Validity of the Documentation Availability Model: Experimental Definition of Quality Interpretation

Raimundas Matulevičius[1,2], Naji Habra[1], and Flora Kamseu[1]

[1] PReCISE, Computer Science Faculty, University of Namur
rue Grandgagnage 21, 5000 Namur, Belgium
{nha,fka}@info.fundp.ac.be
[2] Institute of Computer Science, University of Tartu,
J. Liivi 2, Tartu, Estonia
rma@ut.ee

Abstract. System and software documentation is a necessity when making selection and acquisition decisions, when developing new and/or improving existing system and software functionality. A proper documentation becomes even more crucial for open source systems (OSS), where, typically, stakeholders from different communities are involved. However there exist only limited or no methodology to assess documentation quality. In this paper we present a quality model and a comprehensive method to assess quality of the OSS documentation availability (*DA*). Our contribution is threefold. Firstly, based on the criteria defined by Kitchenham *et al.* we illustrate the theoretical validity of the *DA* model. Secondly, we execute the first step towards the empirical validity of the *DA* model. Finally, our work results in a comprehensive and empirically grounded interpretation model of the documentation quality.

Keywords: Documentation availability, theoretical and empirical validity, quality metrics and indicators, open source software documentation.

1 Introduction

Nowadays, information systems (ISs) are based on open source software (OSS) products either partially or fully. OSS applications might vary from the middleware and platform levels (e.g., operating systems, software development environment) used to support ISs' development to the direct management and dissemination of information (e.g., calendars, text editing and Web applications). Working with a system of interrelated products implies the necessity to have and maintain a high-quality documentation. This is true for any system and software products, but especially this becomes critical for the OSS products, where documentation is also used for communication [6] between distributed stakeholders.

There exist a number of quality models (e.g., ISO-9126 [13], McCall [19], Boehm *et al* [1]) addressing software *product* criteria like maintainability, usability, and reliability. Other models (e.g., CMM [20], SPICE [5]) consider software *process* criteria like maturity or capability with the underlying hypothesis that good process would result in a good product. Although it is recognised that a *high-quality documentation* is necessary to different stakeholders (users, developers and acquirers) [15], little is done to consider the quality criteria of the documentation itself.

A proper assessment of the software documentation is necessary for several reasons. Firstly, it is a part of the overall product assessment at the decision making process during product acquisition. Secondly, assessing documentation can help a choice between several alternatives. Thirdly, assessment results might help developers to improve the documentation quality *per se*. Documentation quality assessment is also important in agile approaches, which apparently contradict the absolute necessity of extensive documentation and focus on delivering the workable software products as the main measure of software process progress. Even in such cases, software product and process quality depends on a minimal documentation while the inherent rule is to avoid unnecessary documentation. Therefore, a quality approach necessitates some means (i.e., some precise criteria) to evaluate whether the documentation is really adequate or too excessive with respect to the context of use.

To achieve a documentation assessment, one needs a comprehensive quality model comprising precise guidelines and criteria which are (*i*) *consensually admitted* by the different stakeholders as reflecting their perception of documentation quality, (*ii*) *easy to apply*, and which produce (*iii*) quality *indicators in a repeatable and reproducible* manners. To our knowledge, there is no an established model today. Thus, our long-term research goal is to establish such a model.

In a previous work [18] we have elaborated a method to assess documentation availability (*DA*). Our method with its underlying model supports investigation of the documentation context and judges about the completeness of the information. In this paper, we make a step towards the theoretical and empirical validation of the *DA* method. As the part of the empirical validation we report on the performance test, where we have applied our *DA* method to assess documentation quality of 28 OSS projects. The findings help us to define the quality interpretation model. In this paper we report a number of observations resulting from our assessment.

The structure of the paper is as follows: in Section 2 we present the documentation availability model and estimation method. In Section, 3 we discuss theoretical and empirical validity of this model. In Section 4, we continue our discussion of the empirical validity by describing the performance test, which investigates the feasibility of our proposal. Finally, Section 5 discusses the findings, presents the conclusions and future work.

2 Documentation Availability

In this section we overview the *DA* model. We present metrics, indicators, estimation method, and quality interpretation model.

2.1 Documentation Availability Model

The quality model for the OSS documentation availability is presented in [18]. It is a part of the bigger effort to develop the QualOSS[1] assessment method [3, 23]. In

[1] QualOSS stands for Quality of Open Source Software. Project is funded by European Commission under the FP6-2005-IST_5 Framework, Contract number N° 33547.

QualOSS, the OSS product is described through four dimensions: (*i*) *the software product*, defined by software code, documentation and test; (*ii*) *the community*, characterised by interconnected community members; (*iii*) *the development process*, expressed by rules that community members should follow when performing activities, and (*iv*) *the tools* used by the community to build, manage and maintain the OSS. Hence, the documentation quality is a part of the aggregated *product* quality.

For documentation quality a distinction could naturally be done between two levels (Fig. 1). Firstly one needs to identify characteristics related to the content of documentation and determining its *accuracy* with respect to what documentation is supposed to describe [21], what is the purpose of documentation, and what stakeholders' behaviour the documentation might lead. Secondly, one needs to investigate characteristics related to the form and determining *completeness and availability* of different structural parts [4]. Although theoretically both aspects are important, assessing the first one appears to be difficult as long as we admit that the documentation could be written manually and/or in a non-formal style. In this work we focus on the second aspect, thus, we develop a systematic method [18] to assess *documentation availability* (*DA*) using two indicators (Fig. 1): *documentation type availability* and *documentation information availability.*

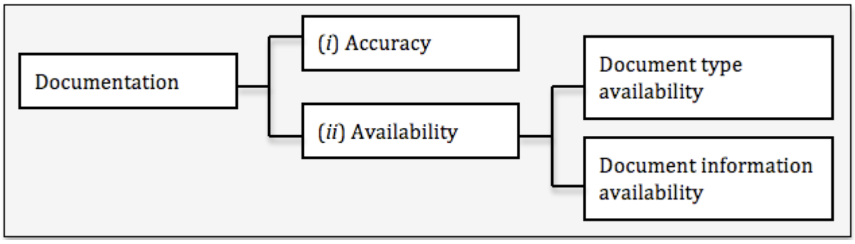

Fig. 1. The Documentation Assessment Model

2.2 Documentation Type Availability

In [18] we define thirteen document types and classified them to four categories: presentation documents, product installation and application documents, documents of product development process, and management and copyright documents. *Document type availability* (*DTA*) characterises availability of documents belonging to a certain document type:

$$DTA = \frac{DF}{DN} \tag{1}$$

here, *DN* is the number of considered document types (i.e., chosen from the set of 13 identified types), *DF* is a number of documents types for which documents are found.

This indicator tells that there should exist a set of *documents* allowing the *stakeholders* to achieve their goals. We indicate that a single stakeholder can play four different roles: *product acquirer* (interested in presentation documents), *product user* (concerned by the product installation and application document), *product developer* (interested in documents of product development) and *product contractors* (concerned by management and copyright document). Depending on the stakeholder's goal, different documents would be evaluated. For example, to achieve the goal of product acquirer, one needs to consider presentation documents; to reach the goal of product user, one needs to evaluate availability of product installation and application documents. In this paper we will focus on the overall documentation assessment taking all four stakeholders' goals together.

It is very important to know the *location* of these documents. It might be situations when a document actually exists, but the stakeholder is not able to find where this document is stored. This means that the document is still not available for the stakeholder.

2.3 Documentation Information Availability

Document information availability (*DIA*) indicates whether the documents contain *organised* and *complete information*, presented at the *complete level of detail*:

$$DIA = \frac{\sum_{i=1}^{DN}(dor_i + dco_i)}{2DN} \tag{2}$$

here dor_i – is an aggregated metric representing the organisation of documents belonging to document type i; dco_i – is an aggregated metric representing the completeness of documents belonging to document type i; DN – is the number of considered document types.

Document organisation can be estimated as easiness to locate information and logical relationships among adjacent chapters, sections and subsections [18]. Hence we defined a number of simple "Yes/No" questions (e.g., *Is there a table of content in the document?*; *Is the document divided to chapters?*, etc) that help estimate document organisation. The value is calculated as a ratio between the number of positively answered questions and the number of considered questions (recall that some questions could be non applicable and should be disregarded):

$$dor = \frac{\sum_{i=1}^{N} r_i}{N} \tag{3}$$

here $\{r_1 \ldots r_N\}$ are answers ("Yes" ≡ 1, "No"≡ 0) to questions about document organisation, N – number of questions having answers "Yes" or "No", and *dor* – organisation of a single document.

A document is *complete* with respect to its content (*content completeness*) if it contains all the information necessary for the concerned stakeholder to satisfy his/her goal(s). Following software development standards [7, 8, 9, 10, 11, 12] we have developed a number of *content templates* describing the content structure of different document types [18]. When analysing a single document, an evaluator checks if this document contains the *information units* (sections, subsections, and paragraphs) as suggested in these templates.

Document is said to be at the *complete* level of detail (*information completeness*) if it defines *all* information units at the *high* level of detail. Information units, identified when analysing content completeness, here, are considered for their information completeness. The *level of detail* is considered on an ordinal scale of four values ("null" ≡ 0, "low" ≡ 1, "average" ≡ 2, and "high" ≡ 3).

Document completeness is an aggregated measure of document content completeness and document information completeness [18]. It is expressed as the sum of multiplications between content completeness and information completeness, divided by the tripled number of analysed information units:

$$dco = \frac{\sum_{i=1}^{M} c_i d_i}{3M} \tag{4}$$

here $\{c_1 \ldots c_M\}$ are the values $\{0, 1\}$ assigned to the answers to questions about content completeness, $\{d_1 \ldots d_M\}$ are estimations of the information completeness according to the ordinal scale $\{0<1<2<3\}$, M – number of information units (questions), and dco – completeness of documents.

2.4 Documentation Availability Estimation Method

The application of the *DA* model consists of six steps (Fig. 2). Firstly, evaluator *defines the purpose and the scope* of the evaluation. This first step produces the scoped document types according to which a *search for documents* is performed in the second step. When screening for documents (step 2), one records the location of the found documents in order to access them after. The second step results in the (references to) documents which availability is under assessment. In the third step the evaluator *assigns documents* to the document types. The fourth step is the *review* of the document-document type pairs. This step should ensure that concerned documents are found and these are correctly assigned to the document types. In the fifth step, *each pair* is analysed for both the document organisation and document completeness. In the sixth step the evaluator computes the *DTA* and *DIA* indicators. As discussed above, *DTA* is estimated (equation 1) according to the number of found documents (resulting from step 4) and the number of considered document types (decided in step 1); *DIA* is computed (equation 2) from the document organisation (equation 3) and document completeness (equation 4) estimated in step 5.

2.5 Indicator Interpretation

Both *DTA* and *DIA* indicator values are estimated on the percentage interval (0 to 100% of availability), higher value means the documentation availability is higher. Following the guidelines of the QualOSS project [3, 23] we defined the indicator interpretation model (Table 1) both at the interval and at the ordinal scales (by mapping the former one to the latter one) [18]. The reason to define interpretation model at two scales is the simplicity to classify things into few categories. For the targeted audience it is much simpler to understand a given category (e.g. Yellow), than to judge about some number, received after calculation is done.

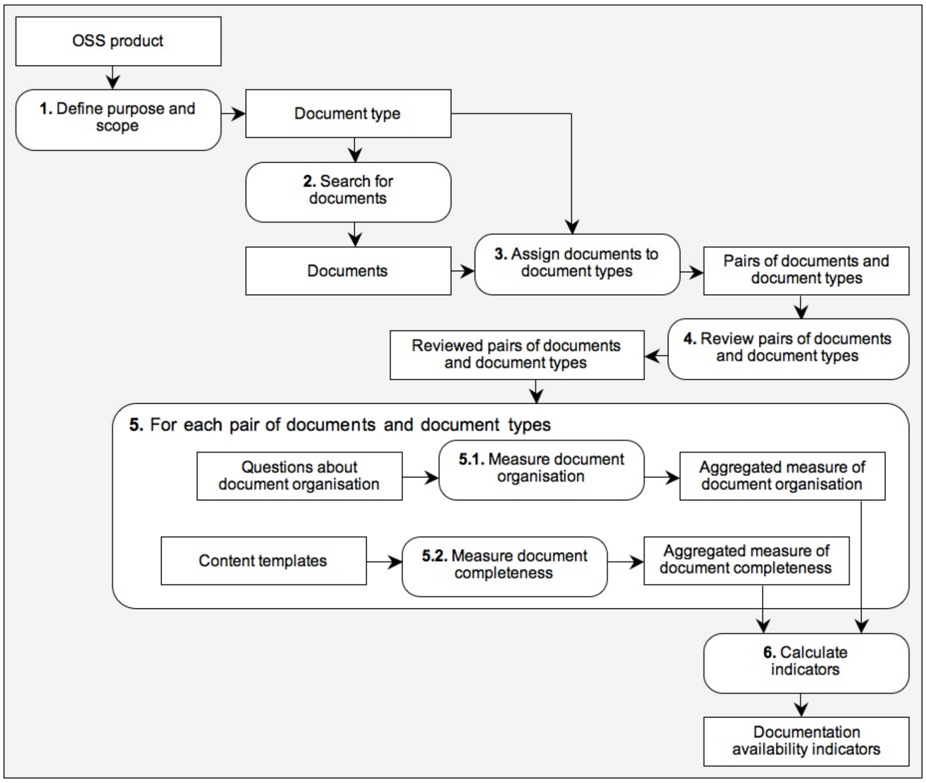
OSS product
1. Define purpose and scope
Document type
2. Search for documents
Documents
3. Assign documents to document types
Pairs of documents and document types
4. Review pairs of documents and document types
Reviewed pairs of documents and document types
5. For each pair of documents and document types
Questions about document organisation
5.1. Measure document organisation
Aggregated measure of document organisation
Content templates
5.2. Measure document completeness
Aggregated measure of document completeness
6. Calculate indicators
Documentation availability indicators

Fig. 2. Documentation Availability Estimation Method

Table 1. Interpretation of *DA* Indicators

Interval scale (%)	Ordinal scale (colour)	Explanation
[0 … 9,99]	*Black*	Not available
[10 … 39,99]	*Red*	*DA* is limited
[40 … 69,99]	*Yellow*	*DA* is average
[70 … 100]	*Green*	*DA* is high

However this interpretation model has several limitations. Firstly, its definition (e.g., thresholds and their suggested explanation) is highly subjective, provided without extensive testing of indicators on the OSS assessments. Secondly, the two indicators cannot be interpreted at the same scale, because their natures, measurement, and estimation inputs are different as defined in the above sections. Finally, although being applied to assess few OSS projects [18], the *DA* method itself was not yet applied to a sufficiently large extend. Thus, its validity needs to be investigated at a more coarse-grained level.

The main goal of this paper is twofold. Firstly, we need to validate the *DA* model [18]. Secondly, we need to establish an empirically grounded *interpretation model* for the *DA* indicators.

3 Validity of the Documentation Availability Model

In order for a quality model to be valid, all its metrics (including aggregated metrics and indicators) have to be valid. For a metric to be valid, the following two conditions must hold [16]: (*i*) this metric must not violate any necessary properties of its elements, and (*ii*) each model used in the measurement process must be valid. In other words to validate the *DA* method we need to show validity of

- the metrics (e.g., *chapter*, *section* used to gather data about document completeness; *information units* used to gather date on document organisation), aggregated metrics (e.g., *document organisation* (see equation 3) and *document completeness* (see equation 4)), and indicators (e.g., *DTA* and *DIA*);
- the measurement scales; and
- the estimation method, presented in Section 2.4.

Kitchenham *et al.* [16] define two major methods to check metrics validity (Fig. 3):

- *Theoretical validation*, which confirms that the measurement does not violate any required properties of measurement elements or of the definition models [16].
- *Empirical validation*, which corroborates that measured attributes are consistent with the values predicted by the models involving the attribute [16].

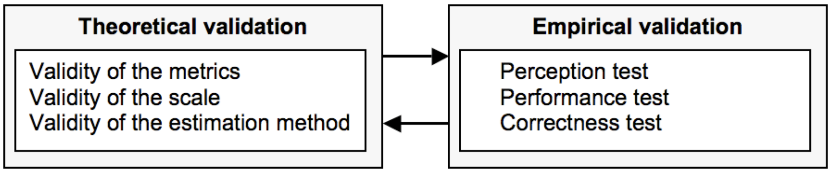

Fig. 3. Theoretical and Empirical Validation

3.1 Theoretical Validation of the Documentation Availability Model

Theoretical methods of validation allow us to say that a metric is valid to a certain defined criteria. Such a list of criteria is presented in [16]. We present them hereafter together with theoretical validity of our *DA* model.

Validity of the metrics. Four criteria are defined for the metric validity:

- *For the metric to be measurable it must allow different entities to be differentiated from each other* [16]. By definition [18] (also see Section 2.2) the *DA* metrics allow distinguishing different documents according to their document types;
- *A valid metric must obey the representation condition* [16]. For instance, the *DA* metrics preserve the intuitive notions about the document type's property and the way in which document type properties distinguish between document types.
- *All scales contributing to a valid metric are equivalent* [16]. It is possible to define different valid scales for the *DA* metrics. For instance, in Section 2 we define information units that are measured in the scale of [“null” ≡ 0, “low” ≡ 1, “average” ≡ 2, and “high” ≡ 3]. However this sale could be extended with intermediate values

(e.g., [“null” ≡ 0, “low” ≡ 1, “somehow low” ≡ 1.5, “average” ≡ 2, “somehow high” ≡ 2.5, and “high” ≡ 3]). Hence both scales contribute to a valid measure.

- *Different entities can have the same metric value* [16]. Applying the *DA* model, we estimate document organisation using the same set of metrics. These metrics can receive the same value for different documents of different document types. Thus, we say that the *document organisation* metrics satisfy all this validity criteria.

Validity of the aggregated metrics and indicators. The aggregated metrics (e.g., *document organisation* and *document completeness*) and indicators (e.g., *DTA* and *DIA*) are valid theoretically because:

- They are based on a model (e.g., *content templates* when calculating *documentation completeness*) concerning the relationships among document properties.
- They are based on a dimensionally consistent model (as defined in [16]), estimating the aggregated metrics and indicators on the percentage scale.
- The discontinuities are not possible because metric values are received by normalising (always larger or equal) numbers of the considered entities (e.g., for the *DTA* the number of documents found is always either equal or lower to the number of considered document types).
- They include the correct measurement scales, characterised as interval.

Validity of the documentation availability estimation method (or measurement protocol as called in [16]) is usually validated by peer acceptance. As mentioned in Section 2, the *DA* model and the estimation method are the parts of larger effort to construct a standard QualOSS assessment method. The *DA* estimation method was reviewed by the project partners several time [23]. It has also been tested in few minor OSS assessments [18]. Our experience shows that the *DA* estimation method is unambiguous, self-confident, and prevents the problems such as double counting.

Our theoretical validation corresponds to the application of [16] reported in [17]. But we also admit that the full consensus could be reach only through the iteration of discussions with the community (in this case, partners of the QualOSS project).

3.2 Empirical Validation of the Documentation Availability Model

To corroborate a metric empirically, one needs to perform experiments to show whether people agree that a measurable property exists and whether a mapping to value captures the understanding of the property [16]. In general, empirical validation could be performed through perception, performance and correctness tests.

The *perception test* involves investigation of the artefact usability, ease of use, and user satisfaction. Currently, this approach is not applicable for our *DA* model, because the model is a novel proposal and there is no sufficient experience to report on its perception on a large extend. The *performance* test describes the application of the artefact to see its feasibility. In this work we apply the performance test and we assess documentation availability of 28 of OSS projects. We report our experience in Section 4. Finally, currently we are designing the *correctness* test. We plan to contact the OSS community members in order to receive their feedback on the results received during the performance test. We hope to show that the performance of the *DA* model is indeed correct and corresponds to the perception of the experts.

4 Performance Test

In this section we present the first step towards the empirical validation of the *DA* model: a performance test to investigate our model feasibility. In Section 4.1 we describe its performance test design. Section 4.2 discusses validity threats. Section 4.3 presents the major results. Finally, in Section 4.4, we will refine the *DA* interpretation model on basis of the performance test results.

4.1 Design

As discussed in Section 2, one of our research goals is also to define an empirically grounded interpretation model for the documentation quality. We formulate the following **research question**:

How do the current indicators help to understand and interpret quality of the open source documentation availability?

Firstly, this research question challenges the application of the *DA* method to assess the *DA* of the OSS products. This shows the *DA* method validation through the performance test, as discussed in Section 3.2. Secondly, we will investigate whether the indicator values received for the sample of OSS projects cover the whole range of the ordinal scale defined for these indicators. If it is not the case, based on the assessment results we will refine the initial interpretation model (Table 1).

The **research method** is pretty straightforward. First, we formulated the research question presented above. Then, we defined a sample of the OSS projects. Next, we applied our *DA* model and calculated the values of *DTA* and *DIA* for each selected project. Finally, we analysed and interpreted the received results.

The **assessment sample** includes 28 OSS projects (see Table 2). We selected them based on the OSS survey provided in [14]. The selected OSS projects cover different Information System application domains, such as content management systems (e.g., *Plone*, *JetSpeed*, and *Jakarta Structs*), email systems, calendar systems, and address-book systems (e.g., *Thunderbird*, *Evolution*, and *Sup*), text viewing/editing systems (e.g., *Open Office writer*, *xEmacs*, *Evince*, and *xPDF*), Web application support systems (e.g., *Zope*, *Httpd*, *jMeter*, and *Galeon*), operating and their management systems (e.g., *FreeBSD*, *NetBSD*, and *Nautilus*), programming languages and their environments (e.g., *Python*, *Eclipse* platform, *gcc backend*, and *Findbugs*).

4.2 Threats to Validity

Before presenting the performance test results, we discuss some validity threats [24]:

- Reliability of the *QA* model could be seen as the *internal validity threat*. However, as we illustrated in Section 3.1 our proposal is theoretically valid. In addition it was developed systematically based on existing system and software development standards [7, 8, 9, 10, 11, 12]. Based on the assumption that the used system and software development standards (that provide also guidelines for the documentation preparation) also apply to the documentation of the OSS products, the *QA* model is systematically developed as illustrated in [18].

- Reliability of the assessment data on the 28 OSS projects could be found as the *conclusion validity* threat. Collecting of these data (step 5, Fig. 2) is a manual task, thus, it contains a certain degree of subjectivity. To mitigate this threat the assessment was always executed by at least two assessors. The first assessor was taking the actual metrics on the OSS documents. The purpose of the second assessor was to monitor and to review the assessment results. For some projects the assessment step (step 5, Fig. 2) was iterated few time in order to ensure the assessment quality and to decrease the result subjectivity.
- The majority of the OSS assessments was done by the same assessor (who was not the author of the *DA* model). Thus, there might be the *internal validity threat of maturation* [24], meaning that over the time the assessor learned how to take measurement and this resulted in the better/worse scores for the later projects. To mitigate this risk the measurement was monitored by a second evaluator who is one of the authors of the *DA* model.
- The internal validity might be influenced by the selection of the OSS projects for the evaluation. Our sample was selected according to the results of the larger survey [4]. This means that we selected the projects that are already quite popular either when developing Information Systems or when working with them.
- A possible threat to external validity is the nature of our experiment. We did not wish to select any OSS for an actual practical application (e.g., to support real decision). Here, we are only interested in testing the feasibility of the *DA* model.

4.3 Results

Table 2 presents the assessment results. Following the interpretation model presented in Section 2, the documentation type availability is high (*green*) for 19 projects, and it is average (*yellow*) for 9 projects. The documentation information availability is average (*yellow*) for 11 projects and limited (*red*) for 17 projects. Those assessment results confirm that:

- The range of each interpretation category is too broad. For example *DTA* interpretation is *green* both in the case when documents were not found for 4 document types (score 76.92%), and also when documents for all document types are found (score 100%). Similar situation can also be observed for the *DIA* indicator: the difference between the best and the worst evaluated projects at the *red* category is relatively large (e.g., *DIA* of *gcc* is *39.76%*, *DIA* of *Omitux* is *15.56%*, difference is *24.2%*)
- The full range of the assessment is not covered for neither of the indicators. For instance none documentation was interpreted as limited (*red*) or as unavailable (*black*) for the *DTA* indicator; none documentation was found of high (*green*) availability, nor unavailable (*black*) with respect to the *DIA* indicator.

To mitigate these limitations of the interpretation model we refined it. The major idea is to assign to each category of the ordinal scale a sufficient number of values from the internal scale. In addition we observed (specifically for the *DIA* indicator) that it is possible to define four groups of data that is relatively close to each other.

Table 2. Documentation Assessment Results

OSS projects	**Interval scale**		**Ordinal scale**	
	DTA, %	**DIA,** %	**DTA,** colour	**DIA,** colour
Python	92.31	52.45	Green	Yellow
PLONE	84.62	52.08	Green	Yellow
Thunderbird (Mozilla)	100	50.46	Green	Yellow
Zope	84.62	49.47	Green	Yellow
FreeBSD	84.62	47.31	Green	Yellow
PhPMyAdmin	76.92	46.56	Green	Yellow
NetBSD	84.62	43.25	Green	Yellow
Eclipse	84.62	42.98	Green	Yellow
Writer (OpenOffice)	76.92	42.20	Green	Yellow
Hadoop	100	41.72	Green	Yellow
Xemacs	100	41.28	Green	Yellow
Gcc	76.92	39.76	Green	Red
VLC	76.92	38.74	Green	Red
HTTPD1.3	69.23	34.95	Yellow	Red
Jetspeed	84.62	34.83	Green	Red
Jakarta Struts	76.92	34.27	Green	Red
Evolution	92.31	32.72	Green	Red
Jmeter	92.31	32.08	Green	Red
k3b	69.23	31.52	Yellow	Red
Evince	84.62	30.85	Green	Red
xPDF	69.23	28.96	Yellow	Red
Findbugs	69.23	27.62	Yellow	Red
Nautilus	76.92	26.98	Green	Red
CVSanaly	69.23	24.54	Yellow	Red
Yanolc	46.15	18.41	Yellow	Red
Galeon	61.54	17.01	Yellow	Red
Sup	61.54	16.15	Yellow	Red
Omnitux	53.85	15.56	Yellow	Red

The refined interpretation model is shown in Table 3. Here the mapping between interval scale and ordinal scale is different for both indicators, and, thus, corresponding to our empirical findings. In Table 4 we present a matrix that define correspondence between two *DA* indicators based on the refined interpretation model.

Table 3. Refined Interpretation Model for the *DA* Indicators

Indicators	**Interval scale (%)**	**Ordinal scale (colour)**	**Explanation**
Documentation type availability	[0,00 … 57,99]	*Black*	Not available
	[58,00 … 72,49]	*Red*	*DA* is limited
	[72,50 … 88,49]	*Yellow*	*DA* is average
	[88,50 … 100]	*Green*	*DA* is high
Documentation information availability	[0,00 … 21,49]	*Black*	Is not available
	[21,50 … 36,99]	*Red*	*DA* is limited
	[37,00 … 44,99]	*Yellow*	*DA* is average
	[45,00 … 100]	*Green*	*DA* is high

Table 4. Matrix of the OSS Project Assessment

Indicators		Documentation information availability (*DIA*)			
		Black	**Red**	**Yellow**	**Green**
Documentation type availability (*DTA*)	**Green**	-	*Evolution* and *jMmeter*	*Hadoop* and *Xemacs*	*Python* and *Thunderbird*
	Yellow	-	*Jetspeed, Jakarta Struts, Evince, Nautilus*	*NetBSD, Eclipse, Writer, gcc* and *VLC*	*PLONE, Zope, FreeBSD* and *PhPMyAdmin*
	Red	*Galeon* and *Sup*	*HTTPD1.3, k3b, xPDF, Findbugs* and *CVSanalY*	-	-
	Black	*Yanolc* and *Omnitux*	-	-	-

"-" means that sample project is not found.

5 Discussion and Conclusions

In this section we discuss our results and present conclusions. We also situate our proposal into the state of the art, and provide some future research directions.

5.1 Conclusions

In this paper we presented a quality model to assess documentation availability for the OSS products. Our major discussion includes the analysis of the validity both at theoretical and at empirical level. Firstly, following the criteria defined in [16] we confirm the theoretical validity of the *DA* model, including its metrics, metric scales, and quality estimation process.

Next, we investigate the empirical validity through the performance test that helps to assess feasibility of our proposal. Our work results in the refinement of the interpretation model for the documentation availability indicators, as shown in Table 3. The new interpretation model is based on the empirical results and is more flexible to judge about the quality of the documentation itself: for example, using the refined model one gets the assessment data that are clustered into nine categories (using previous model it was separated to only three). This expands the assessment scale and gives more flexibility to judge about the documentation quality, especially, when the assessment purpose is selection and acquisition of OSS software to practice.

Although in the performance test we did not have as goal to select the best-documented OSS projects, the results points out few important suggestions for the OSS documentation improvement. For instance, availability of documents of a certain type does not guarantee availability of the information within these documents (e. g., *Evolution* and *jMeter*). This means that, although there are many documents for these OSS projects, those documents do not contain a sufficient level of guidelines, help, instructions, and etc. for the stakeholders.

We can also notice an opposite trend: some documents contain a lot of useful information, where the overall project documentation availability might be of a lower degree (e.g., *PLONE, Zope, FreeBSD* and *PhPMyAdmin*). This means that the existing project documentation is of high quality to satisfy goals only for a limited group of stakeholders (for instance, there exist high-quality product installation and application documents to satisfy goals of product acquirers, but there is no product development documentation to fulfil goals of product developers).

A limitation of the refined interpretation model is that there is missing sample of projects that would illustrate its full feasibility (some of the cells in Table 3 are still empty). This does not mean that there is no such sample of project documentation; it might be the case that we did not happen to select such.

5.2 Related Work

Software development standards [7, 8, 9, 10, 11, 12] suggest documentation templates for different development stages. These templates include structural guidance to prepare documentation, however the standards do not explain how to assess quality of the documentation. Following the software development standards, the *DA* method suggests the guidelines on how to assess the structural completeness of documentation. Later, based on the structural completeness results, the information completeness is investigated.

Literature reports on few proposals to measure documentation quality. Davis *et al.* [2] define a set of qualitative characteristics (e.g., non-ambiguity, completeness, correctness, consistency, verifiability, traceability, modifiability, and others) to measure quality of requirements specifications. Further in [4] a comprehensive metrics, like page count, readability, pages per day and software dependent metrics, are proposed. However both works do not go beyond the metric definition phase. Elsewhere in [22], a process to assess documentation content quality is defined. The documentation content is assessed through the following characteristics: ownership, readability, accuracy, thoroughness, format, accessibility, currency, effectiveness, accountability, and ease of update. The assessment is summarised into a quality/value matrix comprising four subjective values: poor, fair, good, and excellent.

The *DA* method is specifically dedicated to assess the documentation availability. Similarly to [2] it relies on the qualitative characteristics, and analyses organisation and structure of the documentation like in [4]. During measurement the evaluators need to deal with the certain degree of subjectivity. Differently from the mentioned works, the *DA* method provides a systematic and comprehensive process (Fig. 1) to assess documentation of different document types and to fulfil goals of various stakeholders. In addition to all these works the *DA* method also provides the indicators and explains how it is possible to interpret these indicators.

5.3 Future Work

In this paper we have described the performance test to investigate the feasibility of the *DA* model. This is the first step towards a model empirical validation. The next step is the correctness test, where we are planning to perform an interview of community members on the documentation quality of the selected OSS projects. This would allow us to compare two sets of the data for the documentation availability and will answer the question about the empirical validity of the *DA* model.

Although we have developed the availability assessment model for the documentation quality of the OSS product, our future plan is to adapt it for documentation assessment of any type of systems. This is possible because we have based our model on the standards of the software development, like [7, 8, 9, 10, 11, 12]. The major difference for the measurement process stands in the definition of the measurement goals and scope. For example for the general software documentation assessment, the goal of becoming a member of the community might not be applicable. However other goals – like obtaining the products (especially for the commercial-of-the-shelf product selection), using the (general) product, or improving the current product – are often within the scope of the general system assessment.

Finally our long-term research plan includes expansion of quality model (Fig. 1) with the *accuracy* measurement. This will require understanding of the various metrics for documentation readability, understandability, consistency and others [2].

Acknowledgments. We would like to thank Judith Chambou Simo for her contribution to this research. We also thank the partners of the QualOSS project for the stimulating discussions on the OSS quality.

References

1. Boehm, B., Brown, J.R., Kaspar, J.R., Lipow, M., MacLoed, G.J., Merritt, M.J.: Characteristics of Software Quality, TRW Series of Software Technology. North-Holland Pub., Amsterdam (1978)
2. Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledeboer, G., Reynolds, P., Srimani, P., Ta, A., Theofanos, M.: Identifying and Measuring Quality in a Software Requirements Specification. In: Proceeding of the 1st International Software Metrics Symposium, pp. 141–152 (1993)
3. Deprez, J.-C., Haaland, K., Kamseu, F.: QualOSS Methodology and QualOSS Assessment Method, Deliverable D4.1, `http://www.qualoss.org/deliverables` (last checked 28.02.2010)
4. Le Vie Jr., D.S.: Documentation Metrics: What Do You Really Want to Measure? `http://www.stc.org/intercom/PDFs/2000/200012_06-09.pdf` (last checked 28.02.2010)
5. El Emam, K., Drouin, J.-N., Melo, W.: SPICE. In: The Theory and Practice and Software Process Improvement and Capability Determination. IEEE Computer Society, Los Alamitos (1998)
6. Forward, J.A., Lethbridge, T.C.: Qualities of Relevant Software Documentation: an Industrial Study, `http://www.site.uottawa.ca/~tcl/gradtheses/aforward/papers/aforward_icse2003_sub.pdf` (last checked 28.02.2010)
7. IEEE: IEEE Recommended Practice for Software Design Descriptions, IEEE Std 1016-1998 (1998)
8. IEEE: IEEE Standard for Software Maintenance, IEEE Std 1219-1998 (1998)
9. IEEE: IEEE Recommended Practice for Software Requirements Specification, IEEE Std 830-1998 (1998)
10. IEEE: IEEE Standard for Software Project Management Plans, IEEE Std 1058-1998 (1998)
11. IEEE: IEEE Standard for Software Test Documentation, IEEE Std 829-1998 (1998)
12. IEEE: IEEE Standard for Software User Documentation, IEEE Std 1063-2001 (2001)

13. ISO/IEC: Information Technology – Software Product Evaluation– Quality Characteristics and Guide Lines for their Use. ISO/IEC IS 9126, Switzerland (1991)
14. Izquierdo, D., Herraiz, I.: Qualoss 3.1 measurement Targets, Deliverable D3.2, `http://www.qualoss.org/deliverables` (last checked 28.02.2010)
15. Jazzar, A., Scacchi, W.: Understanding the Requirements for Information System Documentation: an Empirical Investigation. In: Proceedings of the ACM Conference on Organizational Computing Systems (COOCS 1995), pp. 268–279. ACM, New York (1995)
16. Kitchenham, B., Pfeeger, S.L., Fenton, N.: Towards a Framework for Software Measurement Validation. IEEE Trans. on Soft. Eng. 21(12) (1995)
17. Loconsole, A., Borstler, J.: Theoretical Validation and Case Study of Requirements Management Measures. Technical report UMINF-03.02, Department of Computing Science, Umea University (2003)
18. Matulevičius, R., Kamseu, F., Habra, N.: Measuring Open Source Documentation Availability. In: Proceedings of the 12th International Conference on Quality Engineering in Software Technology (CONQUEST 2009), dpunkt.verlag GmbH, pp. 83–102 (2009)
19. McCall, J.A., Richards, P.K., Walters, G.F.: Factors in Software Quality, RADC TR-77-369, Vols I, II, III, US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055 (1977)
20. Paulk, M.C., Curtis, B., Chrissis, M., Weber, C.: Capability Maturity Model for Software: Version 1.1. Technical Report SEI-93-TR-24, Software Engineering Institute, Carnegie Mellon University (1993)
21. Singh, N.: Maintaining Quality Control in Documentation. In: Proceedings of Society for Technical Communication (2002)
22. Schiesser, R.: How does your process documentation measure up?, `http://articles.techrepublic.com.com/5100-10878_11-1053164.html` (last checked 28.02.2010)
23. Soto, M., Ciolkowski, M., Deprez, J.-C., Ruiz, J., Herraiz, I., Campos, C.G., Matulevičius, R.: Metrics and Indicators of the Standard QualOSS Assessment Method, Deliverable D4.2, `http://www.qualoss.org/deliverables` (last checked 28.02.2010)
24. Wohlin, C., Runeson, P., Høst, M., Ohlsson, M.C., Regnell, B., Wesslen, A.: Experimentation in Software Engineering. Kluwer Academic Publishers, Boston (2002)