U N I V E R S I T Y OF T A R T U
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

Kaarel Tark

A Revision of the Documentation Availability Model for Open Source Software

Bachelor's thesis

Supervisor: Raimundas Matulevičius

Author: .... "....." 2011
Supervisor: .... "....." 2011
Allow to defence
Professor: .... "....." 2011

TARTU 2011

Table of content

Table of Figures

List of Tables

Introduction

Nowadays a lot of information systems (IS) rely partly or fully on Open Source Software (OSS). Thus it becomes a necessity to strive for the high quality tools in order to have a proper means to support the business processes. The quality of a project deliverable can be examined from different aspects: quality of code, quality of tool and quality of documentation. We focus on the quality of documentation.

There exist different approaches to assess quality of the software products. For instance evaluations of quality [13] could be performed (*i*) using detailed qualitative properties or (*ii*) through general quality frameworks. A systematic survey of these approaches could be found in [12]. Matulevičius *et al.* [10,11] have proposed a quality model to assess quality of documentation availability (DA). This model is based on the existing software development standards [3,4,5,6,7,8] and includes 12 different documentation completeness templates. Previously the *DA model* has been applied to evaluate 24 OSS products. The information from the previous research shows that some of the templates have a lot of entries that are not used when evaluating the OSS documentation. Additionally there might be some entries that are needed to understand the OSS documentation, but these are not yet included into the current *DA model*. We observe that there exist a number of limitations when applying it to assess the OSS projects. Thus we construct a *DA model* by revising the existing model's completeness templates from the perspective of the OSS documentation; *i.e.*, we revise document types, separate document type's entries, introduce weights to document type entries, and revise the equation for the document completeness calculation. In such a way we result in a revised DA model, which is oriented to the documentation domain of the OSS products.

To validate our work we assess 14 OSS products that support the business process and enterprise modelling. The validation shows that the document completeness entries eliminations and modifications exclude the need to consider documentation completeness entries that are irrelevant for OSS.

This thesis is structured into seven chapters. In Chapter 1 we describe what is OSS, introduce the document types, standards and describe previous evaluations of the software documentation. In Chapter 2 we give a short summary of existing *DA model* [10,11] and describe its' limitations. In Chapter 3 we focus on the research method. In Chapter 4 we introduce the changes made to the previous *DA model*. In Chapter 5 we describe how the *revised DA model* is supported by the spreadsheet tool to guide the OSS documentation evaluation. In Chapter 6 we describe how we tested the *revised DA model* on the new set of the OSS products. Finally in Chapter 7 we discuss the threats to validity, highlight the advantages and possible disadvantages of the *revised DA model*, and introduce the future work.

We also include a list of appendixes where we provide the revised templates and weights, tools' websites and a CD where there are supporting spreadsheet template, unrevised *DA model* templates, *Document organisation* template, analysis table respect to previous evaluation and the new evaluation results.

.

CHAPTER 1: **Open Source Software and its Documentation**

In this chapter we give a brief introduction what is Open Source Software and what are the existing standards. We will end up with previous evaluations of software documentation.

1.1. Open Source Software Definition

The term Open Source Software (OSS) refers to software, under Open Source license [16], in which the user can access the source code, modify it and depending on the license to build a release or compile it. The software is often produced in communities which use agreed tools to develop the project: it involves writing code, writing documentation, communication, testing and fixing errors. The documentation of OSS project defines the rules of the development process and requirements of the developed tool. OSS is one of the current trends in developing software. Currently, most Information Systems (ISs) are developed in part or fully on OSS. OSS projects often provide IS development with frameworks, tools, operating systems and applications. OSS is typically free and comes with the source code needed to adapt it to the users' needs. Most open source licenses let users redistribute the software, including possible changes. They also allow these users to charge for redistribution as long as source code changes are publicly available.

Open Source Software (OSS) refers to software that consists of mainly three parts[2]: *source code*, which can be modified and compiled into a tool; the *tool*, which is the compiled source code supporting work activities, and *documentation*, which defines how the source code is written and how the tool should work.

OSS is often developed by loosely (or self-) organised communities of programming enthusiasts, communicating *via* the Internet. Anyone with an interest and some requisite degree of experience is welcome to contribute to development. Potentially hundreds of people contribute to a project, providing a diverse group of talents and techniques [15]. Therefore, in order the project to be successful, agreements on the development process, development tools, project communication and documentation are needed.

Documentation, important to all software projects, is critical for OSS projects where documentation is a part of communicating among participants in a software development project *i.e.* stakeholders (more of the stakeholders can be found in 2.2). The documentation covers everything that a new stakeholder would need in order to become a member of the project: requirements, development strategies, project management, code writing rules, committing rules, building tutorials, *etc*.

1.2. Documentation standards

Software standards describe which documents a software development process should contain. These are for example, requirements specifications [4], design descriptions [3], maintenance rules [5], user documentation [8], project management plans [6] and software test documentation [7]. The standards provide templates that can be used to prepare necessary documents. They define what kind of parts should a document consist of, but do not give a solution to the estimation of documentation quality. To overcome this limitation, in our work we select a documentation estimation model [10,11] which focuses on the documentation organisation and completeness to evaluate the documentation availability.

1.3. Previous evaluations of Software documentation

There are few studies that provide means to evaluate software documentation. Schaisser *et al.* provide quality characteristics (e.g. readability, accuracy, thoroughness, ease of update, effectiveness, *etc*) to assess document's quality [14]. A different set of quality characteristics is given by [9]: readability, pages written per day, metrics depending on the tool. They also define the documentation quality in prospect to the goal of the specific project. In that case the document's importance is defined by the community members and therefore might not be defined by the metrics needed to measure, but the metrics that are easily found.

A documentation evaluation model that current work is based on is dedicated to documentation availability [10,11]. It defines documentation quality by qualitative and organisational aspects such as "*Is it structured to chapters?*", "*Does it have a glossary?*", "*Is every method in the code commented?*". Authors look for the information units, define information completeness levels and analyse the structure of each document type. The model is described in Chapter 2.

1.4. Summary

In this chapter we introduced what is OSS and gave a brief overview of the existing documentation standards end previous evaluations. In Chapter 2 we will give a overview of the *Documentation Availability model* used in [10,11].

CHAPTER 2: **Documentation Availability Model**

DA model uses an approach to systematically evaluate documentation generated for software. It takes into account different aspects of *documentation information availability* and *document type availability*. These aspects are explained in more detail in following chapters.

Evaluation of a documents quality dwells from the reactions needed to be indicated for different stakeholders. Based on the interest of a stakeholder any documents quality can be analysed by two criteria as described in (Fig. 1)*: Accuracy* and *Availability*. In this work we focus on the *Documentation Availability*.

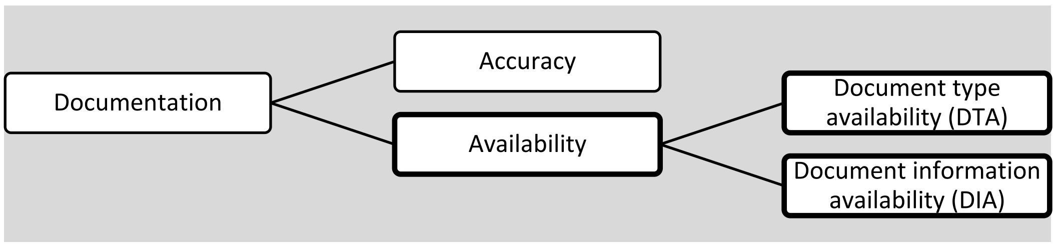

Documentation
Accuracy
Availability
Document type availability (DTA)
Document information availability (DIA)

Fig. 1 - The Documentation Assessment Model (adapted from [11])

2.1. Definition of documentation availability

Documentation availability is characterised through three aspects: *(i)* "*Is there a set of OSS documents needed for the OSS stakeholders to achieve their goals?*" *(ii) "Is the physical or electronic location of these documents is known to the OSS stakeholders?*" *(iii) "Do the documents contain organised and complete information presented at the complete level of detail?*".

2.2. Stakeholders and Document types

Stakeholders. A stakeholder can occupy four roles in the context of the project: *product acquirer*, *product user*, *product developer* and *product contractor*. The documentation quality is important to stakeholder in different respects. *The product acquirer* is a consumer whose interest is to obtain a product. For example, to achieve the goal of product acquirer, one needs to consider presentation documents. *The product user* will use the OSS product to facilitate his/her business purposes. To reach the goal of product user, one needs to evaluate availability of product installation and application documents. *The product developer* develops a new OSS product or improves functionality of an existing OSS product. To develop we need good documentation about the architecture, design and existing modules. Finally, the product *contractor* has a goal to become a member of the OSS community to influence the projects' future development directions. For a contractor we need future development and requirements documents. Depending on the stakeholder's interests, different documents need to be evaluated.

Documentation Types. The documents created for an OSS project are divided into four main groups. Depending on document's goals they are grouped as (Fig. 2): *(1) Presentation documents* are used to advertise the product, *(2) Product installation and application documents* describe installation and the support of the program. This group includes five different document types: *Installation guide, Introductory guide, Frequently asked questions, User manual guide* and *Accumulative experience notes, (3) Documents of product development process* describe the product requirements, functionalities, design, testing, development and maintenance. This group also consists of five different document types: *Requirements, Design, Implementation, Testing and Maintenance documents,* and *(4) Management and copyright documents* describe how to manage the OSS deliverables and to guarantee legacy of the OSS products. It consists of two different document types: *Management documents, Copyright documents*. All the templates can be found in the appendix A.

2.3. Document Location

To evaluate documents quality one needs to know its' location or find it first. *Document's location* is defined as the place from where it is possible to obtain OSS documentation. The documentation might be interactive or in paper format. When document exists, but it is not found by the stakeholder then the document is still counted as being not available.

2.4. Characteristics of documentation availability

A document is complete in the respect to organisation if it is divided into certain structural parts to make the information gathering from a document easy for a stakeholder. *Document organisation* is described through the easiness to find information from a document: *"does it have chapters?*", *"does it have a table of content?*", *etc*. Equation 1 is describing document's organisation level. The value is the ratio between the number of positively answered questions and the number of questions considered (full document organisation template is presented in Appendix B):

$$dor = \frac{\sum_{i=1}^{N} r_i}{N} \qquad (1)$$

Equation 1 - $\{r_1 \dots r_N\}$ 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.

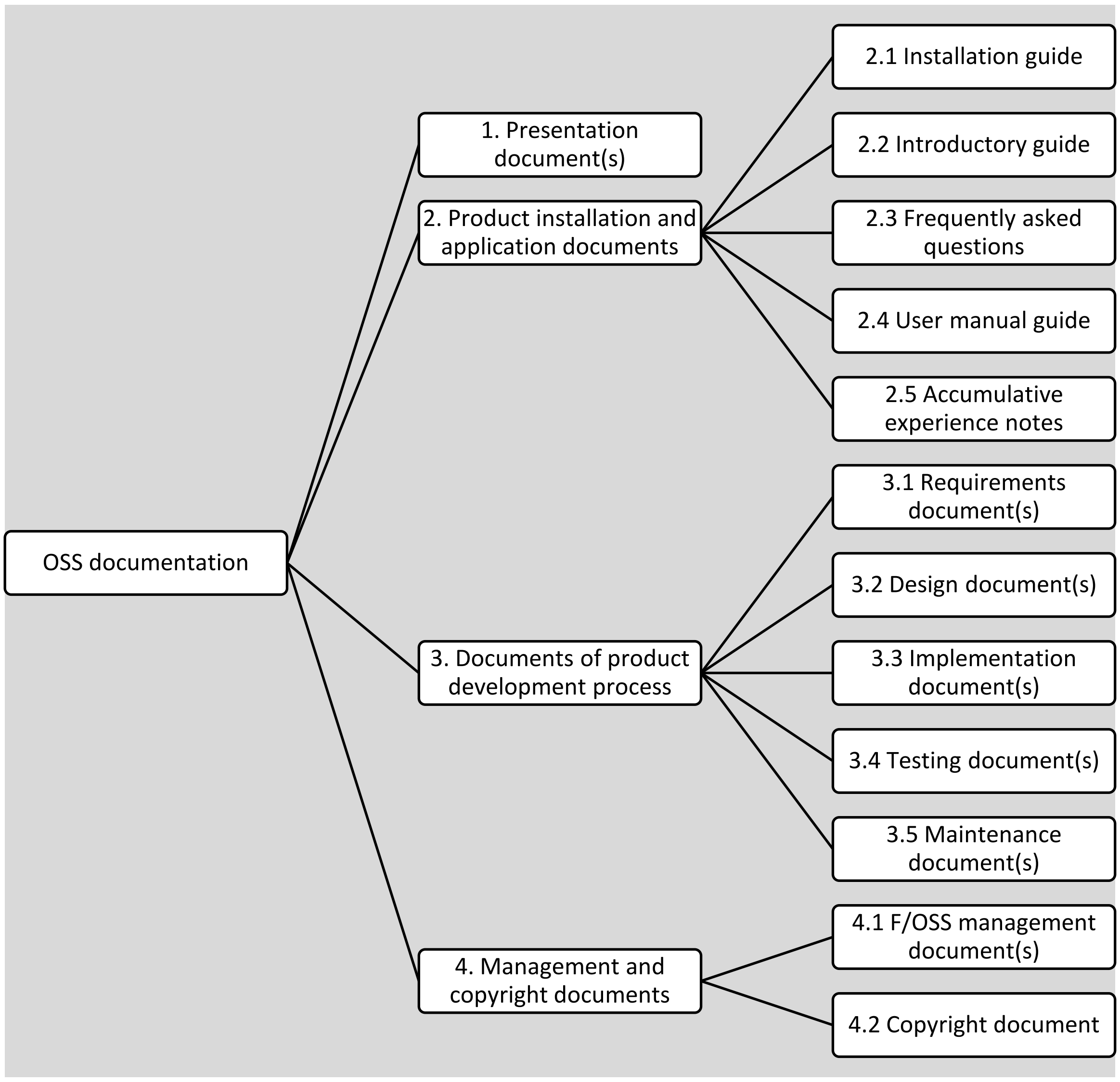

OSS documentation
1. Presentation document(s)
2. Product installation and application documents
3. Documents of product development process
4. Management and copyright documents
2.1 Installation guide
2.2 Introductory guide
2.3 Frequently asked questions
2.4 User manual guide
2.5 Accumulative experience notes
3.1 Requirements document(s)
3.2 Design document(s)
3.3 Implementation document(s)
3.4 Testing document(s)
3.5 Maintenance document(s)
4.1 F/OSS management document(s)
4.2 Copyright document

Fig. 2 - OSS Documentation groups and types

A document is complete to the respect to its' content if it fits the needs of a stakeholder to achieve his/her goal(s). To clarify and ease the evaluation each entry is given a question about document completeness: "Does the <entry> exist in the <measured document>?" In Equation 2 variable *c* describes whether this question is answered "Yes" (value 1) or "No" (value 0).

$$dco = \frac{\sum_{i=1}^{M} c_i d_i}{3M} \qquad (2)$$

Equation 2 - $\{c_1 \ldots c_M\}$ 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.

Document has the complete level of detail (*information completeness*) if it defines all information units at the high level of detail. The level of completeness is considered on an ordinary scale of 4 values (0<1<2<3). If information unit is not found in a document, it equals null. An entry has a low level of detail (value 1) if it is only mentioned in the document. An entry has an average level of detail (value 2) if it is presented and discussed in the document. Entry has high level of detail (value 3), if it is presented, discussed and illustrated with examples. The evaluation is subjective and depends on the evaluator's experience and expectation for an entry. Variable *d* in equation 2 describes the level of document *information completeness*.

Document completeness is a metric describing document content completeness and document information completeness. It is the sum of multiplications between content completeness and information completeness, divided by the tripled number of analysed information units.

2.5. DA estimation process

The application of the *DA* model consists of six steps (Fig. 3).

- First, evaluator *defines the purpose and the scope* of the evaluation.
- When searching for documents (step 2) the location of every document is recorded in order to access them later.

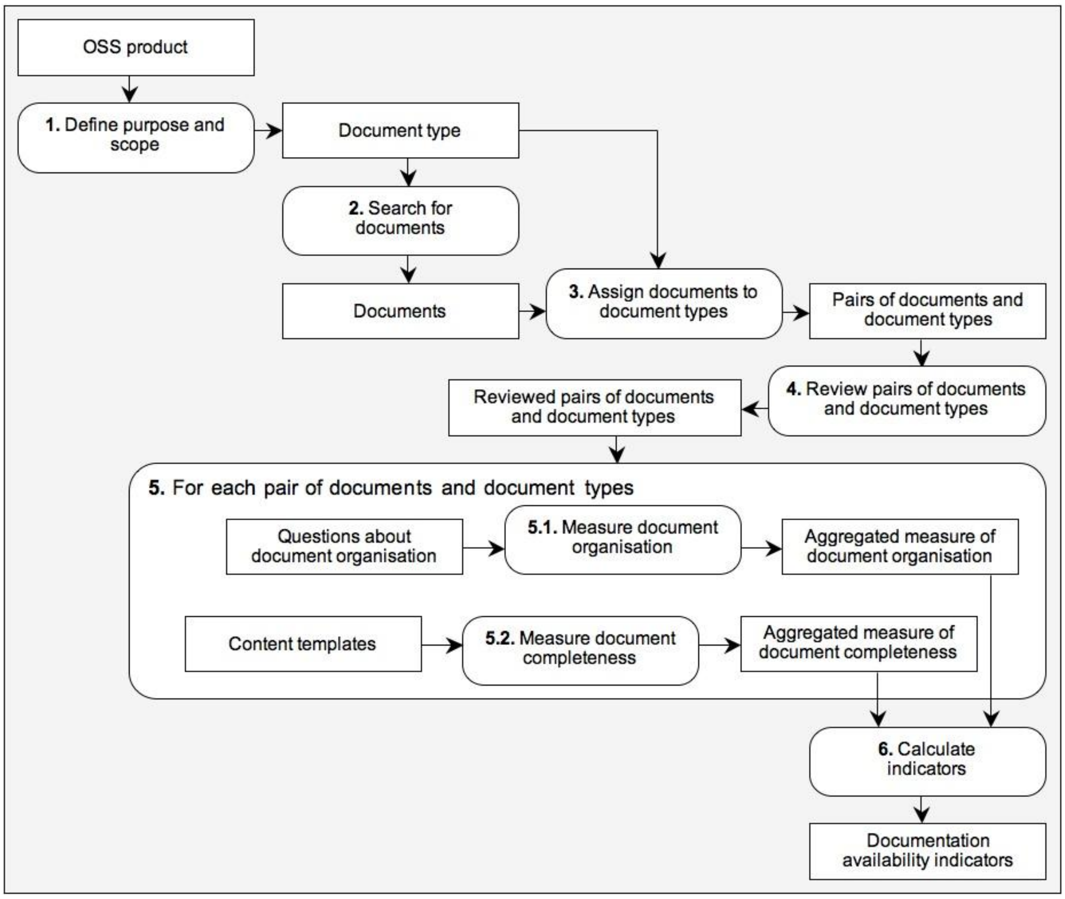

OSS product
1. Define purpose and scope
Document type
2. Search for documents
Documents
3. Assign documents to document types
Pairs of documents and document types
4. Review pairs of documents and document types
Reviewed pairs of documents and document types
5. For each pair of documents and document 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. 3 - Documentation Availability Estimation Method (adapted from [11])

- In the third step documents are divided into different document type categories.
- The fourth step the document-document type pairs are reviewed.
- In the fifth step, each pair is analysed for both the document organisation and document completeness.

- In the sixth step the *DTA* and *DIA* indicators are calculated. *DTA* is estimated (Equation 3) 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 4) from the document organisation (Equation 1) and document completeness (Equation 2) estimated in step 5.

2.6. Indicators and Interpretation

Document type availability (*DTA*) characterises availability of documents belonging to a certain document type:

$$DTA = \frac{DF}{DN} \qquad (3)$$

Equation 3 - *DN* is the number of considered document types; *DF* is a number of documents types for which documents are found.

Documentation Information Availability *(DIA)* characterizes whether the document is organised and contains complete information in a certain level:

$$DIA = \frac{\sum_{i=1}^{DN}(dor_i + dco_i)}{2DN} \qquad (4)$$

Equation 4 - dor_i – is an aggregated metric representing the easiness to find info from a document belonging to a certain 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.

DTA and DIA indicators. Both *DTA* and *DIA* indicator values are calculated on the percentage interval (from 0% to 100% of availability), higher value means the documentation availability is higher. The interpretation model is shown in Table 1. Here the mapping between interval scale and ordinal scale is different for both indicators corresponding to their empirical findings. The scale is adapted from [11] where the scales are empirically defined based on the knowledge gathered from the validation results.

Table 1 - Interpretation Model for the DA Indicators (adapted from [11])

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

2.7. Limitations

The *DA model* is based on general standards [3,4,5,6,7,8] for software development and does not take into account the peculiarity of the OSS projects. Therefore it has limitations when applying it to the OSS assessment:

- The OSS documentation is not addressed at full extent. While looking into the OSS tools we found new document types, necessary for complete documentation, that are not present in the existing model.
- Some evaluation criteria are not relevant for the OSS projects. In the previous evaluation [10,11] 67 documentation completeness entries exist that have not been evaluated.
- *DA model* does not take into account some information on OSS projects which is available and specific to the OSS product. There exist completeness entries that are specific and relevant for OSS development, but are not present in the existing model.
- The different relevance of each template entry is not taken into account as there exists weight for document types, but the relevance of each documentation completeness entry is not taken into account. This means for example that "*Unit implementation*" is treated with the same importance as "*Introduction*" in implementation document template.
- The ranges of the interpretation scales are wide. The *DIA* to be evaluated as green scale is 45% - 100% and 0% – 21,5 % to be treated as black. Only 23,5 % differs from being really low level from being high level. Fulfilling only one stakeholder's needs might change the DIA from not available to high level.

2.8. Summary

In this chapter we have introduced the existing *DA model* for software evaluation described in [10,11]. As OSS projects have characteristics that are not described in standards, this model cannot be fully used in assessing quality of documentation of OSS projects. In the next chapter we describe the research method which we applied to overcome these limitations.

CHAPTER 3: **Research Method**

In this chapter we describe the research method, which we have applied to overcome the influence of limitations of existing *DA Model* [10,11] described in the end of previous paragraph. Fig. 4 presents the process of the work with each processes outcomes. The structure of this chapter is based on the activities described on the figure.

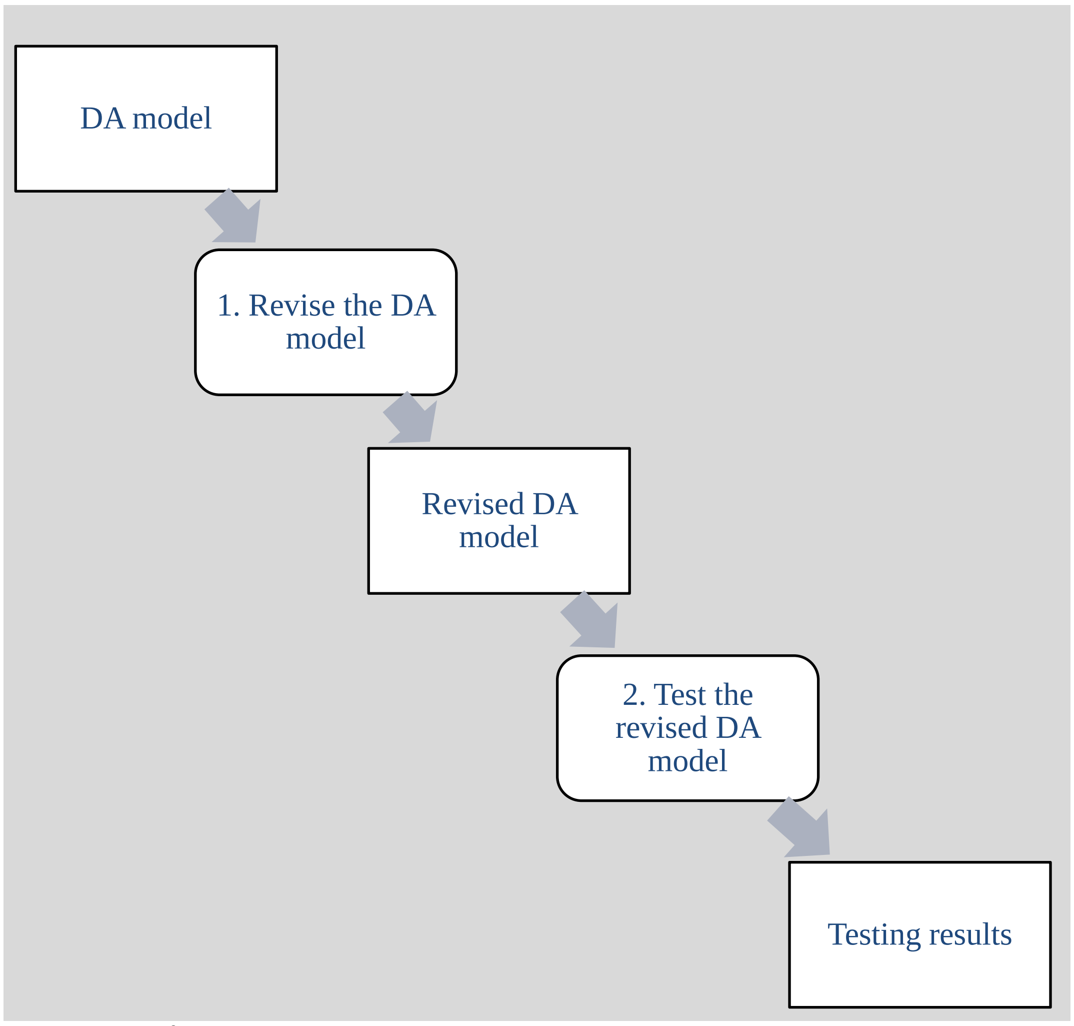


Fig. 4 - The process of revising the templates

3.1. Revising of the DA model

In this paragraph we describe the process of revising the DA model. Fig. 5 represents the process that was used during the revision and we will be referring back to each step in this paragraph. The revisions of *DA model* are described in Chapter 4.

First we focus on the document types available for OSS projects. We looked into the OSS projects (step 1) to find new document types (that did not exist in the *DA model*) that are often used and relevant for OSS documentation. Based on the analysis we defined a new document type specialised for the OSS projects. Next we worked through the existing document types (step 2) to see whether they are defined in the right level of detail and whether they need redefining to make them more relevant for OSS. Last we searched for document types that are not needed or used in OSS (step 3) to see if we can exclude them. In all those cases the decisions were based on our subjective opinion.

We analysed the existing templates based on the results of previous performance test. The results can be found in the Appendix C. We removed all the entries (step 4) that were not evaluated at all in the previous evaluation [10,11]. Then we looked for entries that had minimal evaluation and low level of information completeness. For each entry the exclusion decision was based on the results (Appendix C), information gathered from looking at the tools documentation and our subjective opinion. As there could be important entries in different document types, that were relevant for OSS and were not defined in the model, we looked at the tools and based on the information introduced new document completeness entries (step 5) that were necessary in the documentation of the OSS products.

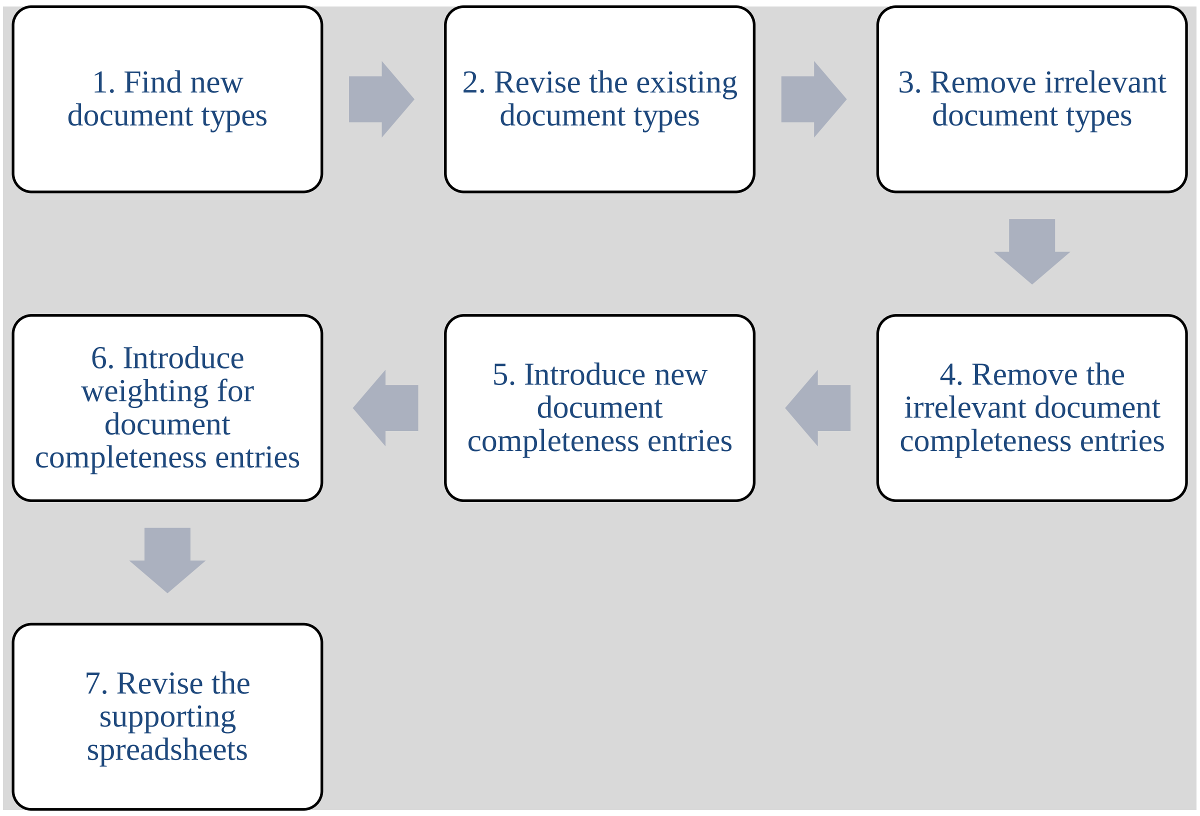

1. Find new document types
2. Revise the existing document types
3. Remove irrelevant document types
4. Remove the irrelevant document completeness entries
5. Introduce new document completeness entries
6. Introduce weighting for document completeness entries
7. Revise the supporting spreadsheets

Fig. 5 - The process of revising document types

We defined weights (step 6) for documentation completeness entries. The weight of each entry is based on the previous evaluation and was calculated as follows: the number of entry evaluations was divided by all the evaluations in that document type. As we excluded some of the entries before, we only used the ones that were left in the template. For all those entries the final weights were given after some generalisation and rounding. The new document completeness entries weights are described in Chapter 4.

Finally, we revised the evaluation supporting spreadsheet (described in Chapter 5) according to changes made to the templates and document types' new completeness calculation. The document organisation templates were used the same as in previous performance test. We did not remove any document types as none of them was counted as irrelevant. We removed all the entries that were to be excluded and added the new ones to documentation completeness based on the information gathered during looking into the OSS projects' documentation. Then we added the new weights that were described in the previous paragraph to the *DIA* calculation (Formula 5).

3.2. Test the revised DA model

To evaluate the tools we carried out the process described in Fig. 3, but with the respect to new revised templates (Appendix D) on seven Business Process Management (BPM) and Unified Modelling Language (UML) tools under the Open Source license.

3.3. Summary

In this chapter we described the workflow we used to evolve the existing *DA model* in [10,11]. We also presented the method that was used to find the tools and evaluate the OSS projects documentation. In the last paragraph we gave a brief overview of the process of testing the *revised DA model*.

In the next chapter we describe the revisions done during the research process. We present the new documentation completeness templates with new and revised entries, describe the documentation completeness entries and introduce the supporting spreadsheet used for evaluation.

CHAPTER 4: **Revision of the Document Availability Model**

In this chapter we describe the changes made to *DA model* (described in Chapter 2) during the revision process (described in Chapter 3). The structure of this chapter is based on the research process described in Fig 4.

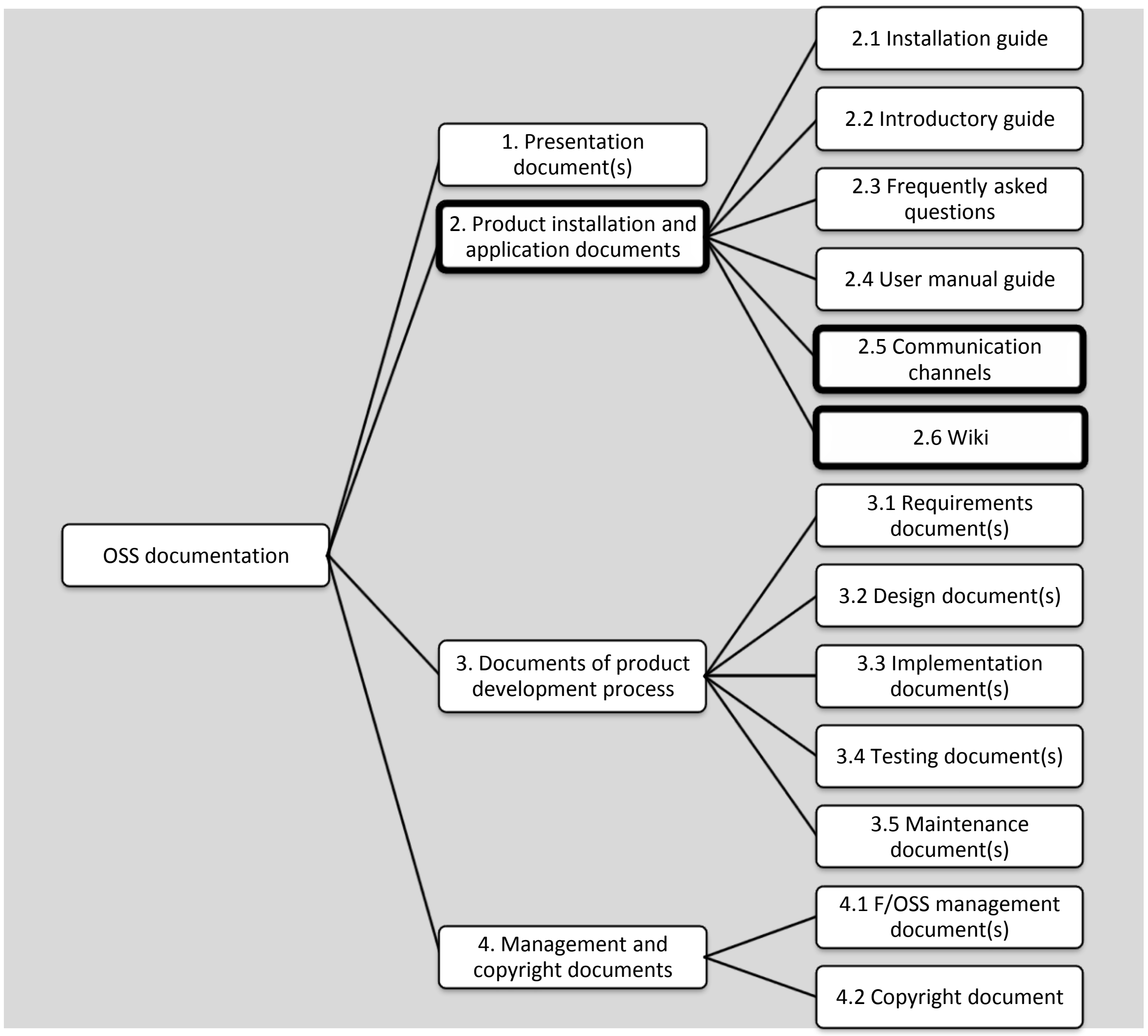

OSS documentation
1. Presentation document(s)
2. Product installation and application documents
2.1 Installation guide
2.2 Introductory guide
2.3 Frequently asked questions
2.4 User manual guide
2.5 Communication channels
2.6 Wiki
3. Documents of product development process
3.1 Requirements document(s)
3.2 Design document(s)
3.3 Implementation document(s)
3.4 Testing document(s)
3.5 Maintenance document(s)
4. Management and copyright documents
4.1 F/OSS management document(s)
4.2 Copyright document

Fig. 6 - Enhancement of document types highlighting the document types that have been changed

4.1. Enhancement of the documentation typology

As described above the documents are divided into 4 different groups depending on interests of the stakeholders in the project. Fig. 6 illustrates the changes made to document types during the model revision process. Document types that are shown with thick border were modified. In *Product installation* and *Application documents* respect to OSS project we changed the title of *Accumulative Experience notes* to *Communication channels* as it refers clearly to what it is being used in OSS projects. We described a new document type *Wiki*, because many of the OSS products use it as a documentation repository. Based on the structure of *Wiki* in OSS we described documentation completeness entries to it. The new document completeness template is described in Table 2. The document organisation template was used the same as in previous evaluation (Appendix B).

Table 2 – Template for document type completeness of ID2.6 Wiki

Concepts used by Measure or Measurement Procedure	**Basic Measure (unique name)**
Document information availability, Document completeness	**completeness_of_ID2.6_Wiki**
Does "Getting started" exist in the ID.2.6?	Provides with the basic information how to install and use the tool.
Does FAQ exist in the ID.2.6?	Provides information on frequently asked questions.
Does Debugging exist in the ID.2.6?	Covers the main debugging methods.
Does "Releases information" exist in the ID.2.6?	Provides information about the previous and upcoming releases.
Does "User documentation" exist in the ID.2.6?	Provides information what and how the end-user uses the program.
Does "Developer documentation" exist in the ID.2.6?	Provides a guide that covers the development strategies, tools and main classes.
Does "Contributing to the wiki" exist in the ID.2.6?	Provides information how all community members can contribute to the wiki.
Does "Tutorials" exist in the ID.2.6?	Provides basic tool use cases with detailed explanations.
Does "External Resources" exist in the ID.2.6?	Provides references to related work outside the project.
Does "Events, Courses, Conferences" exist in the ID.2.6?	Provides with information about community meetings.

4.2. Revision of the document completeness entries

In this section we describe the changes made to the templates assessing *document type completeness*. In Fig. 7 the document types where changes have been made are marked with think border. For each template we describe the changes made respect to the templates used in [10,11]. All the revised templates, including the ones where no changes were made, can be found in the Appendix D.

4.2.1. Requirements document

In *Requirements document* (ID.3.1) a lot of entries were not evaluated in OSS projects. As all these entries seem to be irrelevant for OSS projects they have been deleted from the template. The reason for irrelevancy of these entries is mainly because the OSS projects usually do not have a specific customer for who these requirements are made. The requirements are often described just before a new development cycle.

4.2.2. Design document

In *Design document* (ID.3.2) the number of entries has been shortened and some entries have been extended. The template's entries that were too excessive for "*Design documents*" ("*Data dependencies*" and "*Data detailed design*"), have been put together under entry "*Data description*". The entries that were not used were excluded.

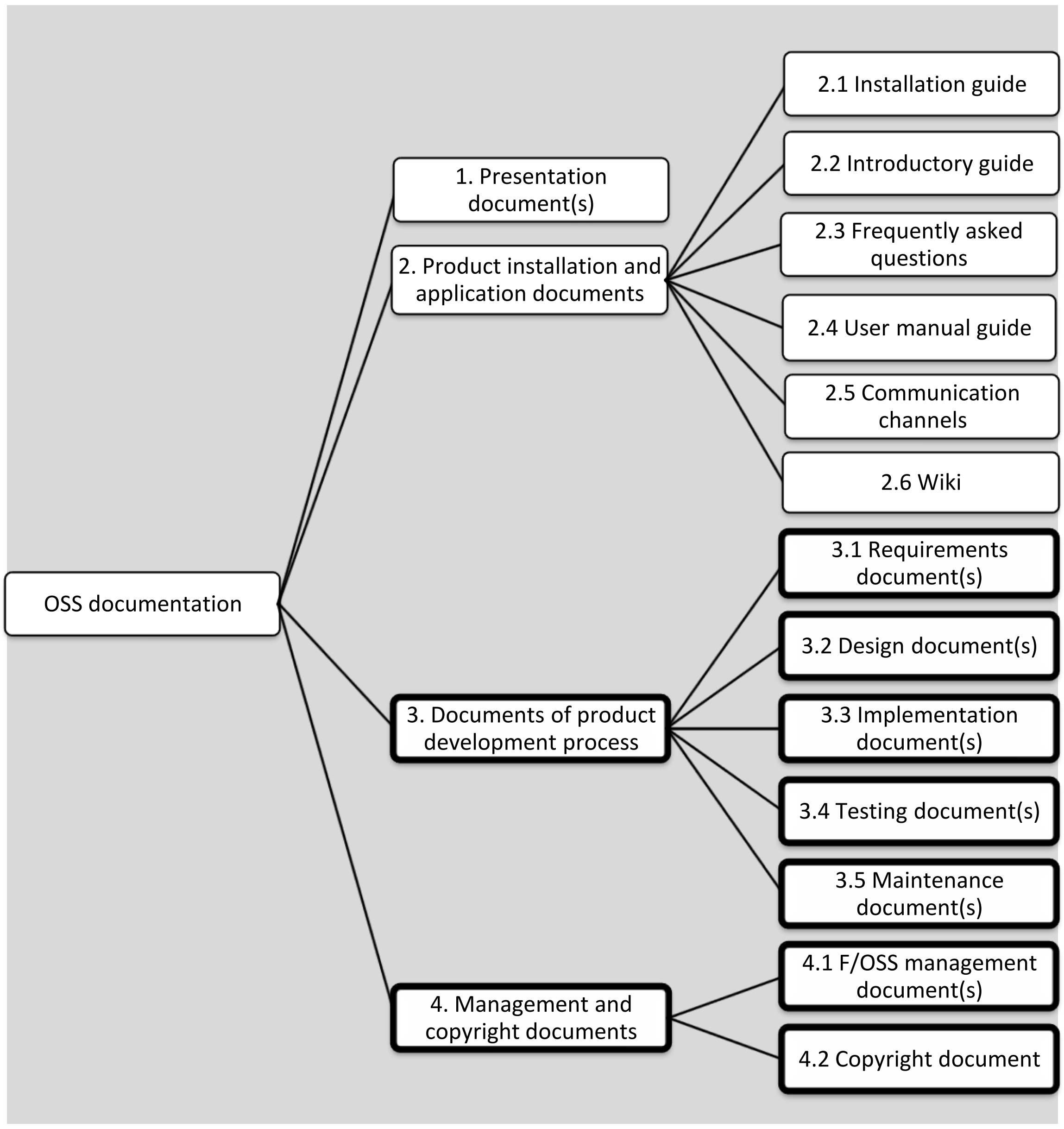

OSS documentation
1. Presentation document(s)
2. Product installation and application documents
2.1 Installation guide
2.2 Introductory guide
2.3 Frequently asked questions
2.4 User manual guide
2.5 Communication channels
2.6 Wiki
3. Documents of product development process
3.1 Requirements document(s)
3.2 Design document(s)
3.3 Implementation document(s)
3.4 Testing document(s)
3.5 Maintenance document(s)
4. Management and copyright documents
4.1 F/OSS management document(s)
4.2 Copyright document

Fig. 7 - Document groups and document types highlighting the entries that have been changed

4.2.3. Implementation document

In *Implementation document* (ID.3.3) the "*Appendix*" entry has been renamed to "*Comments in the source code*" to describe more accurately its content. The template has been extended with entries "*Building from source*", "*Tools for implementation*", "*Code commit rules*", "*Application programming interface (API)*" and "*How to commit*". These added entries are specific for OSS software development.

4.2.4. Testing document

OSS projects often lack testing, but it is important when the goal of the community is to generate a quality product. In *Testing document* (ID.3.4) a lot of entries have been excluded as not being used in OSS development. From previous evaluation [10,11] we can see that a lot of tools have minimal testing documents and it is like to influences on the quality of the product in the end. In the revised template there are 11 entries, but in the standard there were 51 [7]. For example we removed entries "*Staffing and training needs*", „*Approvals*“, „*Status*“, „*Impact*“, „*Variances*“, „*Summary of results*“, *etc*. The exclusion of entries has been done by information from previous evaluation and our subjective opinion. The testing document for OSS project usually covers a release, because the requirement and functionalities to be developed are often made clear just before a new development cycle.

4.2.5. Maintenance document

In *Maintenance document* (ID.3.5) all the entries that were not used during the previous evaluation have been excluded. Exclusion of entries with minimal valuation has been decided by the writer depending on their description and relevance to OSS projects.

4.2.6. Management document

The *Management document*'s (ID.4.1) template has been shortened and a lot of entries that are not used by OSS projects have been excluded. For example "*Product acceptance plan*", "*Infrastructure plan*", "*Budget control plan*", "*Budget allocation*", "*Project staff training plan*", "*Estimation plan*", "*Project deliverables*". These entries are irrelevant for OSS projects as these projects are usually free and they do not have a specific client.

4.2.7. Copyright document

As copyright violations in relation to OSS projects are abundant the existing *Copyright document*'s (ID.4.2) template has been extended to provide better protection to contributors. The previous template has been replaced by a template with ten new entries (Table 3).

Table 3 - *Copyright document* completeness template

Entry	**Description**
Does "Definitions part" exist in the ID.4.2?	Defines the concepts of the document.
Does " Grant of copyright" exist in the ID.4.2?	Describes how the copyrights conditions are related to You, work and contributors.
Does "Grant of Patent license" exist in the ID.4.2?	Describes how the patent of the product is related to You.
Does "Redistribution part" exist in the ID.4.2?	Defines how the Tool may be redistributed and what conditions it must meet.
Does "Submission of contributions" exist in the ID.4.2?	Describes the submissions licence.
Does "Trademark" exist in the ID.4.2?	Describes how the trademark is used or how they may be used.
Does "Disclaimer of warranty" exist in the ID.4.2?	Defines the warranties and its conditions.

Does “Limitation of liability” exist in the ID.4.2?	Defines the liability of the contributors and You.
Does “Accepting Warranty or additional liability” exist in the ID.4.2?	Specifies how You may use the distributed product on your own behalf and how the liability and warranty extends to it.
Does “Applying or the interpretation of the licence” exist in the ID.4.2?	Covers information about how to use the licence in Your project.

4.3. Weights of the document completeness entries

Finally we added weights to each entry of the template because the parts that are important to all OSS projects should have higher weight than the ones that are not essential to all of the projects. The weight for each entry is calculated by the sum of maximum points for completeness in one document divided by the sum of points gained for completeness in one entry. The data for the calculation was taken from the previous evaluation. The weights have been rounded and generalised. For the new entries added during the research the weights were added based on our subjective opinion. For the new document type “*Wiki*”, we added weight *0.1* for all entries (ten entries altogether). The weights for each template entry can be found in Appendix D.

Based on the new weights we defined a new formula for calculating *Document completeness* (based on Equation 2). It is the sum of multiplications between content completeness, information completeness and weight, divided by the information completeness scale maximum value:

$$dco = \frac{\sum_{i=1}^{M} c_i d_i q_i}{3} \qquad (5)$$

Equation 5 - $\{c_1 \ldots c_M\}$ 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 $\{0<1<2<3\}$, *q* weight of the entry, and *dco* – completeness of documents.

4.4. Summary

Previously we described the main contributions of the work. We revised one document type, introduced one new template, removed irrelevant document completeness entries, added new document completeness entries, added weights to document completeness entries and defined a new formula (Equation 5) for document completeness calculation.

In the next chapter we present the supporting spreadsheet that is used for evaluation of the documentation availability.

CHAPTER 5: **Tool Support**

To evaluate OSS project's *documentation availability* we use supporting spreadsheet which template can be found in Appendix E. The spreadsheet is divided into 6 parts. We describe the spreadsheet in the order as given in Fig. 8.

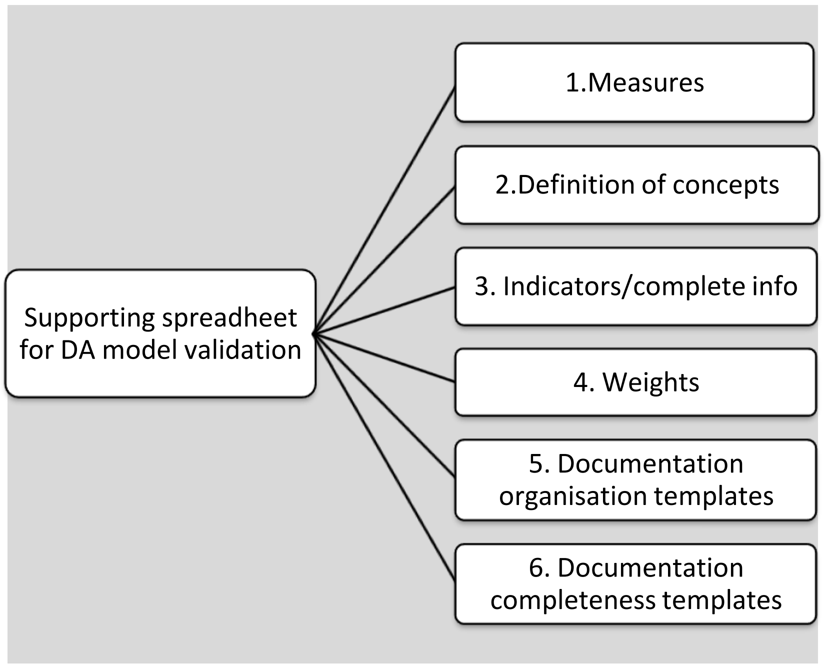

Supporting spreadsheet for DA model validation
1.Measures
2.Definition of concepts
3. Indicators/complete info
4. Weights
5. Documentation organisation templates
6. Documentation completeness templates

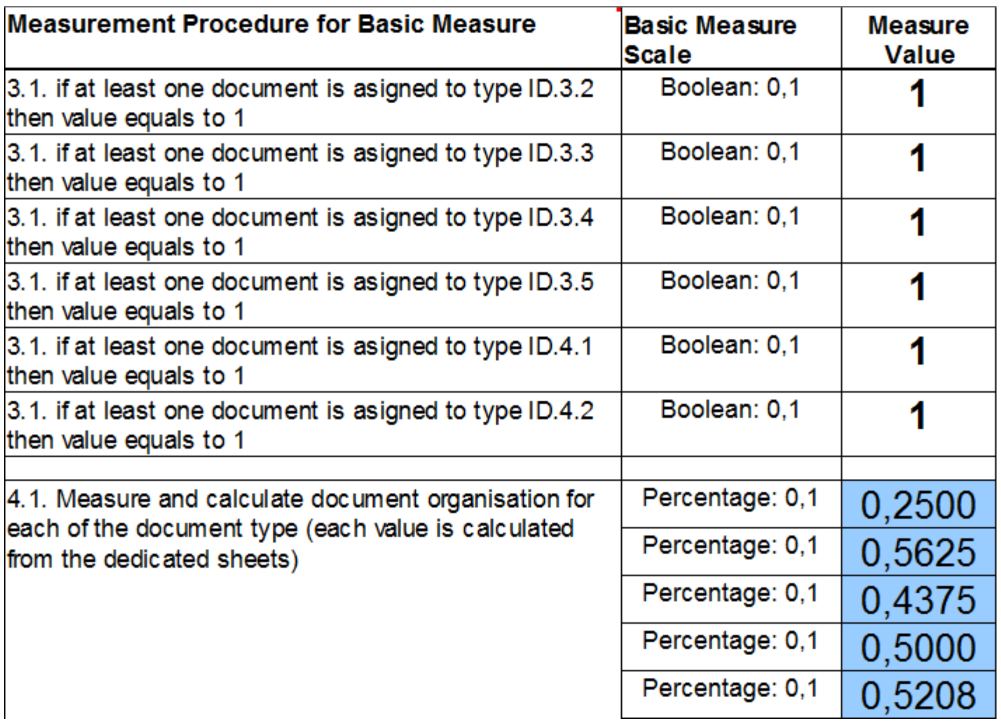
Fig. 8 - Structure of supporting spreadsheet

5.1. Measures

In measures sheet we define all the metrics used for evaluating projects' documentation. We have metrics for document type availability, documentation organisation, documentation completeness. All the document types' evaluations are summarised in that sheet. In Fig. 9 the values with light blue are the results for document types' organisation and completeness. When organising the documents to document types we also evaluate the existence of each document type. Value "1" means that the document type exists. Value "0" means that this type of document was not found.

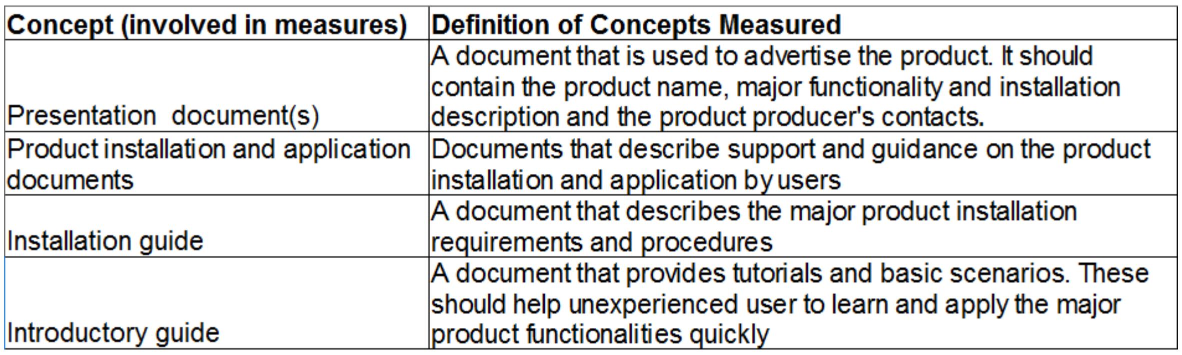
5.2. Definition of concepts

Definition of concepts sheet gives the definitions of the main concepts used during the measuring process. It is given as a list of definitions where in the first column we define the concept and in the second column we describe it. A screenshot from the Definition of concepts page has been added (Fig. 10).



Measurement Procedure for Basic Measure	Basic Measure Scale	Measure Value
3.1. if at least one document is asigned to type ID.3.2 then value equals to 1	Boolean: 0,1	1
3.1. if at least one document is asigned to type ID.3.3 then value equals to 1	Boolean: 0,1	1
3.1. if at least one document is asigned to type ID.3.4 then value equals to 1	Boolean: 0,1	1
3.1. if at least one document is asigned to type ID.3.5 then value equals to 1	Boolean: 0,1	1
3.1. if at least one document is asigned to type ID.4.1 then value equals to 1	Boolean: 0,1	1
3.1. if at least one document is asigned to type ID.4.2 then value equals to 1	Boolean: 0,1	1
4.1. Measure and calculate document organisation for each of the document type (each value is calculated from the dedicated sheets)	Percentage: 0,1	0,2500
	Percentage: 0,1	0,5625
	Percentage: 0,1	0,4375
	Percentage: 0,1	0,5000
	Percentage: 0,1	0,5208

Fig. 9 - Part of Measures sheet

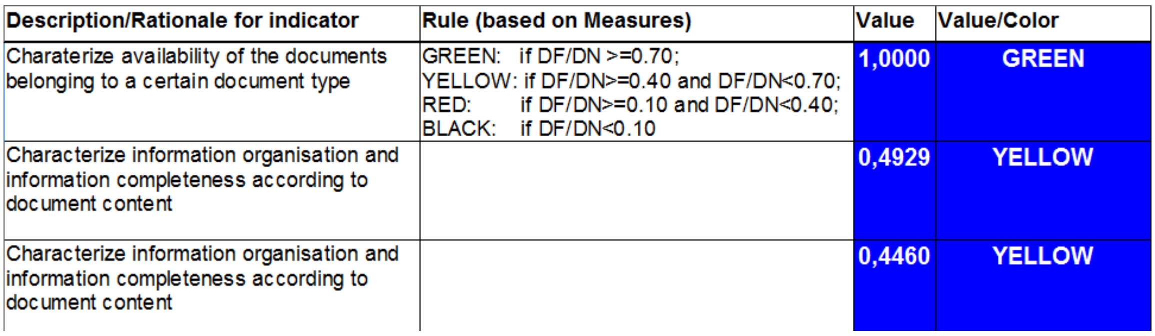


Concept (involved in measures)	Definition of Concepts Measured
Presentation document(s)	A document that is used to advertise the product. It should contain the product name, major functionality and installation description and the product producer's contacts.
Product installation and application documents	Documents that describe support and guidance on the product installation and application by users
Installation guide	A document that describes the major product installation requirements and procedures
Introductory guide	A document that provides tutorials and basic scenarios. These should help unexperienced user to learn and apply the major product functionalities quickly

Fig. 10 - Part of Definition of concepts sheet

5.3. Indicators/complete info

In the sheet the characteristics of documentation availability, organisation and completeness are given for the evaluated tool. In Fig. 11 the first column describes the metrics, second column describes the interpretation model, third has the valuations and in the fourth column is the value based on the indicator. This sheet is automatically completed when all the other sheets have been evaluated.

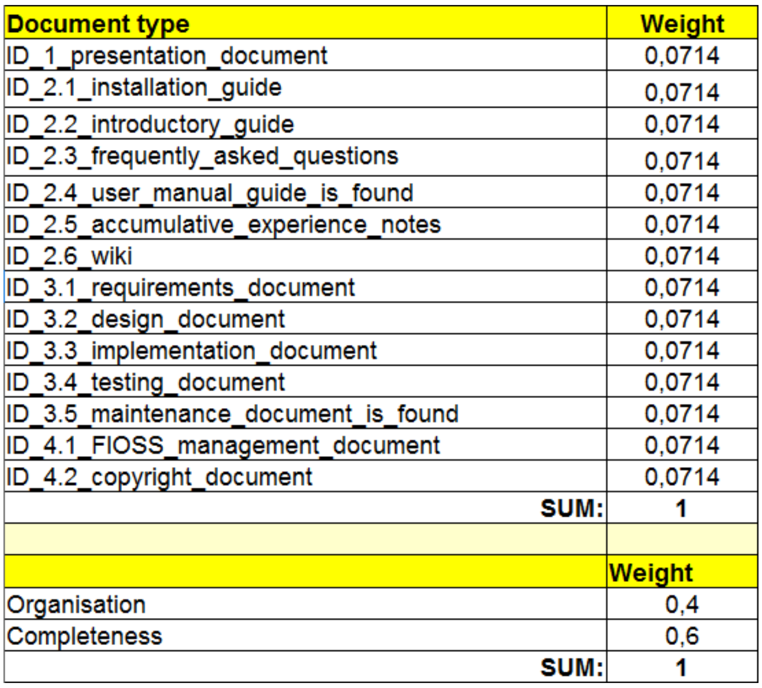


Description/Rationale for indicator	Rule (based on Measures)	Value	Value/Color

Charaterize availability of the documents belonging to a certain document type	GREEN: if DF/DN >=0.70;
	YELLOW: if DF/DN>=0.40 and DF/DN<0.70;
	RED: if DF/DN>=0.10 and DF/DN<0.40;
	BLACK: if DF/DN<0.10
		1,0000	GREEN

Characterize information organisation and information completeness according to document content		0,4929	YELLOW
Characterize information organisation and information completeness according to document content		0,4460	YELLOW

Fig. 11 - Part of Indicators/complete info sheet



Document type	Weight
ID_1_presentation_document	0,0714
ID_2.1_installation_guide	0,0714
ID_2.2_introductory_guide	0,0714
ID_2.3_frequently_asked_questions	0,0714
ID_2.4_user_manual_guide_is_found	0,0714
ID_2.5_accumulative_experience_notes	0,0714
ID_2.6_wiki	0,0714
ID_3.1_requirements_document	0,0714
ID_3.2_design_document	0,0714
ID_3.3_implementation_document	0,0714
ID_3.4_testing_document	0,0714
ID_3.5_maintenance_document_is_found	0,0714
ID_4.1_FIOSS_management_document	0,0714
ID_4.2_copyright_document	0,0714
SUM:	1
	Weight
Organisation	0,4
Completeness	0,6
SUM:	1

Fig. 12 - Weights of different document types, documentation completeness and organisation

5.4. Weights

In Fig. 12 the weight for each document type (all even) and *documentation organisation* and *documentation completeness* are defined. This is a informative sheet as all the weights are constants.

5.5. Documentation organisation templates

The document organisation templates are filled after filtering OSS's project's documents into document types. For each entry in the template we search in the document for accordance. Document organisation is evaluated, the same way as in [10,11], with values (Yes/No/NA). Based on the entries questions Yes – 1, No – 0 and if the template's part was not found then NA. The value is calculated as a ratio between the number of positively answered questions and the number of considered questions. Only the columns on the right with white background are modified and the yellow colour field (the organisation of that document type) is calculated automatically.

Question	Basic Measure (unique name)	Measurement Procedure for Basic Measure	Basic Measure Scale	Value
What is organisation of documents belonging to document type ID.1. Presentation documents	organisation_of_ID1_ presentation_document	Sum of all document organisations divided by the document number	Percentage: 0,1	0,2500
	number_of_documents_belonging_to_ID1		1,5	2
	Organisation of Document 1	**sum all answers and divide by the number of considered answers**		**0,3**
Is document divided to chapters?	chapters	check the document and locate if division to chapters exist. If yes, evaluate as 1, if no – as 0	Boolean: 0,1	1
Is document organised to sections and/or subsections?	sections	check the document and locate if division to sections exist. If yes, evaluate as 1, if no – as 0	Boolean: 0,1	1

Fig. 13 - Document organisation template in the spreadsheet

5.6. Document completeness templates

After filling one document type's organisation template we fill in the documentation completeness template (example shown in Fig. 14) of that document type. The document completeness template consists of header, where the main information and classifications of that document type is described, question, where the entries evaluation question is defined, description, to help the evaluation process, measures scale where the measures for each entry are given. For each entry first, we evaluate the existence (if exists then 1, if not then 0). The next row (where the scale is from 0 – 3) is used for documentation completeness evaluation. As in document organisation template we only fill the *Value* column. Entries in the right most columns (indicated by yellow) are the weights of each entry and the top yellow field gives a numeric value for the documentation completeness of that document type.

Concepts used by Measure or Measurement Procedure	Basic Measure (unique name)	Basic Measure Scale	Time	Value	Question weight
Document information availability, Document completeness	completeness_of_ID3.4 testing_document	Percentage: 0,1		**0,0833**	
Does "Test plan identifier" exist in the ID.3.4?	Specify the unique identifier assigned to this test plan	0,1		0,00	**0,05**
		0,1,2,3		0,00	
Does "Introduction" exist in the ID.3.4?	Summarize the software items and software features to be tested. The need for each item and its history may be included	0,1		1,00	**0,10**
		0,1,2,3		1,00	

Fig. 14 - Document completeness template in the spreadsheet

5.7. Summary

For this chapter we described the supporting tool that we used for the evaluation process of *documentation availability*. We went through all the sheet types that the *revised DA model* has. In the next chapter we will evaluate the tools with filling up the supporting spreadsheet for each OSS tool.

CHAPTER 6: **Evaluation of OSS Tools**

In this chapter we give a brief overview of the tools that we used and give the results gained from evaluation. The evaluation has been done with the support of template described in Chapter 5. The evaluation was performed between 15.04.2011 and 15.05.2011. Tools' websites that were evaluated are given in Appendix F and the filled spreadsheets in Appendix G.

6.1. Tools

AndroMDA is an open source MDA (Model-driven architecture) framework - it takes any number of models combined with any number of androMDA plugins and produces any number of custom components.

ArgoUML is the leading open source UML (Unified Modelling Language) modelling tool that includes support for all standard UML 1.4 diagrams. It runs on any Java platform and is available in ten languages.

BOUML is a free UML 2 tool box allowing specifying and generating code in C++, Java, Idl, Php and Python. The project has been chosen for evaluation as the project has been closed because of license violations.

Dia is roughly inspired by the commercial Windows program 'Visio,' though more geared towards informal diagrams for casual use. It can be used to draw many different kinds of diagrams. It currently has special objects to help draw entity relationship diagrams, UML diagrams, flowcharts, network diagrams, and many other diagrams.

PapyrusUML is aiming at providing an integrated and user-consumable environment for editing any kind of EMF (Eclipse Modelling Framework) model and particularly supporting UML and related modelling languages.

StarUML is a software modelling platform that supports UML. It is based on UML version 1.4 and provides eleven different types of diagrams, and it accepts UML 2.0 notation. It actively supports the MDA (Model-driven architecture) approach by supporting the UML profile concept.

Taylor is a specialised UML modelling tool based on Eclipse. It uses convention-based techniques to generate the maximum code from streamlined UML models. Templates are included for generating JEE applications.

ProcessMaker is business process management (BPM) and workflow system designed to optimise the business operations and workflow management for small to medium sized businesses and organisations.

Orchestra is a complete solution to handle long-running, service oriented processes. It provides out of the box functionalities to handle complex business processes. It is based on the BPEL (Business Process Execution Language) standard.

Activity is a light-weight workflow and Business Process Management tool. Activity runs in any Java application, on a server, on a cluster or in the cloud.

uEngine is a BPM system that is integrated with notable open source applications - Liferay Enterprise Portal, Mondrian OLAP Server, JBoss Drools BRE, and Apache Axis II. It has multiple process instances control and event-driven flow control.

CuteFlow is a web-based document circulation and workflow system. All operations like starting a workflow, tracking, workflow-definition or status observation can be done within a comfortable and easy to use web interface.

Archi is targeted toward all levels of Enterprise Architects and Enterprise Modellers. It is intended to provide a low cost to entry (*i.e.* free) solution to users who may be making their first steps in the ArchiMate language or who are looking for a fully-featured, professional cross-platform ArchiMate modelling tool for their company or institution.

6.2. Evaluation results

Table 4 presents the assessment results following the interpretation model presented in [10]. The documentation type availability is high (*green*) for four projects, and it is average (*yellow*) for two projects. The documentation information availability is average (*yellow*) for one project and limited (*red*) for four projects.

Table 4 - Evaluation results

OSS Tool	Interval scale		Ordinal scale	
	DIA (%)	DTA(%)	DIA (colour)	DTA (colour)
Activity	53	93	Green	Green
Orchestra	52	100	Green	Green
ArgoUML	52	100	Green	Green
ProcessMaker	51	93	Green	Green
PapyrusUML	37	79	Yellow	Yellow
diaUML	34	71	Red	Red
Taylor	30	57	Red	Black
uEngine	26	50	Red	Black
AndroMDA	25	57	Red	Black
Archi	21	79	Black	Yellow
boUML	21	43	Black	Black
IntalioBPM	20	57	Black	Black
StarUML	19	57	Black	Black
CuteFlow	16	43	Black	Black

The results show that the *DIA* is still pretty low to be evaluated as green. The maximum evaluation of *DIA* was 53% and the lowest was 16%. One project (*Archi*) has average level of *DTA* but, still limited level of DIA. For all the other projects there seems to be a correlation between *DTA* and *DIA*. In Table 5 the tools evaluation is shown in respect to indicators.

Table 5 - Matrix of the OSS Project Assessment

		Document Information Availability (DIA)			
		Black (0 – 21,5%)	**Red (21,5 – 37%)**	**Yellow (37 – 45%)**	**Green (45 - 100%)**
Document Type Availability (DTA)	**Green (88,5 – 100%)**				*ArgoUML, ProcessMaker, Orchestra, Activity*
	Yellow (72,5 – 88,5%)	*Archi*		*PapyrusUML*	
	Red (58 – 72,5%)		*diaUML*		
	Black (0 – 58,0%)	*StarUML, IntalioBPM, CuteFlow, boUML*	*AndroMDA, Taylor, uEngine*		

In previous evaluation [10,11], the number of entries that were not evaluated (when evaluating 24 OSS tools) was 67. In the *revised DA model* there were ten entries that had no evaluation in 14 tools that we used for validation. Four of the non-valuated entries were from testing document and only four tools had a testing document present.

6.3. Summary

During validation of our *revised DA model we* evaluated 14 different OSS tools. We found four projects where the documentation information and organisation was present in high level (*ArgoUML*, *ProcessMaker*, *Orchestra*, and *Activity*). There were also four projects where *DIA* and *DTA* were minimal (*StarUML*, *IntalioBPM*, *CuteFlow*, *boUML*). Three of the tools (*AndroMDA*, *Taylor*, *uEngine*) have level of *Not available* for *DTA*, but *DIA* is evaluated as limited. This might mean that the documents satisfy some stakeholders in high level, but not for all. Project *Archi* has average level of *DTA*, but the information in the documents does not fulfil the needs of stakeholders.

CHAPTER 7: Conclusions and Future Work

In this thesis we consider the *DA model* [10,11], revise its' documentation completeness templates regarding the OSS documentation domain. In this chapter we describe the threats to the validity, provide the conclusions, and highlight the potential future research. We try to answer the question whether the *revised DA model* is more relevant to OSS evaluation than the model developed in [10,11].

7.1. Threats to validity

This thesis is not without validity threats:

- The validity of our revision is influenced by the subjective judgement of the author of this thesis. However we acknowledge that all the decisions are based either on the previous research results [10] or by the careful investigation of the information available with the selected OSS documentations.
- The assessment of the OSS products, illustrated in Chapter 6, rely on the subjective judgement. This evaluation is a manual task and the results depend on the assessor's experience. To ensure the quality of the results for some projects the evaluation was iterated few times, when the assessor felt unsure about some evaluations. In comparison to the previous experience [10, 11], there the OSS evaluations were performed and reviewed by several assessors.
- We applied the *revised DA model* on 14 tools, but the results range was still pretty same as in [10], despite the fact that we eliminated the entries that were not used in previous evaluations and added weights to *DIA* entries. The reason might be that we might have not selected the OSS tools (see Chapter 6) that have good quality of documentation.

7.2. Discussion

We eliminated all the entries that were not used in previous evaluation. Therefore we do not need to evaluate documentation parts that are irrelevant to OSS projects. In [10,11] there existed 67 entries that were not used in 24 projects that they evaluated. In our revised model with 14 tools we had 10 entries that had no evaluation, four of which were from the testing document. Thus we result in a more relevant assessment to the OSS documentation than the previous *DA model*.

With including weights to documentation completeness entries we help to stress on the quality aspects that are more important to different stakeholders (who might have different goals). For example, the existence of unit implementation (weight 0,25) is more important than the existence of references (weight 0,05), as identified from analysis of the previous comparisons.

The *revised DA model* is focused on documentation availability: it takes into account the documentation organisation as in [9] and documentation completeness. It also uses some of the characteristics from [1,14]. For example readability, ease of update from [14] and correctness, consistency from [1] are evaluated in document type organisation. Thoroughness [14] and completeness [1] are taken into account when evaluating *documentation completeness*. Compared to [1,9] the *revised DA model* gives a straightforward method with tool support to evaluate a project's quality. When applying the *revised DA model* to OSS project's documentation, it shows the project's documentation information availability and document type availability. Based on the knowledge we can decide whether all the stakeholders' interests are present and which document types need to be improved.

For *revised DA model* to become usable by practitioners it should be tested on more projects. The model should be thoroughly analysed from the documentation organisation aspect as well. As for our evaluation the structure of organisation was not sufficient as most of the documents were in webpage not in document format. To make it more usable there should be developed a IS where practitioners could add new evaluations and the documentation quality level could be calculated in respect to all previous evaluations in the system.

7.3. Conclusions

In this work we presented a quality model and revised it to assess documentation availability for the OSS products. Our main goal was to improve the existing model and make it more relevant for OSS products. Firstly we took over the existing *DA model* [10,11] and then revised it from *documentation completeness* aspect.

The analysis results based on previous evaluation and looking into the OSS tools documentation showed that the *DA model* described in [10,11] did not assess the OSS documentation at full extent. Based on the examination and looking into the OSS projects documentation we introduced a new document template *ID.2.6 Wiki*. We also revised the entries of documentation completeness templates. We removed all the entries that were not evaluated in the previous evaluation process in [10,11] and some which had low level of evaluation. As we removed some entries we also found new entries that are relevant for OSS. For example for "*Copyright"* document we added ten new entries ("*Definitions", "Grant of copyright", "Grant of Patent license", "Redistribution", "Submission of contributions", "Trademark", "Disclaimer of warranty", "Limitations of liability", "Accepting warranty", "Applying or the interpretation of the license"*). The calculation of documentation completeness was also changed: we added weight to each completeness entry, so that each entry has an importance indicator. The weights are calculated by the data from the previous evaluation (Appendix C).

Finally we investigated the validation of the *revised DA model* through performance test. The new model has fewer entries than the previous to check, therefore, it shortens the evaluation time for each tool. The results are still relevant as the entries that were removed were not used by OSS projects. In previous evaluation [10,11] there were 67 entries that were not evaluated during performance test, in the revised one 10. With adding weights to documentation completeness entries we take into account the importance of an entry unit and therefore the results are more accurate and depend on the previous evaluation.

In the performance test we did not have any goals to find the best and worst-documented project. Rather the aim of the test was to analyse whether the revisions made to the *DA* *model* improved the model and give suggestions how to improve the quality of documentation in OSS projects. For example, the average level of document type availability does not guarantee the high level of documentation completeness (*e. g, Archi*). This means that despite there are a lot of documents, they do not cover the needs of stakeholders. We also found projects where the document type availability was minimal, but the documentation completeness respect to the document types available was high; for example, the *diaUML, Taylor, uEngine, AndroMDA* tools. This means that they only satisfy the needs of some stakeholders but not all.

7.4. Future work

Regarding the future work, the next step would be to revise the document organisation templates as well to make it applicable for OSS.

Although we tested our revised DA model on 14 tools, the evaluation for previous tools in selected in [10,11] should be repeated to see how the revision process changed the assessment results. In such a case the evaluation should be carried on by the same assessor as in the first time in order to keep the expectations for entries, and to reduce a factor of subjectivity.

To contribute to validity of our *DA model*, a workshop of at least 7-10 persons should be held. The goal of this workshop could be evaluation of different OSS tools and gathering of participants' opinions about the different aspects of the DA model.

Finally the long-term result should be an IS where the quality of documentation in OSS projects could be evaluated in respect to previous evaluations. The weights of each completeness entries would be calculated by documentation quality results.

RESÜMEE

Avatud lähtekoodiga tarkvaraprojektide Dokumentatsiooni Kättesaadavuse mudeli optimiseerimine

Bakalaureusetöö (6 EAP)
Kaarel Tark

Avatud lähtekoodiga tarkvara (inglise keeles – Open Source Software) on üks uusimaid trende tänapäeva tarkvaraarenduses. Nagu nimigi ütleb, on avatud lähtekoodiga tarkvara kood avalik, ning võimaldab seega kõigil huvilistel osaleda tarkvaraarenduse protsessis. Tänasel päeval põhinevad paljud infosüsteemid rohkemal või vähemal määral avatud koodiga tarkvaral. Tarkvara tootmine sel viisil on odav võrreldes traditsioonilise tarkvaraarendusega kuna projektis osalejad edendavad tarkvara tavaliselt oma enda huvist ja vabast tahtest. Kuna avatud lähtekoodiga tarkvara projektides osalejad on erineva taustaga ja oskustasemega, on ka projektide kvaliteet kõikuv. Tagamaks kvaliteetset lõpptoodet on oluline hinnata jooksvalt arendamise käigus projekti hetkeseisu, et teada kuidas parandada või säilitada toote kvaliteeti. Projekti kvaliteedi hindamisel võib lähtuda mitmetest eri aspektidest: tarkvara koodi kvaliteedist, toote kvaliteedist või toote dokumentatsiooni kvaliteedist. Käesolevas bakalaureusetöös oleme keskendunud avatud lähtekoodiga projektide dokumentatsiooni kvaliteedile ja kvaliteedi hindamisele.

Selleks, et adekvaatselt hinnata tarkvara projekti dokumentatsiooni kvaliteeti, on vajalik vastavate meetodite olemasolu. Hetkel on olemas vaid mõned meetodid hindamaks avatud lähtekoodiga tarkvaraprojekti kvaliteeti, kuid nende peamiseks puuduseks on kindlate mõõtmiskriteeriumite puudumine. Käesoleva töö eesmärgiks on välja töötada dokumentatsiooni kvaliteedihindamise mudel hindamaks avatud lähtekoodiga tarkvara.

Uue mudeli aluseks on eelnevalt väljatöötatud dokumentatsiooni kvaliteedihindamise mudel (Dokumentatsiooni Kättesaadavuse mudel, inglise keeles *Documentation Availability (DA) model*). See mudel põhineb erinevate tootearenduses eksisteerivate huvigruppide - toote omandaja, toote kasutaja, toote arendaja, arenduse finantseerija – vajadustest dokumentatsiooni järele. Antud mudel ei ole loodud spetsiaalselt avatud lähtekoodiga tarkvara hindamiseks vaid baseerub üldistel IEEE tarkvaraarenduse standarditel. Seetõttu ei kata DA mudel täielikult avatud lähtekoodiga tarkvaraprojektide vajadusi.

Antud töös analüüsisime ja optimiseerisime dokumentatsiooni kvaliteedi hindamise mudelit (DA mudel) lähtudes varasemalt läbi viidud uuringu andmete analüüsist, ja avatud lähtekoodiga tarkvara dokumentatsiooni uurimisest. DA mudeli adapteerimisel kasutuseks avatud lähtekoodiga tarkvara dokumentatsiooni kvaliteedi hindamiseks (1) elimineerisime mudelist dokumentatsiooni sisutiheduse kirjed, mis ei ole avatud koodiga tarkvara puhul kasutusel (2) lisasime mudelisse relevantsed kirjed, näiteks "*Koodi*

kompileerimine", "*Arendusvahendid*" ja dokumendi tüübid, näiteks "*Wiki*" (3) iga kirje osatähtsuse määramiseks ühe dokumendi tüübi lõikes lisasime dokumentatsiooni sisutiheduse kirjetele kaalud.

Muudatuste tulemina valmis uus spetsiaalselt avatud lähtekoodiga tarkvara dokumentatsiooni kvaliteedihindamise mudel. Mudeli valideerimiseks hindasime 14 avatud lähtekoodiga tarkvaral põhinevat äriprotsesside analüüsi ja tarkvara modelleerimise projekti dokumentatsiooni. Tulemused näitavad, et uus mudel sobib avatud lähtekoodiga tarkvara dokumentatsiooni hindamiseks ning võimaldab hinnata projektide kvaliteeti täpsemalt kui töö aluseks võetud DA mudel.

ABSTRACT

A Revision of the Documentation Availability Model for Open Source Software

Bachelors thesis (6 EAP)
Kaarel Tark

Open source software is one of the current trends in software development. Many information systems (IS) are built more or less on OSS. The OSS development process is cheaper as the contributors do it to for free from their own interest. The background of the contributors varies, as varies their skill level. In order to have a good quality product you have to evaluate the current situation to see how to maintain or increase the quality of a product. The quality of a tool can be analysed from different aspects: code quality, tool quality or documentation quality. Our work focuses on documentation quality.

There are only couple of methods how to evaluate OSS documentation quality. For our work basis we chose *Documentation Availability* model (*DA model*). This model is based on the needs of different stakeholders (*product acquirer*, *product user*, *product developer* and *product contractor*). The main limitation of the model is that it is not designed for OSS, but is rather based on the IEEE general software development standards. Therefore it does not apply completely for the OSS documentation.

In this thesis we analysed and revised the *DA model*, based on the data of the previous research and on the data we observed at the selected OSS documentations. The revised *documentation availability* model (1) excludes unused completeness entries (2) includes relevant entries for example "*Code commit rules*", "*Building from Source*" (3) adds new document types for example "*Wiki*". We also enforce documentation completeness entries with the weights, which helps take into account importance of separate entries.

To validate our proposal we have analysed 14 OSS products from the Business Process management and software modelling domain. The results show that the new introduced model applies to the OSS development process and can be used to evaluate OSS documentation quality more relevantly than the model used before.

REFERENCES

[1] 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)

[2] Deprez, J.-C., Haaland, K., Kamseu, F.: QualOSS Methodology and QualOSS Assessment Method, Deliverable D4.1, http://www.qualoss.org/deliverables (last checked 27.04.2011)

[3] IEEE: IEEE Recommended Practice for Software Design Descriptions, IEEE Std 1016-1998 (1998)

[4] IEEE: IEEE Recommended Practice for Software Requirements Specification, IEEE Std 830-1998 (1998)

[5] IEEE: IEEE Standard for Software Maintenance, IEEE Std 1219-1998 (1998)

[6] IEEE: IEEE Standard for Software Project Management Plans, IEEE Std 1058-1998 (1998)

[7] IEEE: IEEE Standard for Software Test Documentation, IEEE Std 829-1998 (1998)

[8] IEEE: IEEE Standard for Software User Documentation, IEEE Std 1063-2001 (2001)

[9] Le Vie D. S. Jr.: Documentation Metrics: What Do You Really Want to Measure? http://www.stc.org/intercom/PDFs/2000/200012_06-09.pdf (last checked 12.05.2011)

[10] Matulevičius, R., Kamseu, F., Habra, N.: Measuring Open Source Documentation Availability. In: proceeding of the 12th International Conference on Quality Engineering in Software Technology (CONQUEST 2009), dpunkt. Verlag GmbH, pp. 83 -102 (2009)

[11] Matulevičius, R., Kamseu, F., Habra, N.: Validity of the Documentation Availability Model: Experimental Definition of Quality Interpretation. In: B. Pernici (Ed.): CAISE 2010, LNCS 6051, pp. 236 -250, 2010. Spring-Verlag Berlin Heidelberg 2010.

[12] Moody, D.L.: Theoretical and Practical Issues in Evaluating the Quality of Conceptual Models: Current State and Future Directions. Data and Knowledge Engineering 55 (3) 243--276. (2005)

[13] Piattini, M., Genero, M., Poels, G., Nelson, J.: Towards a Framework for Conceptual Modelling Quality. In: Genero, M., Piattini, M., Calero, C. (eds.) Metrics for Software Conceptual Models, Imperial College Press, London 1--18. (2005)

[14] Schiesser, R.: How does your process documentation measure up? http://articles. techrepublic.com.com/5100-10878_11-1053164.html (last checked 10.05.2011)

[15] The Case for Government Promotion of Open Source Software
http://www.netaction.org/opensrc/oss-whatis.html (last check: 19.03.2011)

[16] The open source definition | Open Source Iniative
http://www.opensource.org/docs/osd, (last check: 31.01.2011)

[17] The open source Licences by Category | Open Source Iniative
http://www.opensource.org/licenses/category (last check: 10.02.2011)

APPENDIX A: Unrevised templates

The documentation completeness templates that were used in [10,11] are described in the CD. In the first column there are the entries definition and in the second column the evaluation aspects. The templates are given for all 12 documentation information completeness templates that were used during previous evaluation.

APPENDIX B: Document organisation template

The existing document organisation template is given in the CD. Each row represents a entry that was evaluated.

APPENDIX C: Analysis table respect to previous evaluation

This appendix covers the data analysis regarding previous evaluation in [10,11] and can be found in the CD. In the first row there are the tools' names. In the middle merged column there are the document template's codes (for example ID.2.1 which refers to Installation documents). In the left (for example c1) are the entry identifiers. In the right there is a column called *Count* which describes how many times that entry has been used in previous evaluation.

APPENDIX D: Revised Templates and Weights

In this appendix we are describing the new document completeness templates that were used during evaluation. In the first column we describe an entry question. In the second column there's the description of that question. In the right most columns the weight of that entry is given.

Entry	Description	Weight
Does "Name of the product" exist in the ID.1 Presentation document?	It is the name of the product.	0,25
Does "Version of the product" exist in the ID.1 Presentation document?	It is the version of the product.	0,25
Does "Release date of the product" exist in the ID.1 Presentation document?	The date of the release of the product.	0,20
Does "Snapshot of the product" exist in the ID.1 Presentation document?	That is a copy of a set of files and directories of the and their location.	0,15
Does "Ways to contact the FlOSS" exist in the ID.1 Presentation document?	Different ways to contact the FlOSS (email, mailing list, address,...).	0,15

Entry	Description	Weight
Does "Prerequisites, Requirements" exist in the ID.2.1?	Documented need for the installation procedure.	0,30
Does "The source file (Website, Zip File or SVN)" exist in the ID.2.1?	Different source code files for the installation and their location.	0,35
Does "Installation, Configuration, Authentication" exist in the ID.2.1?	It is information related to the installation procedure, the configuration file and the authentication data used to effectively put the program in a computer system so that it can be executed.	0,25
Does "Language Packs" exist in the ID.2.1?	List of language available for the installation procedure.	0,10

Entry	Description	Weight
Does “Introduction” exist in the ID.2.2?	Describe the intended audience, scope, and purpose for the document and include a brief overview of the software purpose, functions, and operating environment.	0,20
Does “Installation” exist in the ID.2.2?	It is information related to the installation procedure, the configuration file and the authentication data used to effectively put the program in a computer system so that it can be executed.	0,15
Does “Getting started” exist in the ID.2.2?	Information needed to start effectively using the product.	0,15
Does “Basis concepts and functionalities” exist in the ID.2.2?	They are the main functionalities and concept in the product.	0,20
Does “Getting help” exist in the ID.2.2?	Answer of the main questions of the user.	0,15
Does “Further reading” exist in the ID.2.2?	More references that can help to have more information.	0,15

Entry	Description	Weight
Does “Overview” exist in the ID.2.3?	FAQ about the purpose, scope, and objectives of the project, the project assumptions and constraints.	0,15
Does “Technology (installing and running the tool...)” exist in the ID.2.3?	FAQ about the technology of the product	0,20
Does “Mirrors” exist in the ID.2.3?	FAQ about the location and some resources (SVN, CVS, repositories, ...) of the product.	0,10
Does “Community” exist in the ID.2.3?	FAQ about the community of the OSS.	0,10
Does “Licensing” exist in the	FAQ about the license of the product.	0,10

ID.2.3?		
Does “The help system” exist in the ID.2.3?	FAQ about the support system for the OSS.	0,05
Does “Language” exist in the ID.2.3?	FAQ about the different language of the product.	0,05
Does “Development” exist in the ID.2.3?	FAQ about the development of the product.	0,15
Does “Others” exist in the ID.2.3?	FAQ about the other issues in the OSS.	0,10

Entry	Description	Weight
Does “Introduction” exist in the ID.2.4?	Describe the intended audience, scope, and purpose for the document and include a brief overview of the software purpose, functions, and operating environment.	0,15
Does “Information for use of the documentation” exist in the ID.2.4?	Include information on how it is to be used and an explanation of the notation.	0,20
Does “Concept of operations” exist in the ID.2.4?	Explain the conceptual background for use of the software, using such methods as a visual or verbal overview of the process or workflow; or the theory, rationale, algorithms, or general concept of operation.	0,15
Does “Procedures” exist in the ID.2.4?	Instructional mode documentation that provides directions for performing procedures. Instructions shall include preliminary information, instructional steps, and completion information.	0,15
Does “Information on software commands” exist in the ID.2.4?	Explain the formats and procedures for user-entered software commands, including required parameters, optional parameters, default options, order of commands, and syntax.	0,15

Does “Error messages and problem resolution” exist in the ID.2.4?	Address all known problems in using the software in sufficient detail such that the users can either recover from the problems themselves or clearly report the problem to technical support personnel.	0,10
Does “Related information sources” exist in the ID.2.4?	Contain information on accessing related information sources, such as a bibliography, list of references, or links to related web pages.	0,10

Entry	Description	Weight
Does “Announcements” exist in the ID.2.5?	Community information of upcoming releases, meetings, conferences, testing periods...	0,25
Does “Mailing lists” exist in the ID.2.5?	The tool development team has a mailing list.	0,25
Does “Discussions forums” exist in the ID.2.5?	There’s a public discussion forum.	0,25
Does “Wiki” exist in the ID.2.5?	Collection of Web pages designed to enable anyone who accesses it to contribute or modify content, using a simplified mark-up language.	0,25

Entry	Description	Width
Does “Getting started” exist in the ID.2.6?	Provide with the basic information how to install and use the tool.	0,10
Does FAQ exist in the ID.2.6?	Provide information on frequently asked questions?	0,10
Does Debugging exist in the ID.2.6?	Covers the main debugging methods.	0,10

Does “Releases information” exist in the ID.2.6?	Provide information about the previous and upcoming release.	0,10
Does “User documentation” exist in the ID.2.6?	Provide information what and how the end-user uses the program.	0,10
Does “Developer documentation” exist in the ID.2.6?	Provide a guide that covers the development strategies, tools and main classes.	0,10
Does “Contributing to the wiki” exist in the ID.2.6?	Provide information how all community members can contribute to the wiki.	0,10
Does “Tutorials” exist in the ID.2.6?	Provide basic tool use cases with detailed explanations.	0,10
Does “External Resources” exist in the ID.2.6?	Provides references to related work outside the project.	0,10
Does “Events, Courses, Conferences” exist in the ID.2.6?	Provide with information about community meetings.	0,10

Entry	Description	Weight
Does “Purpose” exist in the ID3.1?	Delineate the purpose of the requirements document; Specify the intended audience for the requirements document.	0,10
Does “Product function” exist in the ID3.1?	Provide a summary of the major functions that the software will perform.	0,15
Does “Constraints” exist in the ID3.1?	Provide a general description of any other items that will limit the developer’s options.	0,10
Does “Assumptions and Dependencies” exist in the ID3.1?	List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements.	0,10

Does “User interfaces” exist in the ID3.1?	The logical characteristics of each interface between the software product and its users; All the aspects of optimizing the interface with the person who must use the system.	0,10
Does “Reliability” exist in the ID3.1?	Specify the factors required to establish the required reliability of the software system at time of delivery.	0,10
Does “Maintainability” exist in the ID3.1?	Specify attributes of software that relate to the ease of maintenance of the software itself.	0,10
Does “Portability” exist in the ID3.1?	Specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems.	0,10
Does “Other requirements” exist in the ID3.1?	Other requirement might include specification of issues, off-the-shelf solutions, new problems, tasks, cutover, risks, costs, user documentation and training, other ideas for solutions.	0,15

Entry	**Description**	**Weight**
Does “Purpose” exist in the ID.3.2?	A design document is a representation or model of the software system to be created. The model should provide the precise design information needed for planning, analysis, and implementation of the software system. It should represent a partitioning of the system into design entities and describe the important properties and relationships among those entities. The design description model used to represent a software system can be expressed as a collection of design entities, each possessing properties and relationships.	0,05
Does “Scope” exist in the ID.3.2?	Identify the software product(s) to be produced; Explain what the product(s) will, and will not do; Describe benefits, objectives, and goals.	0,05

Does “Reference documents” exist in the ID.3.2?	Provide a complete list of all documents referenced elsewhere in the design document.	0,05
Does “Module description” exist in the ID.3.2?	The decomposition description records the division of the software system into design entities. It describes the way the system has been structured and the purpose and function of each entity. For each entity, it provides a reference to the detailed description via the identification attribute. The attribute descriptions for identification, type, purpose, function, and subordinates should be included in this design view. This attribute information should be provided for all design entities.	0,25
Does “Data description” exist in the ID.3.2?	Provide information that covers used data types, data transportation, data analysis, data conversion.	0,05
Does “Intermodal dependencies” exist in the ID.3.2?	The dependency description specifies the relationships among entities. It identifies the dependent entities, describes their coupling, and identifies the required resources. This design view defines the strategies for interactions among design entities and provides the information needed to easily perceive how, why, where, and at what level system actions occur. It specifies the type of relationships that exist among the entities such as shared information, prescribed order of execution, or well-defined parameter interfaces. The attribute descriptions for identification, type, purpose, dependencies, and resources should be included in this design view. This attribute information should be provided for all design entities.	0,20
Does “Data dependencies” exist in the ID.3.2?	The dependency description of data in different processes.	0,05
Does “Module interface” exist in the ID.3.2?	The entity interface description provides everything designers, programmers, and testers need to know to correctly use the functions provided by an entity. This description includes the detail of external and internal interfaces not provided in the software requirements specification. This design view consists of a set of interface specifications for each entity. The attribute descriptions for identification, function, and interfaces should be included in	0,20

	this design view. This attribute information should be provided for all design entities.	
Does “Process interface” exist in the ID.3.2?	Covers the interfaces of business processes related to the tool. Points out the main situations where different processes interfere with each-other or outer systems.	0,05
Does “Module detailed design” exist in the ID.3.2?	The detailed design description contains the internal detail of each design entity. These details include the attribute descriptions for identification, processing, and data. This attribute information should be provided for all design entities.	0,05

Entry	**Description**	**Weight**
Does “Introduction” exist in the ID.3.3?	Describe the specific purpose, goals, and scope of the software implementation effort.	0,05
Does “References” exist in the ID.3.3?	Identify the documents placing constraints on the implementation effort, documents referenced by the implementation plan, and any supporting documents supplementing or implementing the implementation plan including other plans or task descriptions that elaborate detail of this plan	0,05
Does “Definitions” exist in the ID.3.3?	Define or reference all terms required to understand the implementation plan.	0,10
Does “Software decomposition to separate implementation units” exist in the ID.3.3?	Describe how the overall software is decomposed for implementing it. Explain how interfaces between separate software units are implemented.	0,20
Does “Unit implementation” exist in the ID.3.3?	Characterise how each individual software unit is implemented within the software.	0,25
Does “Traceability” exist in the ID.3.3?	Describe how each of the implemented software units satisfies the design and requirements solutions. Describe which software units were not implemented in the current version of the software.	0,05

Does “Comments in the source code” exist in the ID.3.3?	Provide the source code. The source code must be fully explained by the complementary text (usually natural language).	0,05
Does “Building from Source” exist in the ID.3.3?	Give the tutorial how to build from the source code.	0,05
Does “Tools for implementation” exist in the ID.3.3?	Defines the tools to be used by the OSS community.	0,05
Does “Code commit rules” exist in the ID.3.3?	Defines the rules how to use code commitment in the community.	0,05
Does “Application programming Interface (API) exists” exist in the ID.3.3?	Does the project API exist?	0,05
Does “How to commit” exist in the ID.3.3? How to commit	Does it cover the exact information about how the code commitment is to be done in the community?	0,05

Entry	Description	Weight
Does “Test plan identifier” exist in the ID.3.4?	Unique identifier assigned to this test plan.	0,05
Does “Introduction” exist in the ID.3.4?	Summarize the software items and software features to be tested. The need for each item and its history may be included .	0,10
Does “Test items” exist in the ID.3.4?	Identify the test items including their version/revision level. Also specify characteristics of their transmittal media that impact hardware requirements or indicate the need for logical or physical transformations before testing can begin.	0,05

Does “Features to be tested” exist in the ID3.4?	Identify all software features and combinations of software features to be tested. Identify the test design specification associated with each feature and each combination of features.	0,10
Does “Approach” exist in the ID3.4?	Describe the overall approach to testing. For each major group of features or feature combinations, specify the approach that will ensure that these feature groups are adequately tested. Specify the major activities, techniques, and tools that are used to test the designated groups of features.	0,10
Does “Responsibilities” exist in the ID3.4?	Identify the groups responsible for managing, designing, preparing, executing, witnessing, checking, and resolving.	0,05
Does “Test items” exist in the ID3.4?	Identify and briefly describe the items and features to be exercised by this test case.	0,05
Does “Purpose” exist in the ID3.4?	Describe the purpose of this procedure.	0,10
Does “Special requirements” exist in the ID3.4?	Identify any special requirements that are necessary for the execution of this procedure.	0,10
Does “Procedure steps” exist in the ID3.4?	Include the steps of the procedure.	0,15
Does “Description” exist in the ID3.4?	Identify the attributes of the environments in which the testing is conducted. Identify the items being tested including their version/revision levels.	0,10
Does “Test incident report identifier” exist in the ID3.4?	Specify the unique identifier assigned to this test incident report.	0,05

Entry	**Description**	**Weight**
Does “Definitions” exist in the ID.3.5?	Define or reference all terms required understanding the maintenance plan.	0,05

Does “Organisation” exist in the ID.3.5?	Describe the organisation of the software maintenance effort. Describe the lines of communication with the software maintenance effort including external organisations, the authority for resolving issues raised in the software maintenance effort, and the authority for approving software maintenance products.	0,05
Does “Scheduling priorities” exist in the ID.3.5?	Describe what the priorities are in time.	0,10
Does “Resource summary” exist in the ID.3.5?	Summarize the software maintenance resources, including staffing, facilities, tools, finances, and special procedural requirements.	0,10
Does “Responsibilities” exist in the ID.3.5?	Identify an overview of the organisational element(s) and responsibilities for maintenance activities.	0,10
Does “Tools, techniques and methods” exist in the ID.3.5?	Describe the special documents, software maintenance tools, techniques, methods, and operating and test environment to be used in the maintenance process.	0,05
Does “Problem/ modification identification/ classification, and prioritisation” exist in the ID.3.5?	Identify actions to be performed in case of normal modifications and upcoming probable problem situations. Should have priorities and solutions to the situations.	0,25
Does “Anomaly resolution and reporting” exist in the ID.3.5?	Describe the method of reporting and resolving anomalies, including the criteria for reporting an anomaly, the anomaly distribution list, and authority for resolving anomalies.	0,20
Does “Standards, practices, and conventions” exist in the ID.3.5?	Identify the standards, practices, and conventions that govern the performance of maintenance actions including internal organisational standards, practices, and policies.	0,05
Does “Quality control of plan” exist in the ID.3.5?	Describe how the plan is reviewed, updated, and approved to ensure plan correctness and currency.	0,05

Entry	Description	Weight
Does “Purpose, scope and objectives” exist in the ID.4.1?	Define the purpose, scope, and objectives of the project and the products to be delivered.	0,10
Does “Assumptions and constraints” exist in the ID.4.1?	Describe the assumptions on which the project is based and imposed constraints on project factors such as the schedule, budget, resources, software to be reused, acquirer software to be incorporated, technology to be employed, and product interfaces to other products.	0,10
Does “Project deliverables” exist in the ID.4.1?	List the work products that will be delivered to the acquirer, the delivery dates, delivery locations, and quantities required to satisfy the terms of the project agreement.	0,05
Does “Schedule and project summary” exist in the ID.4.1?	Provide a summary of the schedule and budget for the software project.	0,05
Does “Evolution of the plan” exist in the ID.4.1?	Specify the plans for producing both scheduled and unscheduled updates to the SPMP.	0,05
Does “External interface” exist in the ID.4.1?	Describe the organisational boundaries between the project and external entities.	0,10
Does “Internal structure” exist in the ID.4.1?	Describe the internal structure of the project organisation to include the interfaces among the units of the software development team.	0,10
Does “Roles and responsibilities” exist in the ID.4.1?	Identify and state the nature of each major work activity and supporting process and identify the organisational units that are responsible for those processes and activities.	0,20
Does “Staffing plan” exist in the ID.4.1?	Specify the number of staff required by skill level, the project phases in which the numbers of personnel and types of skills are needed, and the duration of need.	0,05
Does “Resources acquisition plan” exist in the ID.4.1?	Specify the plan for acquiring the resources in addition to personnel needed to successfully complete the project.	0,05
Does “Work activities” exist in the ID.4.1?	Specify the various work activities to be performed in the software project.	0,05

Does “Methods, tools and techniques” exist in the ID.4.1?	Specify the development methodologies, programming languages and other notations, and the tools and techniques to be used to specify, design, build, test, integrate, document, deliver, modify and maintain the project deliverable and non-deliverable work products.	0,05
Does “Process improvement plan” exist in the ID.4.1?	Include plans for periodically assessing the project, determining areas for improvement, and implementing improvement plans.	0,05

Entry	Description	Weight
Does “Definitions part” exist in the ID.4.2?	Defines the concepts of the document.	0,10
Does “Grant of copyright” exist in the ID.4.2?	Describe how the copyrights conditions related to You, work and contributors.	0,10
Does “Grant of Patent license” exist in the ID.4.2?	Describes how the patent of the product is related to You.	0,10
Does “Redistribution part” exist in the ID.4.2?	Define how the Tool may be redistributed and what conditions it must meet.	0,10
Does “Submission of contributions” exist in the ID.4.2?	Describe the submissions licence.	0,10
Does “Trademark” exist in the ID.4.2?	Describe how the trademark is used or how they may be used.	0,10
Does “Disclaimer of warranty” exist in the ID.4.2?	Define the warranties and its conditions.	0,10
Does “Limitation of liability” exist in the ID.4.2?	Define the liability of the contributors and You.	0,10
Does “Accepting Warranty or additional liability” exist in the ID.4.2?	Specify how you may use the distributed product on you on behalf and how the liability and warranty extends to it.	0,10
Does “Applying or the interpretation of the licence” exist in the ID.4.2?	Covers information about how to use the licence in Your project.	0,10

APPENDIX E: **Supporting spreadsheet**

The supporting spreadsheet is provided in the CD. The structure of the spreadsheet has been described in Chapter 5.

APPENDIX F: **Tools websites**

Here are the tools' websites that we used during evaluation.

Tool	**Project website**
AndroMDA	http://www.andromda.org/docs/whatisit.html
ArgoUML	http://argouml.tigris.org/
boUML	http://bouml.free.fr/
DiaUML	http://live.gnome.org/Dia
Papyrus	http://www.eclipse.org/modeling/mdt/papyrus/
StarUML	http://staruml.sourceforge.net/en/
Taylor	http://sourceforge.net/projects/taylor/
ProcessMaker	http://www.processmaker.com
intalioBPM	http://community.intalio.com/
Orchestra	http://orchestra.ow2.org/xwiki/bin/view/Main/WebHome
Activity	http://www.activiti.org/index.html
uEngine	http://www.uengine.org
CuteFlow	http://www.cuteflow.org/index.html
Archi	http://archi.cetis.ac.uk/

APPENDIX G: **Tools evaluation**

The results of 14 tools evaluation are provided. For each tool a filled supporting spreadsheet and document with tool documents are provided. The appendix is in the CD.