2010 Seventh International Conference on the Quality of Information and Communications Technology

A Metrics-Based Approach to Technical Documentation Quality

Anna Wingkvist*, Morgan Ericsson†, Rüdiger Lincke* and Welf Löwe*
*School of Computer Science, Physics, and Mathematics,
Linnaeus University, Sweden
Email: {anna.wingkvist|rudiger.lincke.|welf.lowe}@lnu.se
†Department of Information Technology,
Uppsala University, Sweden
Email: morgan.ericsson@it.uu.se

***Abstract*—Technical documentation is now fully taking the step from stale printed booklets (or electronic versions of these) to interactive and online versions. This provides opportunities to reconsider how we define and assess the quality of technical documentation. This paper suggests an approach based on the Goal-Question-Metric paradigm: predefined quality goals are continuously assessed and visualized by the use of metrics. To test this approach, we perform two experiments. We adopt well known software analysis techniques, e.g., clone detection and test coverage analysis, and assess the quality of two real world documentations, that of a mobile phone and of (parts of) a warship. The experiments show that quality issues can be identified and that the approach is promising.**

***Index Terms*—Information Quality, Quality Assurance, Software Metrics, Documentation**

I. Introduction

The quality of technical documentation and user manuals is an important part of the perceived quality of a product. When end users encounter problems or want to learn more about the product they turn to its documentation. The documentation constitutes a first line of support, and it is important that end users can trust it and get the information they require.

As documentation moves from printed books and booklets to electronic versions, it is no longer bound to the limitations of "stale" paper. The electronic version can, for example, include powerful navigation (such as search and hyperlinks), video, or even interactive content. The electronic version also has the advantage of being easier to keep up to date.

Electronic versions of the documentation have changed the role of software. Previously, it was used only to produce the documentation, but now software is also used to deliver and access it, and can even be seen as a part of the documentation, especially in regard of presentation. The technical documentation is becoming deeply integrated with the product and the boundary between technical documentation and software is becoming less and less clear. The car manufacture Hyundai is a recent example where technical documentation has come "alive" in form of a software product. They provide the owner's manual of one of their car models as an application for the Apple iPad (which is included in the purchase of a car).

As information production and presentation is bound to software systems, the notion of information quality can be further explored using means of software quality management. The quality of the software used as part of the processes has a large impact on the perceived quality of the documentation. For example, the quality of the browsers, their search function, or the video codecs largely influences the perception of the information presented. Traditional means to manage quality of software systems involves software analysis and testing. It is natural to extend these approaches to cover quality of technical documentation.

We suggest such an approach to assess and assure the quality of technical documentation. Based on our experience with measuring software quality [2, 17], we build on the Goal-Question-Metric (GQM) paradigm [4] and suggest a metrics-based quality model. Quantitative metrics are collected throughout production and use of the technical documentation. The metrics can together be used to answer operational questions regarding quality. The conceptual goals determine the quality as perceived by the stakeholders. Considering the outcome, the metrics can also be used to visualize and pinpoint quality defects.

We validate our metrics-based approach by two experiments at companies producing technical documentation. We adapted a number of metrics and assessed the documentation connected to two products, a mobile phone and a warship.

The rest of the paper is organized as follows. Section II discusses the notion of quality from an information and software perspective. Section III then defines our quality model for technical documentation. Section IV presents the experiments and Section V concludes the paper.

II. Quality

The notion of quality is used in daily language as an intangible trait, something that can be subjectively felt or judged, but often not exactly measured or weighted. Terms like good or bad quality are intently vague and used with no intention of ever being an exact science. Another issue is that quality is multidimensional, and includes an object of interest, the viewpoint on that object and the quality attributes attributed to the object. This can make "quality" a confusing concept.

978-0-7695-4241-6/10 $26.00 © 2010 IEEE
DOI 10.1109/QUATIC.2010.88

In order to discuss quality, and to be able to assess and improve it, it has to be defined. Crosby [6] defines quality as *"conformance to requirements"*. This definition suggests that there must exist a set of requirements that are defined in such a way that they cannot be misunderstood. Something that is of quality conforms to these requirements and anything that does not conform is considered a defect. The requirements are not to be considered universal, but set by an entity or for a single product.

Another notion of quality is given by Juran [13] who defines quality as *"fitness for use"*. This definition considers the customers, and the requirements and expectations that they have on the product and their use of it. Juran further states that since different customers may use the product in different ways, the product must possess multiple elements of fitness of use. These elements are quality characteristics that can be divided into parameters.

The two definitions of quality can seem unrelated, but complement each other in reality. The demands and expectations of the customers guide the requirements set by the producers. So, a conformance to the well motivated requirements generally means that the product will be fit for use as well.

Quality assurance and management processes and practices are standardized, e.g., in the ISO 9000 family of standards [12].

A. Software Quality

McCall et al. [18] present a *quality model* defining eleven quality factors that relate to the three stages of a simplified software life-cycle: revision, operation and transition. McCall et al. also define about 22 metrics that are used to measure the quality of the eleven factors. Several metrics are weighted and used to determine the quality of each factor. Many of the metrics are based on checklists and a 0 to 10 scale, which means that they are subjectively measured. This quality model is standardized in ISO/IEC 9126-1:2001 [11]. Quality models also differentiate perspectives, stages or phases. There are several aspects of quality and not all of these may be appropriate for all perspectives and phases. McConnell [19] differentiate between internal and external quality, i.e., quality that affects the product while produced versus quality when the product is in use. These aspects are standardized in ISO/IEC 9126, as well.

The model by McCall et al. introduces several important ideas. First, there is not one product quality, but several factors that affect the product quality. Second, these factors matter during different periods of the life cycle. Third, the quality factors should be measurable and metrics should be defined. Several modern quality models and metrics suites exist, such as those by Li and Henry [16], Chidamber and Kemerer [5], Abreu [1], and Henderson-Sellers [10]. There are several studies that validate the claim that metrics can be used as an indicator of the quality of software, for example Basili et al. [3] and Harrison et al. [9].

B. Information Quality

Building on Juran [13], Wang and Strong [22] define IQ as fitness for use by information consumers. However, Klein et al. [15] shows that information consumers have trouble pointing out and detecting errors in information and altering the way they use it. Hence, placing all the responsibility on a user is not appropriate. In response to this, Kahn et al. [14] suggests a view where quality also depends on conforming to specifications or requirements (adopted from Crosby [6]). These two views are called user perspective and data perspective, respectively.

In the process of assessing IQ, Ge and Helfert [8] classified typical IQ problems in a two by two model distinguishing user and data perspective, and context-independent and context-dependent aspects, respectively (see Table I). The user perspective/context-independent quadrant indicates IQ problems that may happen while processing the information. The user perspective/context-dependent quadrant indicates IQ problems that are due to unfitness for the intended use by information consumers. The data perspective/context-independent quadrant indicates IQ problems in the content management systems (CMS). These IQ problems can be applied to any data set. The data perspective/context-dependent quadrant indicates IQ problems due to violation of business specifications. These IQ problems can be detected by contextual rules.

This assessment method of IQ is inherently subjective and not automated. To the best of our knowledge, unlike in software quality assessment, automated measurement and testing method do not exist for the assessment of IQ. For example, SQuaRE (Software Product Quality Requirements and Evaluation) uses the software quality model of ISO/IEC 9126-1:2001 and distinguishes the software quality (ISO/IEC 25010) from the data quality (ISO/IEC 25010) of an information system. However, while data quality affects the information quality, they are not the same. The goal of this paper is to adapt automated methods known from software measurement and testing to the assessment of IQ.

III. A Quality Model for Technical Documentation

Information and software quality need to acknowledge the different approaches in defining and assessing quality. A common basis is that the respective notion of quality can be approximated by a set of attributes that describe what defines high quality. These attributes are in turn approximated by a set of metrics.

The various quality models defined for information shows that there can be a huge gap between an ideal definition of quality, an approximation by attributes and what can be measured by metrics. For example, the negative quality attribute *"the information is hard to understand"* (cf. Table I) can be very difficult to approximate by a set of metrics. We conclude that the metrics should be complemented by human insights.

Quality models for information and software (cf. Section II) quickly grow complex. Stvilia et al. [20] point out that one cannot manage information quality without first being able to

Table I
CLASSIFICATION OF INFORMATION QUALITY PROBLEMS (ADAPTED FROM GE AND HELFERT [8]).

	User perspective	**Data perspective**
Context independent	The information: is inaccessible, insecure, hardly retrievable, difficult to aggregate, errors in the information transformation exist.	Inconsistent data format, spelling errors, missing, outdated or duplicate data, incorrect value, incomplete data format, syntax or unique value violation, violation of integrity constraints, text formatting.
Context dependent	The information: is not based on fact, is of doubtful credibility, presents an impartial view, is irrelevant to the work, is compactly represented, is hard to manipulate, is hard to understand.	Violation of domain constraint, of organization's business rules, of company and government regulations, of constraints provided by the database, consists of inconsistent meanings.

measure it meaningfully. We agree with this statement and take it one step further. One cannot manage quality without a meaningful way to communicate the measurements and actual quality issues to the different stakeholders, i.e., there must be a way to effectively visualize the quality (issues).

Based on these observations we formulate three requirements on a quality model for technical documentation.

1) The model should not aim to be static but rather be flexible enough to allow for users to weigh in on what they consider high quality.
2) The model should be based on objective measurements and metrics if possible, but complemented with subjective measurements and metrics (results from check lists, questionnaires, etc.).
3) The model should support visual feedback throughout the quality assessment and assurance process.

The Goal-Question-Metric (GQM) paradigm is an approach to quality that fits the requirements set above well. GQM is a goal-driven approach to quality evaluation based on measurements. It contains both a generic measurement model and a process that describes how to instantiate the generic model to a specific situation as well as how to collect measurements. The GQM approach provides a method to create quality models specific to, for example, a company, project or product. The metrics are an important part and each model is defined in terms of how the quality attributes can be measured.

The measurement model consists of three levels; the conceptual (Goal), the operational (Question), and the quantitative (Metric). Goal is an overarching property that the object of study should possess. In terms of quality, a goal represents a specific quality goal, for example *"high accessibility of information"*. The goal is supported by one or more questions to determine if the goal is achived or not. Questions, such as, *"appropriate access behavior"* or *"completeness of testing"* support the goal. Each question is answered by (quantitative) data provided by metrics. Metrics to support the questions can, for example, include the time it takes to find information and the access path used (cf. Figure 1).

Quality can be abstracted in different ways. It might, for example, be desirable to know if there are any defects, how many defects there are, or where the defects are located? These questions constitute different abstractions of the quality and how it is measured that are suitable for different stakeholders inside and outside of an organization. The customer might

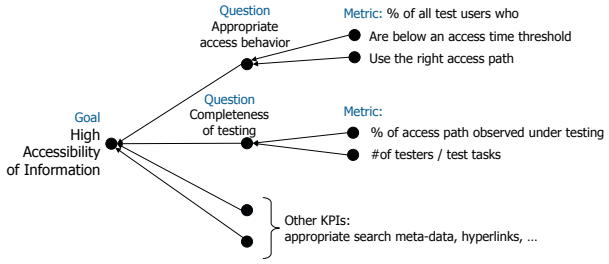


Figure 1. An example of the Goal-Question-Metric approach applied to technical documentation

be satisfied by an assurance that the quality is high, while a technical writer or developer need to know the *"what, where and why"* of each defect. All the abstractions are supported by different views on the same metrics, and correspond to the conceptual, operational, and quantitative levels of the GQM.

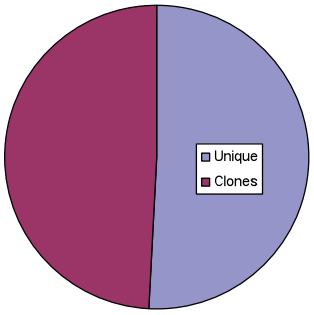
For example, consider the goal to have as little duplicated information as possible in some technical documentation. This can be supported by metrics that measure uniqueness such as percentage of duplicated text and structure. Figure 2 depicts four different views of this quality. Figure 2(a) shows how much of the total technical documentation is unique. Figures 2(b) and (c) show text and structure duplications per document, while (d) shows duplications between two files. The figures are explained in more detail in Section IV-B. All views are abstractions of the same underlying metrics.

IV. STUDY

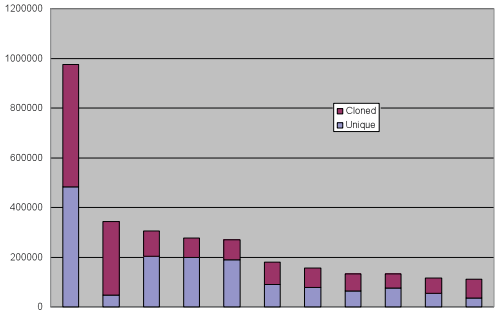
This section described a study conducted where we applied the metrics-based approach (cf., Section III) to measure the quality of the documentation of two products: a mobile phone and a warship. In the case of the warship we were limited to the non-classified parts of the documentation. The study focused on metrics originating from software quality assessment. It was conducted to demonstrate their usefulness in assessment of real world technical documentation.

A. The Setup

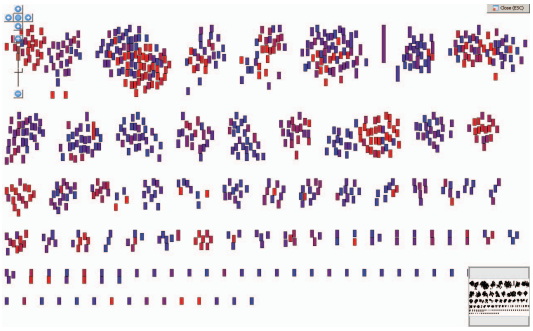
We decided to use two metrics that are well known from a software engineering point of view. We measure *"number of clones"* and *"test coverage"*. Clone detection for software is done to ensure that the source code is easy to comprehend, since repeating patterns that differ only by a few lines or



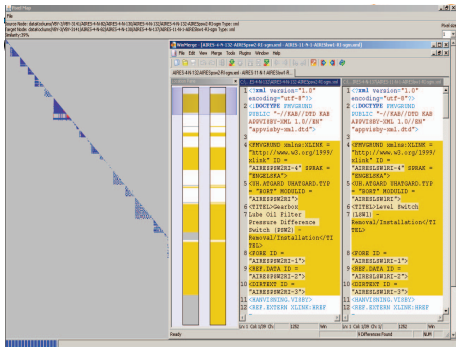
(a) Clones in the total documentation



(b) Ratio of the ten largest documents



(c) Documents clustered by similarity



(d) A comparison of two documents

Figure 2. Different views of the clone detection metric

characters can make it difficult to understand. Also, clones significantly increase the effort needed to enhance or adapt source code. In software engineering, clones can be considered a mark of a poor and lazy programming style and often increase maintenance costs.

Test coverage captures the amount of source code of that has been checked. When testing software, a test program is run with predefined input to see if the expected output is delivered (in time). However, due to theoretical and practical limitations (testing is an expensive software development activity), *complete* testing is not realistic. It is however crucial for the relevance of the test results to ensure that the sampled program runs cover a large portion of the source code. In other words, a successful test may not be relevant if it covers a fraction of the total source code.

Metrics based on clone detection for technical documentation can provide a measurement of a document's uniqueness – what fraction of the document is not part of any other? – and the similarity of documents – what fraction of two documents are clones? It is easy to argue that clone detection-based metrics (uniqueness and similarity of documents) are indications of one aspect of quality in technical documentation; namely maintainability. If the quality model emphasizes maintainability of the documentation, quality assessment should value a high degree of uniqueness and a low degree of similarity between documents. In the case of technical documentation, duplicated information effect the costs for adaptation and maintenance negatively. Redundant text is cumbersome to handle and expensive in regard to translation and printing costs, and should consequently be minimized.

Testing of technical documentation is done manually, e.g., by asking subjects to find information necessary to complete a certain task. The test is successful if sufficiently many subjects found the relevant information (in time). Overall success need to take test coverage into account, i.e., the fraction of the technical documentation that subjects needed to read to find the relevant information. While feedback from testing is a subjective measurement, test coverage can be measured automatically. Test results and coverage should be combined to assure documentation quality; a positive test result of tests that cover only a fraction of the documentations does not assure high quality of the whole documentation. Test success and coverage metrics are also are indicators of usability. If the quality model emphasizes usability of the documentation, the quality assessment should value high success rates and high coverage.

The technical documentation used in these experiments is produced using DocFactory [7]. To assess the quality of the documents we used the software quality analysis tool VizzAnalyzer [21]. VizzAnalyzer supports a wide range of metrics, such as, clone detection and coverage analysis, used as part of this experiment. A number of tools, such as Microsoft Excel and the yEd graph viewer, were used to visualize the results of the metrics.

The documents that were part of the technical documentation analyzed were exported from DocFactory as XML. Access logs were also exported to show different access paths within the documentation. These documents and logs were imported into VizzAnalyzer to calculate the metrics. The data produced by the metrics was then imported into Excel where it was further analyzed.

B. Results

Clone detection determines the similarity between two documents by first comparing the text on paragraph level and then the XML structures. The result is a percentage that indicates the relative size of the two documents that is unique.

We applied clone detection to 890 XML documents of the warship documentation. Out of these 890 documents, 6 were completely unique and 20 were complete copies. The remaining 864 documents were at least in part non-unique. On average, a document was 54.3% unique, but the degree of uniqueness was distributed rather evenly, with a median of 59.2% unique a standard deviance of 24.8%. In total, 49% of the combined content of the 890 documents was copied.

Figure 2 shows the results of the clone detection. Figure 2(a) shows the ratio of unique and cloned document parts integrated over the whole documentation and (b) shows this ratio for the ten largest documents. The former indicates that cloning is an issue (e.g., for a manager); the latter indicates that it is easy to do something about it (e.g., for a product manager) by removing clones from large documents. Figure 2(c) shows (e.g., for a sub project manager) which documents are similar and how similar they are. Each box represents a document and the color represents its uniqueness. The boxes are placed closer to each other if they are similar, so each of the visible clusters represents a set of similar documents. A document can only be part of one cluster. Figure 2(d) shows a view on a very low abstraction level (e.g., for a technical writer) that compares two documents and highlights similar parts.

To test the scalability of our approach, we repeated the experiment with the documentation of a mobile phone. This documentation consists of 8674 XML documents and the analysis time increased to almost an hour. This makes it impossible to include in an editing process.

Test coverage measurement statically analyzes the whole structure of a technical documentation, dynamically logs the documents and hyperlinks followed during testing, and correlates the static and dynamic information. Additionally, it may also be correlated to the outcome of testing.

We applied the test coverage metric to a version of the mobile phone documentation in an early development stage (the whole structure contained only 70 documents) and our own reading of that documentation to understand its structure (testing). During the two minutes of testing, we visited only 17 documents, which accounts for a 20% document coverage. The visit time of the individual documents was between 1 and 20 *sec*.

Figure 3 illustrates how coverage can be correlated to the structure and visualized. Figure 3(a) shows how large part of the documentation that was visited. Figure 3(b) shows which documents were visited and for how long. It depicts the document structure (boxes correspond to individual documents) and encodes the time spent on individual documents in different colors: light-blue boxes are documents not visited at all; the color gradient from dark blue to red corresponds to how long the documents were visited (dark blue representing the minimum and red the maximum amount of time).

We repeated the test coverage experiment with the 8674 XML documents of the full mobile phone documentation. The test coverage measurement scales well even for this large documentation; the analysis took less then a minute.

C. Evaluation

The experiment shows that the two selected metrics can be applied to technical documentation. Both test coverage and clone detection can be used to detect quality issues within the documentation.

Scalability may be an issue when trying to integrate quality assessment in the editing process – it may be necessary assign dedicated quality assessment phases in the information production process.

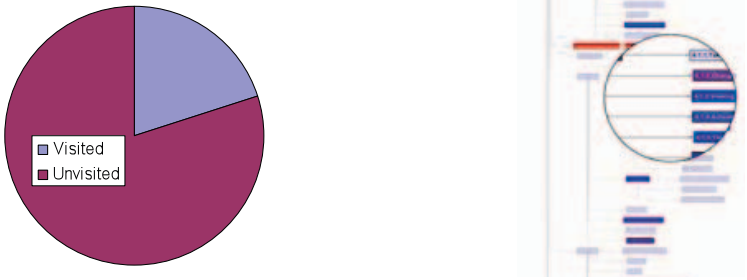
A general issue in using metrics quality assessment is that the metric values can be hard to interpret. What does it mean that 49% of the documentation is non-unique? Is this a sign of poor quality? In software quality assessment, there exist certain recommendations, but more often than not the meaning of the values depends on the product [2]. However, there are still several uses for the metric values. They can be compared to see how the quality changes over time, be used to identify potential issues that can then be investigated in detail, and so on. The use of different views and visualizations supports this.

V. Conclusions and Future Directions

This paper suggests an approach to quality management of technical documentation based on the Goal-Question-Metric paradigm and software quality metrics. We do not attempt to define a complete quality model, but rather leave it to the people in charge of a product, project or company to define the quality goals. The use of metrics to assess quality is key to our approach. The metrics provide a way to assess and visualize quality. An important part of our approach is the use of different views to visualize quality.

To test our approach, we conducted a study using real-world technical documentation and tools. We used clone detection and test coverage, metrics well known from software engineering, to evaluate quality of the technical documentation of a mobile phone and a warship. The study indeed identified some quality issues. We consider this an indication that a metrics-based approach can be used to assess and assure the quality of technical documentation.

The experiments presented in this paper employ a limited number of metrics and only considered (parts of) the technical documentation of two products. The next step would is to increase both the number of metrics and the scope of the technical documentation. The approach taken in the current



(a) Total test coverage (b) The documents visited

Figure 3. Different views of the test coverage metric

study follows the *"conformance to requirements"* view on quality, and future experiments should include the *"fitness of use"* model and consider how technical documentation fits the needs and requirements of the end users. Further work will focus on quality questions relying on metrics that are not as easy to measure automatically. Hence, we need to consider how to incorporate user samples, and how goals and questions should be formulated when there are multiple stakeholders.

ACKNOWLEDGMENT

We would like to thank fellow colleague Ola Petersson for valuable insights and comments that helped improve this paper. We want to acknowledge the Swedish Research School of Management and IT (MIT), Uppsala, for supporting Anna Wingkvist. Morgan Ericsson is supported by the Uppsala Programming for Multicore Architectures Research Center (UPMARC). The work was in part financed by the Knowledge Foundation with the project, "Validation of metric-based quality control", 2005/0218. We also would like to extend our gratitude to Applied Research in System Analysis AB (ARiSA AB, http://www.arisa.se) for providing us with the VizzAnalyzer tool and to Sigma Kudos AB, (http://www.sigmakudos.com) for providing us with DocFactory and raw data.

REFERENCES

[1] F. B. Abreu, "The MOOD metrics set," in *Proceedings ECOOP Workshop on Metrics*, 1995.

[2] H. Barkmann, R. Lincke, and W. Löwe, "Quantitative Evaluation of Software Quality Metrics in Open-Source Projects," in *QuEST '09, Bradford, UK*. IEEE, 2009.

[3] V. R. Basili, L. C. Briand, and W. L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE TSE*, vol. 22, no. 10, pp. 751–761, 1996.

[4] V. R. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach," in *Encyclopedia of Software Engineering*. Wiley, 1994.

[5] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object-Oriented Design," *IEEE TSE*, vol. 20, no. 6, pp. 476–493, 1994.

[6] P. B. Crosby, *Quality is free : the art of making quality certain*. McGraw-Hill, New York, 1979.

[7] "Docfactory," http://www.sigmakudos.com/data/services/docfactory, accessed April 2010.

[8] M. Ge and M. Helfert, "A review of information quality research — develop a research agenda," in *Proceedings of the 12th Int. Conf. on Information Quality*, Nov. 2007.

[9] R. Harrison, S. J. Counsell, and R. V. Nithi, "An Investigation into the Applicability and Validity of Object-Oriented Design Metrics," *Empirical Software Engineering*, vol. 3, no. 3, pp. 255–273, 1998.

[10] B. Henderson-Sellers, *Object-oriented metrics: measures of complexity*. USA: Prentice-Hall, Inc., 1996.

[11] ISO, "ISO/IEC 9126-1 "Software engineering - Product Quality - Part 1: Quality model", 2001.

[12] ISO, "ISO 9000:2005 "Quality management systems - Fundamentals and vocabulary", 2005.

[13] J. Juran, *Juran's Quality Control Handbook*, 5th ed. McGraw-Hill, 1998.

[14] B. K. Kahn, D. M. Strong, and R. Y. Wang, "Information quality benchmarks: product and service performance," *Commun. ACM*, vol. 45, no. 4, pp. 184–192, 2002.

[15] B. D. Klein, D. L. Goodhue, and G. B. Davis, "Can humans detect errors in data? Impact of base rates, incentives, and goals," *MIS Q.*, vol. 21, no. 2, 1997.

[16] W. Li and S. Henry, "Maintenance metrics for the object oriented paradigm," in *IEEE Proc. of the First Int. Software Metrics Symposium*, May 1993, pp. 52–60.

[17] R. Lincke and W. Löwe, "Foundations for Defining Software Metrics," in *Proc. of ATEM 2006, Italy*, 2006.

[18] J. A. McCall, P. G. Richards, and G. F. Walters, "Factors in Software Quality," NTIS, NTIS Springfield, VA, Tech. Rep. Volume I, 1977, nTIS AD/A-049 014.

[19] S. McConnell, *Code Complete, Second Edition*. Redmond, WA, USA: Microsoft Press, 2004.

[20] B. Stvilia, M. B. Twidale, L. C. Smith, and L. Gasser, "Assessing information quality of a community-based encyclopedia," in *Proc. of ICIQ*, 2005.

[21] "Vizzanalyzer," http://www.arisa.se/vizz_analyzer.php, accessed April 2010.

[22] R. Y. Wang and D. M. Strong, "Beyond accuracy: what data quality means to data consumers," *J. Manage. Inf. Syst.*, vol. 12, no. 4, pp. 5–33, 1996.