, the limited size of this paper prohibits giving examples of the interviews.

After performing the interviews, the transcripts were analyzed. The first step in the analysis was to conduct a thorough content analysis of all the elicited concepts. All the responses at the characteristics level were considered first, so that terms close in meaning could be grouped together. The goal here was to reduce the fragmentation of responses that occurred when respondents were using their own language or terminologies, without losing meaning, by grouping elements with widely divergent meanings into the same category. This procedure was repeated at the consequence and value levels. All laddering responses were then expressed in a set of standard concepts. The aggregate set represented the content component of the respondents' quality evaluation structure. The results are then represented in tables, and then modeled using a structured chart, one chart for the users, and one for the developers.

4. Results

Any instance in which a subject links at least two elements together in an asymmetrical fashion (A causes, produces, or leads to B) is defined as a ladder. In all, 17 ladders were elicited from the subjects for the requirements document, the shortest ladder having a length of two and the longest ladder a length of four. And 28 ladders were elicited from the subjects for the evaluation of the e-commerce system, with the shortest ladder having a length of two and the longest ladder a length of six. The typical ladder was comprised of from three to five elements, although there were many two-element ladders (that is, when a number of characteristics lead to the same consequence).

It would be trivial to just address the issue of stakeholder differences between users and developers, as this difference is obvious. However, it is of interest to identify what the desired values and consequences are for each stakeholder, and to determine whether they are the influence for the choice of characteristics and measurements in their evaluation of requirements document and e-commerce system.

The results show that the users all focused on valuing "Warm Relationships with others" and "Job Security". No matter which characteristic was being discussed, the desired consequences and values would always lead to these two values, even though obvious differences between the evaluation of requirement documents and software quality exist.

The characteristics elicited by the users for evaluating the requirements document were related to the characteristics economics, functionality, usability and operational issues. The users did not mention any measurements relating to the characteristics support nor technical issues. This list of characteristics was also mentioned for the evaluation of software quality. The only difference being that the users also mentioned the characteristic support.

The consequence of "TIME" was also continuously being raised in both the evaluation of the requirement
document and the evaluation of software quality. Whether it was referring to timely software development or whether by having a quality system, more time was available, TIME was definitely a desired consequence. “Flexible/can do my job easily in different ways”, “Can do job faster/quicker/saves time”, “Not feel rushed at work/can take my time”, “Time for higher priority work”, “Time for other things”, all focused on wanting the system to save time. Respondents highlighted that a poor system, or poor requirement document which lead to poor systems design, would result in time being taken away from “Time with friends and family”. If the requirement document was not clear, or not well written, then there is potential confusion for the developers. Time would be wasted in trying to understand the specifications, or worse, time would be wasted by incorrect development. As to the e-commerce system, time for the respondents would be highly effected if the e-commerce system were poorly developed. If the system were not flexible, time would be wasted. If the system failed to give correct reports or results, time would be wasted with corrections, reruns, and seeking help from IT. All this would increase stress, and worry, resulting in poor quality of life.

Characteristics such as functionality and usability were also mentioned continuously for both requirement document and software quality. However, it should be noted that the measurements for each differed. Of course, this is expected, as the artifacts are different. It is also interesting that functionality was a major issue for the evaluation of the requirement document. Whilst it is true that functionality is also important for software quality, the users on evaluating the requirements document appear to be highly focused on functionality issues. Usability is important, only in that the requirements document is easy to understand, consistently written, and easy to follow. Whereas usability for the software evaluation focused more on the e-commerce system, and how easy it is to use and learn, and how flexible it is to navigate between screens.

Other characteristics were raised, such as cost or value for money, brand name, and reputation. However they were mainly for the evaluation of the software and not for the evaluation of the requirements document. Not much discussion was given to these characteristics, though the users all stated that these characteristics led to having a system, which delivered better quality of life and better job security.

The results also showed that both users and developers focused heavily on their jobs and how the job affected their personal life. Also, it would appear that there are a lot more similarities between the users and developers when evaluating the requirements document. However, as was the case with the earlier study from Wong & Jeffery [13], the users and developers differ substantially for the evaluation of software quality.

As with the earlier studies, success in the delivery of new software applications boosted the developers’ self-confidence and self-esteem. No matter which characteristic was being discussed, the desired consequences and values would always lead to these values. Unlike the users, technical characteristics played a very important role in the evaluation of software. This supported the findings of earlier studies ([14], [15]). Much of the developers’ focus, were on development and programming design and approaches, the development process, program documentation and tools. The discussions centered around the problems faced when lack of adequate processes, documentations and tools. Frustration was raised when discussion moved towards maintenance and enhancements of poor quality software. Similar comments from a number of the developers highlighted the lack of enjoyment in their job when appropriate programming practices were not followed.

On the hand, the users like the developers found functionality, usability and support to be important for evaluating software quality. However the focus of the developers were different from the users. Rather than support for the developed application, the developers would be looking at support from the manufacturers, for example, the hardware manufacturers, the database supplier. Rather than looking at functionality sought by the users’ job, developers were more concerned with functionality for maintenance and enhancements, easy portability between different pieces of hardware, which they needed to support. Rather than ease of use referring to the users being able to do their job easier, the developers were more concerned with having happier users, so that there would be fewer complaints from the users. Of course, having software, which does not fail, has been identified as important. In all, it is quite obvious from the interviews that what is closes to the hearts of developers are technical issues, even when one considers characteristics such as operations, support, functionality and usability.

With both the requirements document and the software quality, the developers gave no consideration to characteristics such as cost, value for money, brand name, or reputation. Though small amounts of interest came from the users, it is suspected that perhaps the manager would have more interest.
The structures in figures 1, 2, 3 and 4 represent the connections among elements in the means-end chain. Figure 1 represents the users' cognitive structure for the evaluation of the requirements document, and figure 2 the developers' cognitive structure for the evaluation of the requirements document. Figure 3 represents the users' cognitive structure for the evaluation of the software quality, and figure 4 the developers' cognitive structure for the evaluation of the software quality. In both structures, the bolded elements represent the desired values, and the lines joining the elements, the links, which form the ladder from the characteristics to the desired values. The cognitive structures show the motivation for the measurements. From the diagrams it is evident that the measurements, and the characteristics, for requirements document are different from the measurements for software evaluation. However, as can be seen by the cognitive structures, the motivation behind these measurements appears to be similar between the two artifacts. The desired consequences and the values sought, regardless of the artifact are similar for each stakeholder. It is also evident from the tables that the stakeholders differ in the motivation. As such the cognitive structures differ between the stakeholders for the evaluation of each artifact.

5. Conclusion
The goal of this study was to determine whether there are different views of quality for requirement documents. Therefore the following questions are asked. Are there different views of quality for the requirement document? How do the cognitive structures of software product quality compare with the cognitive structures of the requirement document quality?
Though the study is exploratory, and the findings require further empirical work, the results propose possible answers. It would appear from the results that the answers do not appear to be straightforward. To begin with, the results show that measurements used by a particular stakeholder group, e.g. user or developer, to evaluate quality between the requirements document and the final software product differ, and are influenced by the type of artifact being measured. However the artifact type does not influence the motivation for the measurement and the view of quality. The motivation is the same for the requirement document and the software product. This result is consistent for both stakeholders.

Another result is that it is evident that the users and the developers differ regardless of whether the quality evaluation is on the requirements document or the final software product. However, the measurements identified in the requirements phase are similar between the stakeholders, though what motivates the choice of measurement is different.

As with earlier studies of Wong ([13], [38], [39], [40]), the research finds that the desired values sought by the quality evaluator, determines the choice of characteristics used in the evaluation. The research also finds support for requiring a multi-view approach to evaluating requirements documents and for the evaluation of software quality. The cognitive structures show quite clearly how users and developer differ, but show similarities in the characteristics used to measure requirement documents. As such the study has demonstrated the benefit of cognitive structures as a tool for representing the measurements with their motivation. And finally, the research helps to focus on areas of measurements, which are important at each milestone, but also what consequences and values need to be predicted.

These findings are valuable to software quality practitioners and software engineers. The results help to clarify areas of measurements, and the goals sought by the stakeholders. It helps to explain the differences and the similarities but more important, why the differences. Activities such as inspections, walkthroughs, reviews, and testing can be more focused on measurements identified as important to the stakeholder and prioritized. Estimates of impact on the consequences and values can be potentially calculated, even from the requirements document phase, and can be recalculated at each milestone as a means of tracking a software project quality. However, the study has only investigated users and developers of an e-commerce application, and has only focused on two milestones from the development life cycle. There are many more stakeholders, and many more milestones. Further studies are required to address these areas.

6. REFERENCES


