Journal of Cases on Information Technology
Volume 22 • Issue 2 • April-June 2020

Application of Quality in Use Model to Evaluate the User Experience of Online Banking Software

Manar Abu Talib, University of Sharjah, Sharjah, UAE

Areej Alsaafin, University of Sharjah, Sharjah, UAE

Selma Manel Medjden, University of Sharjah, Sharjah, UAE

ABSTRACT

Open source software (OSS) has recently become very important due to the rapid expansion of the software industry. In order to determine whether the quality of the software can achieve the intended purposes, the components of OSS need to be assessed as they are in closed source (conventional) software. Several quality in-use models have been introduced to evaluate software quality in various fields. The banking sector is one of the most critical sectors, as it deals with highly sensitive data; it therefore requires an accurate and effective assessment of software quality. In this article, two pieces of banking software are compared: one open source and one closed source. A new quality in use model, inspired by ISO/IEC 25010, is used to ensure concise results in the comparison. The results obtained show the great potential of OSS, especially in the banking field.

KEYWORDS

Cyclos, E-pay Suite, ISO/IEC 25010, Open Source Software

INTRODUCTION

The impact of open source resources on business has been noted in many sectors, especially with recent gains in various open source technologies from OSS to open source libraries (Hecht & Clark, 2018, Official Statistics of Finland, 2011). Banking is one of these sectors. Bearing in mind the sensitivity of the data managed by banks, banking software has to meet stringent criteria in terms of security and efficiency (Popp, 2015). In order to assess the suitability of software from this perspective, some quality standards must be defined.

Computer software is a term that includes all the parts of a computer system that handle data. Software can be a computer program, a library or any set of instructions that manage data. Open Source Software (OSS) is software that allows users to access and modify the source code, while closed source software provides users with functionality that can only be accessed through its unalterable user interface.

One advantage of OSS is its low cost. Since OSS source code is licensed to be freely accessible, only implementation, maintenance and training charges are necessary to start using an OSS product. However, if any support is needed, OSS users must rely on online communities, while closed source products generally offer after-sale support services. Another important advantage is the flexibility that OSS products provide. Indeed, as the source code of OSS is accessible, users can make any

DOI: 10.4018/JCIT.2020040103

necessary changes in the product. On the other hand, closed source products require users to adapt to the environment provided, without any possibility of making changes.

In all fields, the software components need to be assessed to determine whether the software quality is sufficient for the intended purposes. Recently, a large number of quality models have been introduced to help organizations determine if a given software can perform the required tasks adequately and effectively. These quality models also ensure that the widest range of people can use the software, as they improve accessibility and acceptance, increase efficiency, reduce errors and training requirements, and improve productivity (Bevan, 2001).

The quality of a software product plays a significant role in the success of a business, as it reflects the level of customer satisfaction. Every quality model measures software quality based on a number of characteristics. The first quality model, introduced by Jim McCall (1977), assessed quality factors to evaluate user satisfaction and guide developer priorities. The second quality model was a hierarchical model introduced by Boehm, Brown, Kaspar, Lipow, McLeod, and Merritt (1978). It consisted of primitive characteristics, intermediate level characteristics and high-level characteristics. Finally, a more recent quality model proposed by Geoff (1995) addresses the relationship between quality attributes and sub-attributes.

In this paper, the authors apply Alnanih's new quality in use model (Alnanih, 2015), inspired by ISO/IEC 25010 (ISO/IEC 25010:2011), to assess two online banking software products: an open source product called Cyclos, (Social TRade Organisation, 1970), and a closed source product called E-pay Suite (Canopus Innovative Technologies, 1992). The qualitative approach has four phases: pre-experiment, data gathering, data analysis and evaluation. The aim of this study is to assess whether or not the open source product is capable of performance similar to that of the closed source product.

The rest of this paper is structured as follows: first, we present a comprehensive set of quality in use models for assessing OSS, followed by an explanation of the methodology used to assess the two banking software products—one open source and one closed source. We then provide a description of the experiments conducted and a discussion of the results. Finally, our conclusions are provided, along with some practical applications for the future.

BACKGROUND

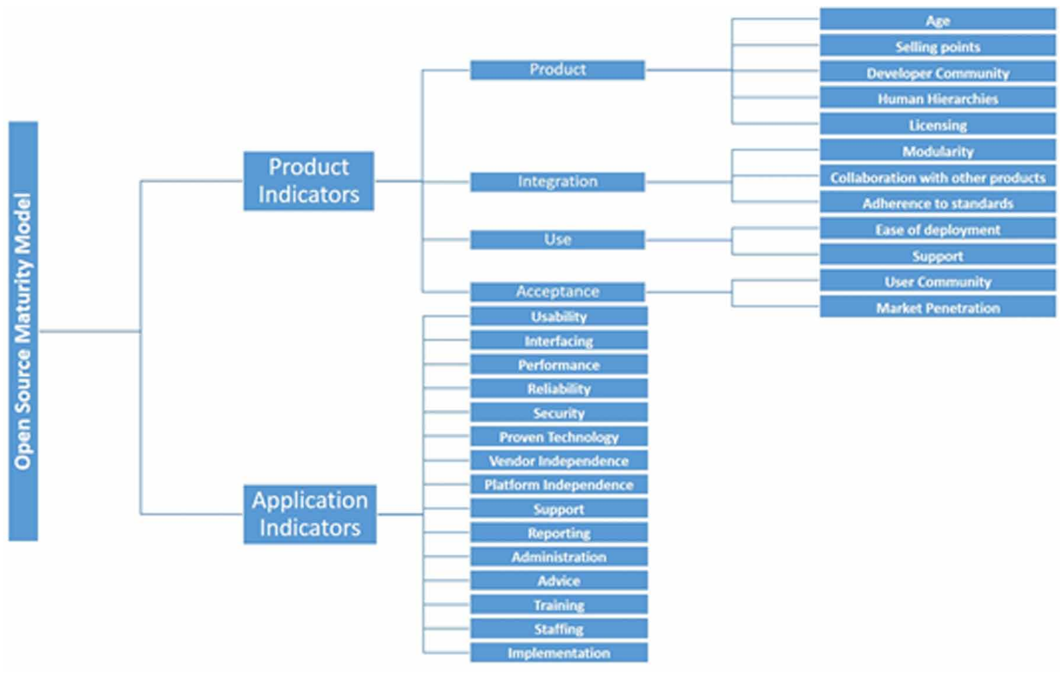
Over the past years, a wide range of open source evaluation tools have been used in different fields. In this section, the authors give an overview of the quality models available in the literature and discuss their applicability to OSS.

Open Source Maturity Model (OSMM)

In 2003, the open source maturity model (OSMM) was developed by Capgemini (Duijnhouwer & Widdows, 2003). The OSMM uses product maturity for the purpose of comparing various software products, with the aim of selecting the product that best fits the organization's objectives. While OSMM is a non-free software, authorized distribution is permitted. This model has two categories of indicator: product and application. The product indicator has four sub-categories: product, integration, use and acceptance. The application indicator considers several environmental, current, and future user requirements (Duijnhouwer & Widdows, 2003). The categories and subcategories of OSMM are illustrated in Figure 1. It is clear from the figure that OSMM considers of all the characteristics of product quality available in ISO/IEC 25010 (ISO/IEC 25010:2011), however, it only assesses the usability in the quality in use category.

In 2015, a survey of 200 Moroccan Small and Medium Enterprises (SMEs) was conducted by Houaich and Belaissaoui in order to identify their needs, knowledge and ability to adopt open source technology (Houaich & Belaissaoui, 2015). The authors matched the SMEs with suitable open source technology by designing a new assessment model, E-OSSEM, using the OSMM product category. This model allowed the authors to choose the most suitable open source technology for each SME.

Figure 1. Open Source Maturity Model (OSMM) (Duijnhouwer & Widdows, 2003)



Another research work by Akbari and Peikar (2014) analyzed Free/Open Source (FOSS) GIS tools in web mapping and spatial databases. The authors compared different WebGIS FOSS tools analytically using OSMM. These tools included UMN MapServer, MapGuide OS and FOSS spatial databases such as PostGIS. This work concluded that UMN MapServer is a completely mature OSS, and that its functionality and quality is comparable to other conventional (closed source) software products. In addition, it demonstrates that PostGIS is a highly competitive closed source software, especially with regard to its 3D functionality.

Open Business Readiness Rating (Open BRR)

In 2007, the Carnegie Mellon West Center sponsored the Open Business Readiness Rating (Open BRR) for SpikeSource, Open Source Investigation, Intel, and CodeZoo (Wasserman, Pal & Chan, 2006). The aim of this model design was to enable institutions to choose open source software that best suited their needs. It was found to improve the time required to evaluate an OSS by using a systematic approach consisting of four phases. In the first phase, a quick assessment was performed to create a shortlist of potential candidates. In the second phase, various metrics were ordered according to their importance. In the third phase, data was gathered and analyzed. Finally, in the fourth phase, the data was translated into a Business Readiness Rating to enhance the decision-making process and increase potential user confidence in OSS. The model employed qualitative metrics, and set weights for each metric. Using these weights, an overall score was assigned to each category by calculating individual metric scores. The measures used in Open BRR are shown in Figure 2.

Authors Das and Wasserman (2007) introduced a web-based application prototype, which helps users search for suitable OSS using Open BRR. They changed the user interface slightly to better comply with the BRR framework. The authors found that Open BRR should be seen as a collection of repositories that gathers data from multiple resources and synthesizes results. The rating must be able to reflect the continued emergence of new repositories, as well as changes in existing ones.

Authors Groven, Haaland, Glott and Tannenberg (2010) assessed the security level of Asterisk by applying Open BRR security measures. Asterisk is a FLOSS framework that aims to build communications applications.

Software Quality Observatory for OSS (SQO-OSS)

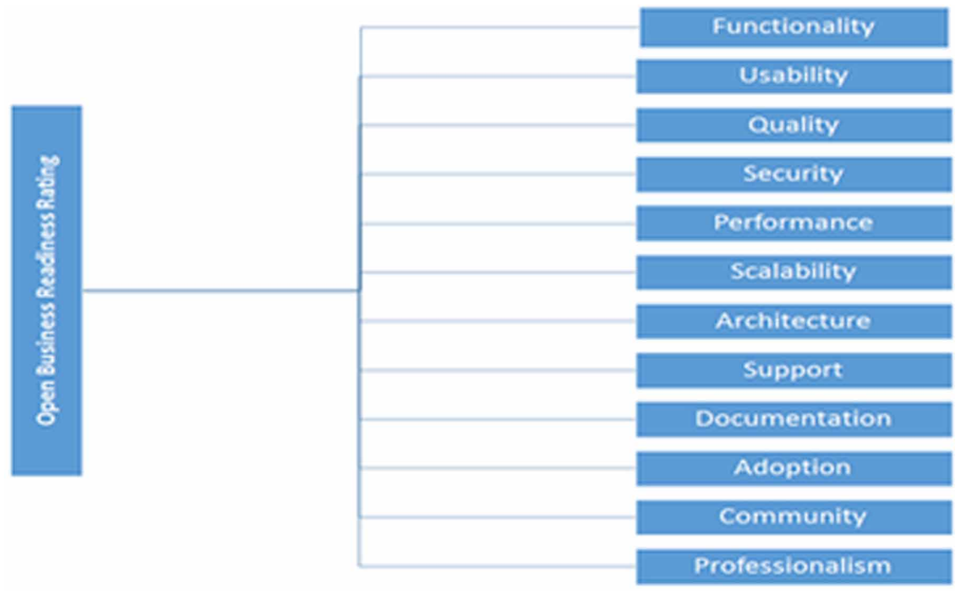
Software Quality Observatory for OSS (SQO-OSS) was developed by Samoladas, Gousios, Spinellis and Stamelos in 2008. This model performs detailed automated OSS quality assessments—including source code evaluations—to help users decide whether the target software meets their needs. This model hierarchically assesses both community processes and source code. The metrics used in SQO-OSS are illustrated in Figure 3.

Authors Groot, Kügler, Adams, and Gousios (2006) presented a quality assessment of the KDE project, which aimed to enable engineers to identify modifications needed to enhance the original product. The study found that the SQO-OSS model helps OSS developers write better software, while simultaneously helping potential users make better-informed choices.

Quality of OSS (QualOSS)

In 2009, Soto and Ciolkowski developed the Quality of OSS (QualOSS) model, which focuses on OSS robustness and evolvability. This model has two main categories of quality characteristics: product-related and community-related (shown in Figure 4). However, QualOSS does not include characteristics from any of the quality in use categories. Soto et al. (2009) applied QualOSS to twenty FLOSS programs, taking into account successful and unsuccessful software products. The aim of the study was to assess whether QualOSS could distinguish between successful and unsuccessful software products. The authors intended to modify and enhance QualOSS according to their evaluation of the results obtained. Introducing their own assessment procedures, the authors provided suggestions and conclusions based on their analysis of the model's assessment of different OSS projects. Researchers Groven et al. (2010) conducted an experiment to assess the security level of FLOSS implementation (Asterisk). They used nine security indicators, as well as the 30-40 pre-existing QualOSS security measures.

Figure 2. Open Business Readiness Rating (Open BRR) (Wasserman et al., 2006)



Evaluation Framework for Free/Open Source Projects (EFFORT)

In 2010, a framework called Evaluation Framework for Free/Open source projects (EFFORT) was designed to evaluate the quality and functionality of the target OSS. The framework assesses product quality, product attractiveness and community trustworthiness. To assess product quality, EFFORT uses ISO 9126 quality characteristics. The EFFORT metrics are shown in Figure 5. In another research work, the authors provided a customized model from EFFORT to evaluate Enterprise Resource Planning (ERP) OSS systems (Aversano & Tortorella, 2013). They applied the customized model to evaluate and compare five ERP OSS systems. The authors concluded that EFFORT was a useful model for assessing and selecting a suitable OSS system, which can lead to a significant reduction in the amount of negotiation between enterprise members. It also reduces the time and cost needed to collect and interpret data. Furthermore, the EFFORT model takes user opinions into account, providing relevance markers linked to metrics and questions during the process of data collection. In their research, Aversano and Tortorella (2011) used the same approach and customized EFFORT to design Free OSS (FOSS) for Customer Relationship Management (CRM) systems. They applied the customized model to four of the most common CRM systems. This led to good results for product quality and attractiveness; however, the results were less viable for community trustworthiness.

Figure 3. Software Quality Observatory for OSS (SQO-OSS) (Samoladas et al., 2008)

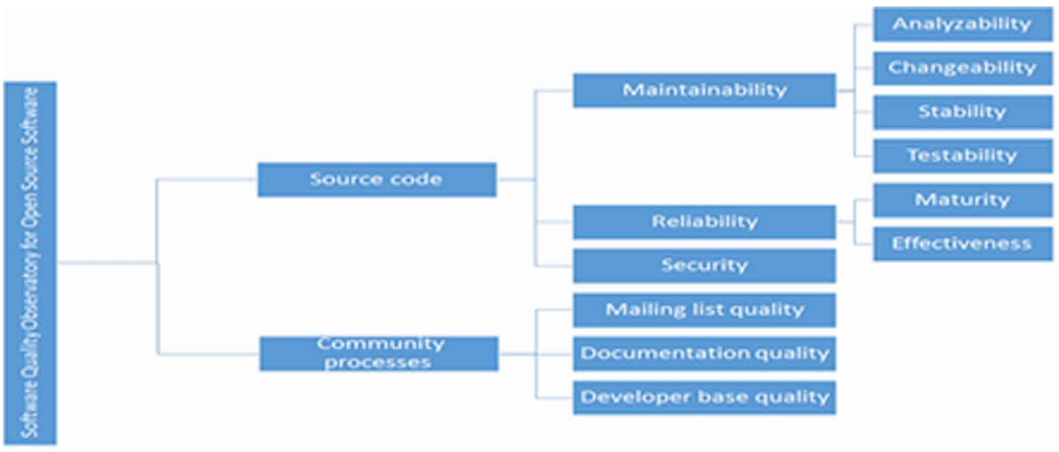


Figure 4. Quality of OSS (QualOSS) (Soto et al., 2009)

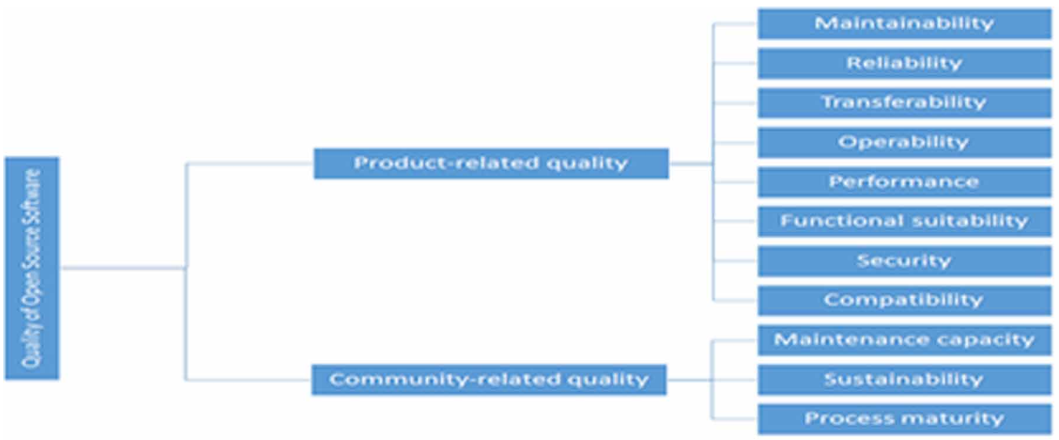
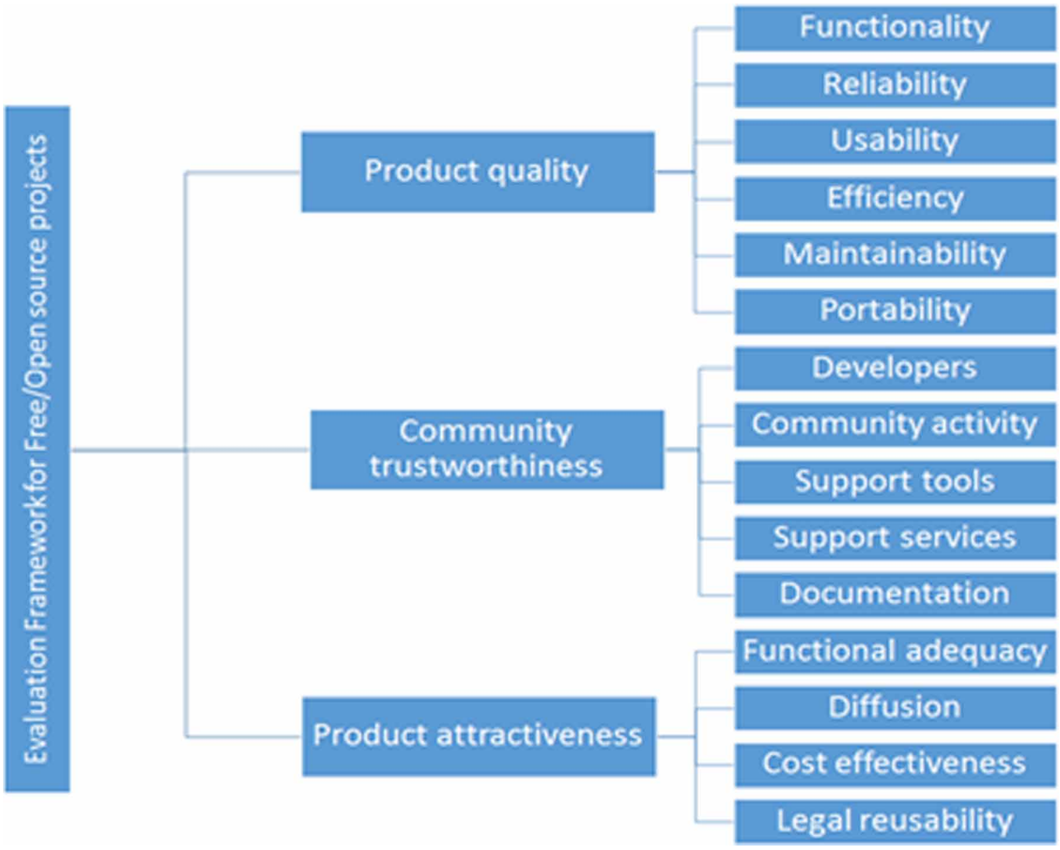


Figure 5. Evaluation Framework for Free/Open Source Projects (EFFORT) (Aversano & Tortorella, 2010)

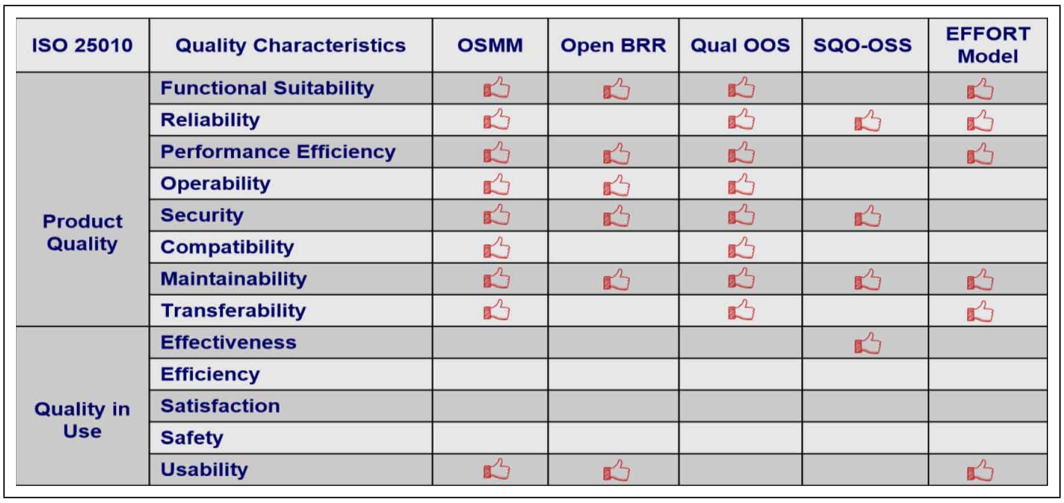


ISO/IEC 25010

In 2011, the ISO/IEC 25010 model was developed to determine the quality characteristics that should be considered when assessing a software product (ISO/IEC 25010:2011). System quality is the degree to which the system meets the stated and implicit needs of different stakeholders, therefore providing value. The needs of these stakeholders are fairly representative of the quality model, which defines the quality in use model and the product quality model. The quality in use model consists of five characteristics, illustrated in Table 1, related to the consequence of the interaction when the software product is used in a particular way. This system model applies to the complete human-computer system, which includes the computer systems and software products in use. The product quality model consists of eight characteristics, illustrated in Table 1, related to the software's static properties and the dynamic properties of the computer system. The model applies to both software products and computer systems. The characteristics specified in both models relate to all software products and computer systems. The characteristics and sub-characteristics provide consistent terminology to identify, measure and evaluate the quality of the system and the software product. They also offer a range of quality characteristics to which declared quality requirements can be compared for completeness. Although the product quality model is designed to assess software and computer systems, many of the characteristics can also be applied to broader systems and services.

The following table presents the model characteristics discussed in this section as compared to the ISO/IEC 25010 (ISO/IEC 25010:2011) features. There is clearly a lack of quality in use evaluation for this type of software in the literature.

Table 1. Comparison between ISO/IEC 25010 and OSMM, Open BRR, QualOSS, SQO-OSS, and EFFORT



ISO 25010	Quality Characteristics	OSMM	Open BRR	Qual OOS	SQO-OSS	EFFORT Model
Product Quality	Functional Suitability	👍	👍	👍		👍
	Reliability	👍		👍	👍	👍
	Performance Efficiency	👍	👍	👍		👍
	Operability	👍	👍	👍		
	Security	👍	👍	👍	👍	
	Compatibility	👍		👍		
	Maintainability	👍	👍	👍	👍	👍
	Transferability	👍		👍		👍
Quality in Use	Effectiveness				👍	
	Efficiency					
	Satisfaction					
	Safety					
	Usability	👍	👍			👍

METHODOLOGY

According to Creswell, there are five qualitative methods: narrative research, grounded theory research, phenomenological research, ethnographic research and case study research. This paper uses case study research, which is concerned with using real-life research methods as well as observations, interviews and reports. It is similar to social sciences research methods used in psychology, medicine and law. Case study research can fall into three categories: single instrumental, collective and intrinsic case study. Case selection can be challenging (Creswell, 2013).

In this paper, the authors apply Alnanih's new quality in use model (Alnanih, 2015), inspired by ISO/IEC 25010 (ISO/IEC 25010:2011), to assess two online banking software products: one open source and one closed source. First, the two software applications to be compared were selected: the OSS chosen was Cyclos and the closed source software was E-pay Suite. The goal of this work is to assess whether the open source product is capable of a similar quality of performance as that of the closed source product. ISO/25010 is an extension of the ISO/9126 standard, which forms the basis for most of the above models (Figure 6).

The qualitative approach will have four phases to be described in detail later in this section: pre-experiment, data gathering, data analysis and evaluation. We identify the tasks and the participants, prepare the spreadsheets and questionnaires and set the hypothesis in Phase One. In Phase Two, we gather data by recording the performance of each user and requiring them to complete the questionnaire. We analyze the data in Phase Three, calculating the quality in use metrics and testing the hypothesis. Finally, we evaluate the results of the computed metrics and evaluate user satisfaction in Phase Four.

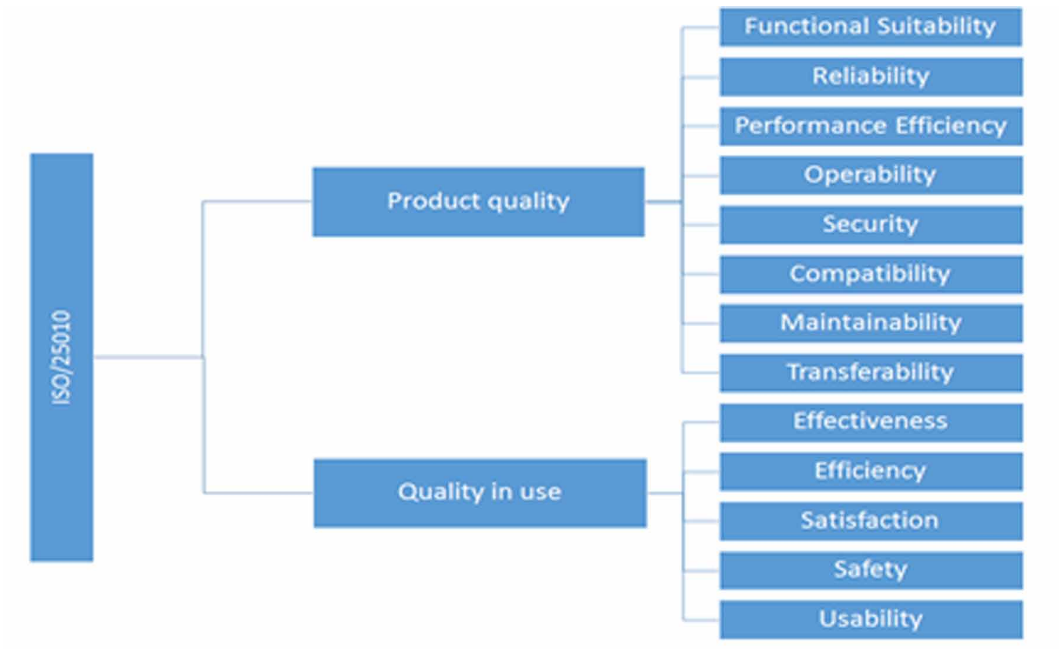
Proposed Quality in Use Approach

ISO/IEC 25010 (ISO/IEC 25010:2011) addresses two types of OSS product quality characteristics; namely, quality in use and product quality. This section discusses quality in use characteristics, as the evaluation of these characteristics in OSS is lacking in the literature.

Quality in use is defined as the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with efficiency, effectiveness, freedom from risk, and satisfaction in specific contexts of use (ISO/IEC 25010:2011)

In this work, Alnanih's new quality in use model is used (Alnanih, 2015). This model consists of the following quality in use characteristics (Figure 7):

Figure 6. ISO/IEC 25010 (ISO/IEC 25010:2011)



- **Effectiveness**: Completeness and accuracy for users in terms of achieving specified goals. Effectiveness is calculated as follows:

$$\frac{Min\ \#\, correct\, actions}{\#\, correct\, actions + \#\, incorrect\, actions}$$

- **Productivity**: The ratio of the functional value of the software produced to the labor and expense of producing it. Productivity is calculated as follows:

$$\frac{Min\ \#\, correct\, actions}{Time\, Period}$$

- **Efficiency**: Measures the resources consumed with respect to completeness and accuracy of achievement of user objectives. Efficiency is calculated as follows:

$$\frac{Effectiveness}{Time\, Period}$$

- **Error safety**: The degree to which the system prevents its users from making mistakes. Error safety is calculated as follows:

$$1-\left(\frac{\#\,incorrect\,actions}{\#\,correct\,actions+\#\,incorect\,actions}\right)$$

- **Cognitive load**: The inherent complexity of the task at hand. Cognitive load is calculated as follows:

$$\frac{\#\,views}{\#\,correct\,actions+\#\,incorrect\,actions}$$

Quality in Use Application

This case study aims to assess whether the OSS Cyclos (Social TRade Organisation, 1970) can be as effective as the closed source software E-pay Suite (Canopus Innovative Technologies, 1992). The authors therefore conducted an experimental assessment to compare the two software products on the basis of the stages shown in Figure 8.

Phase One: Pre-Experiment

In this phase, the authors organized the various parts of the experiment to achieve accurate results. For the aim of this work, the authors selected 10 tasks of different types that were available in both software products. Some tasks were purely related to banking, while others allowed users to customize their accounts. In addition, some tasks were designed to provide the user with a sense of security. The selected tasks were the following:

Step 1: Access personal information.
Step 2: Check previous transactions.
Step 3: Search for transactions occurring during the last month.
Step 4: Transfer $100 to any user.
Step 5: Change the language.
Step 6: Check the account balance.
Step 7: Save an account statement.
Step 8: Print an account statement.
Step 9: Check the details of the last login.
Step 10: Check messages.

These tasks were performed by ten professional participants who were expert users, which means that they were already familiar with online banking tasks (the group was mainly composed of PhD holders). A test on usability conducted by the authors in 1993 (see also Nielsen, 2012) obtained an average of p=0.31 for a number of studied projects. Therefore, 5 participants would be enough to detect 85% of the usability problems available at that test frequency. Virzi (1992) also designed a model on the basis of other projects, where p was found to fall between 0.32 and 0.42. Thus, 80% of the usability problems in a test can be detected with only 4 or 5 users. In the experiment, our expert users showed average to little resistance to using new technology and applications on their smartphones and computer platforms. Each one had used online banking for at least 10 years. Spreadsheets were prepared to organize the data gathered. A table was created for each user as well as for each software product.

To assess user satisfaction after working with both software products, users were asked to fill out a questionnaire containing two types of questions: general questions (e.g. gender), and specific questions that measured their satisfaction with both software products (e.g. recovery from error).

Figure 7. Alnanih New Quality-in-Use Model (Alnanih, 2015)

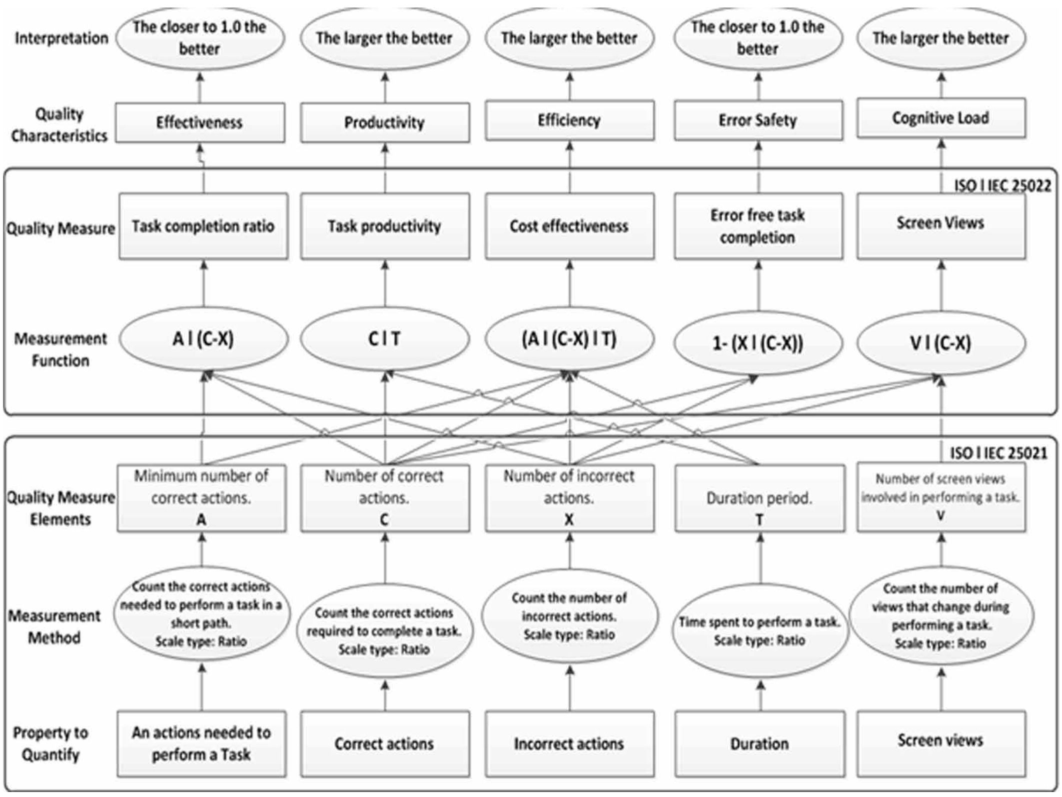
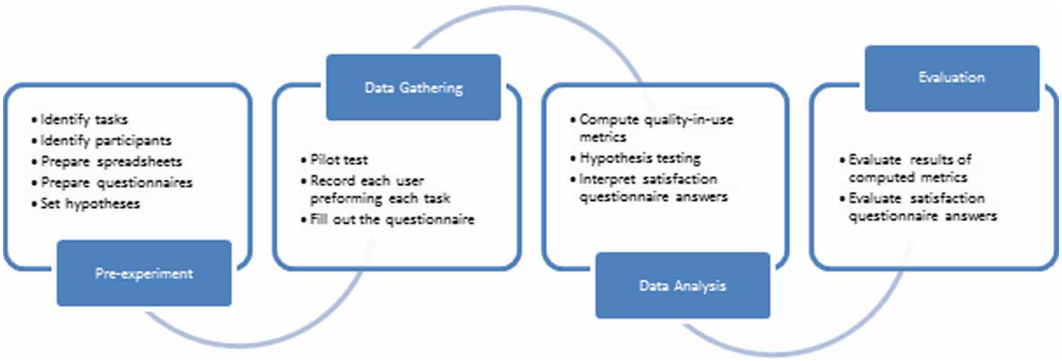

Interpretation
The closer to 1.0 the better
The larger the better
The larger the better
The closer to 1.0 the better
The larger the better
Quality Characteristics
Effectiveness
Productivity
Efficiency
Error Safety
Cognitive Load
ISO I IEC 25022
Quality Measure
Task completion ratio
Task productivity
Cost effectiveness
Error free task completion
Screen Views
Measurement Function
A I (C-X)
C I T
(A I (C-X) I T)
1- (X I (C-X))
V I (C-X)
ISO I IEC 25021
Quality Measure Elements
Minimum number of correct actions. A
Number of correct actions. C
Number of incorrect actions. X
Duration period. T
Number of screen views involved in performing a task. V
Measurement Method
Count the correct actions needed to perform a task in a short path. Scale type: Ratio
Count the correct actions required to complete a task. Scale type: Ratio
Count the number of incorrect actions. Scale type: Ratio
Time spent to perform a task. Scale type: Ratio
Count the number of views that change during performing a task. Scale type: Ratio
Property to Quantify
An actions needed to perform a Task
Correct actions
Incorrect actions
Duration
Screen views

Figure 8. Proposed quality in use methodology


• Identify tasks
• Identify participants
• Prepare spreadsheets
• Prepare questionnaires
• Set hypotheses
Pre-experiment
Data Gathering
• Pilot test
• Record each user preforming each task
• Fill out the questionnaire
• Compute quality-in-use metrics
• Hypothesis testing
• Interpret satisfaction questionnaire answers
Data Analysis
Evaluation
• Evaluate results of computed metrics
• Evaluate satisfaction questionnaire answers

In the experiment, the authors relied on data to either refute or support two hypotheses defined for the purpose of comparing the two software applications. The first hypothesis was a null hypothesis (a general statement to be refuted or supported), while the other was an alternative hypothesis that would be automatically supported if the null hypothesis was refuted.

a. Null hypothesis: There is no relationship between the two measured phenomena.
b. Alternative hypothesis: The opposite of the null hypothesis.

Below is an example of each of our hypotheses:

Effectiveness of Null-HYP: "There is no significant difference between the effectiveness of Cyclos and the effectiveness of E-pay Suite."
Effectiveness of Alt-HYP: "There is a significant difference between the effectiveness of Cyclos and the effectiveness of E-pay Suite."

Phase Two: Data Gathering

This phase constitutes the first transfer phase of the actual experiment, in which we gathered the data used to compare the two software. To ensure that we were ready to conduct the experiment, we first carried out some pilot tests in which we performed the exact steps that the participants would be conducting in the actual experiment. We varied the order in which the software was used by the participants in order to minimize the degree to which that affected their software learning experience, and took a laptop screenshot as each participant performed each of the ten tasks on each of the two software products. Then, for each task, we calculated the number of correct and incorrect actions that the user performed to complete the task. Also, we calculated the execution time for each task and counted the number of screenshots. We reported all this data in the spreadsheets. Having completed all the tasks, each participant was asked to fill out the questionnaire mentioned above.

Phase Three: Data Analysis

In this phase, the gathered data was analyzed to compare the values obtained for each software product. The metrics were computed using the above-mentioned formulas as well as the spreadsheet data. We used an Excel tool to calculate the value of the t-Test, which would be useful for comparing two groups of mean values. The average of all tasks was computed for each user, metric and software product. Then, the t-Test value was calculated using the average calculated for each software product. To interpret the results obtained from the questionnaires, a different type of analysis was used for each of the two types of questions. For the general assessment questions, the authors counted the answers, while for the questions related to ease of use, they set an objective scoring method that would not favor either software product. The two scoring scales are shown in Figure 9.

Phase Four: Evaluation

According to hypothesis testing convention, if the p-value obtained from the t-Test results is greater than alpha (i.e. 0.05), then the null hypothesis cannot be refuted for that factor. If the p-value is less than alpha, then the alternative hypothesis is supported, and the mean value should be used for the comparison between the two software products.

To evaluate the participants' answers, the authors compared the values obtained in the results. In this comparison, the focus was on noticeable differences (greater than 2).

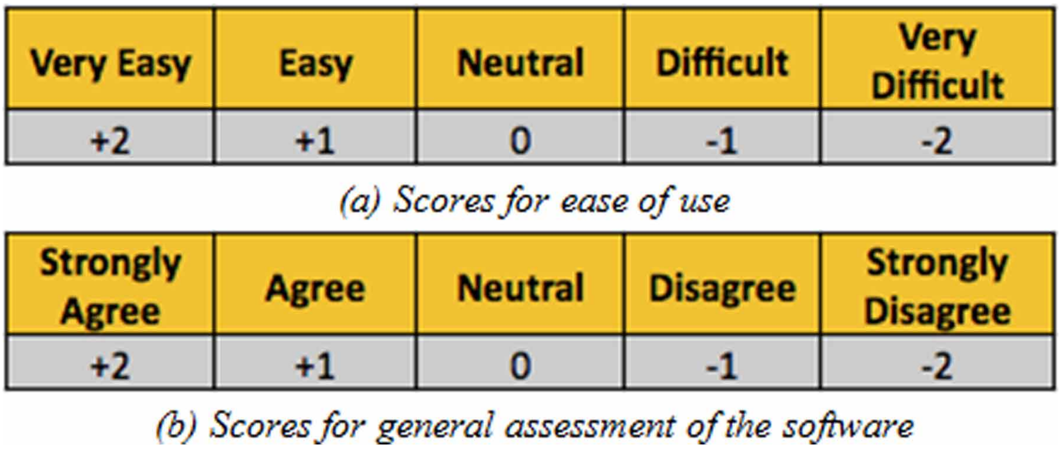
EXPERIMENT RESULTS

To represent the findings, the authors computed the mean value of all averages in each metric for both software products. The results are shown in Figure 10.

Below, the obtained results are explained to indicate whether the hypotheses are supported or not.

- **Effectiveness:** The obtained p-value is greater than the alpha value, which means that Effectiveness Null-HYP is not refuted. In other words, there is no significant difference between the effectiveness of Cyclos and that of E-pay Suite. The effectiveness results of the t-Test are shown in Figure 11.

Figure 9. Scoring method



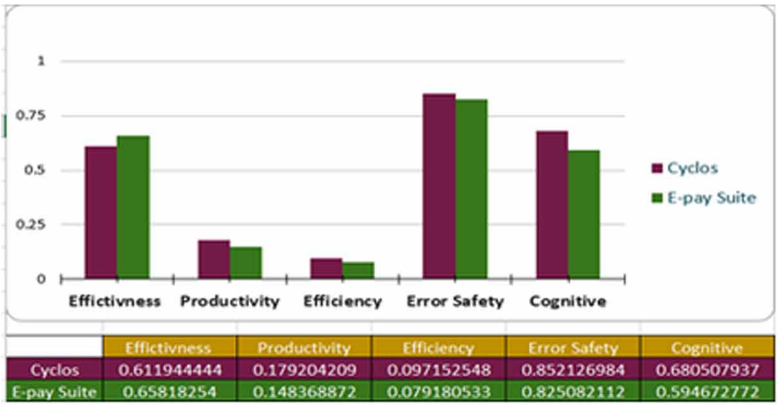
Very Easy	Easy	Neutral	Difficult	Very Difficult
+2	+1	0	-1	-2

(a) Scores for ease of use

Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
+2	+1	0	-1	-2

(b) Scores for general assessment of the software

Figure 10. Overall comparison of Cyclos and E-pay Suite



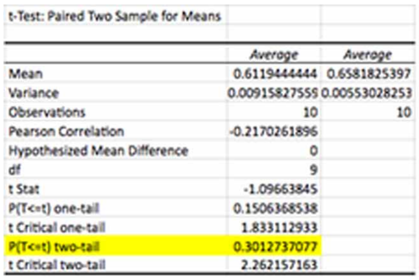
	Effictivness	Productivity	Efficiency	Error Safety	Cognitive
Cyclos	0.611944444	0.179204209	0.097152548	0.852126984	0.680507937
E-pay Suite	0.65818254	0.148368872	0.079180533	0.825082112	0.594672772

- **Productivity:** The obtained p-value is smaller than the alpha value, which means that Productivity Null-HYP is refuted and the mean value is used to compare the two software products. The mean value of the productivity of Cyclos is greater than the mean value of the productivity of E-pay Suite, as shown in Figure 12. In other words, the productivity of Cyclos is significantly superior to that of E-pay Suite. The productivity results of the t-Test are shown in Figure 12.

- **Efficiency:** The obtained p-value is greater than the alpha value, which means that Efficiency Null-HYP is not refuted. In other words, there is no significant difference between the efficiency of Cyclos and that of E-pay Suite. The efficiency results of the t-Test are shown in Figure 13.

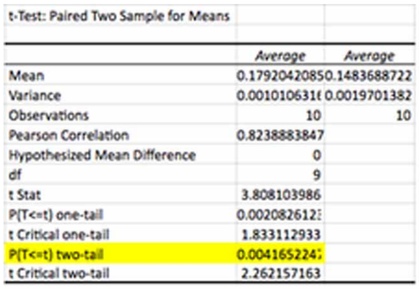
- **Error Safety:** The obtained p-value is greater than the alpha value, which means that Error Safety Null-HYP is not refuted. In other words, there is no significant difference between the error safety of Cyclos and that of E-pay Suite. The error safety results of the t-Test are shown in Figure 14.

Figure 11. t-Test results for Effectiveness



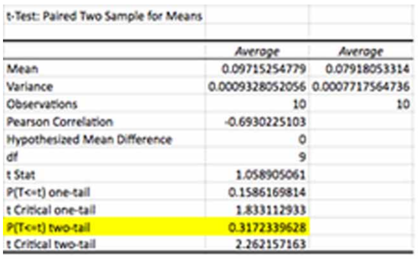
t-Test: Paired Two Sample for Means		
	Average	*Average*
Mean	0.6119444444	0.6581825397
Variance	0.00915827559	0.00553028253
Observations	10	10
Pearson Correlation	-0.2170261896	
Hypothesized Mean Difference	0	
df	9	
t Stat	-1.09663845	
P(T<=t) one-tail	0.1506368538	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.3012737077	
t Critical two-tail	2.262157163	

Figure 12. t-Test results for Productivity



t-Test: Paired Two Sample for Means		
	Average	*Average*
Mean	0.1792042085	0.1483688722
Variance	0.001010631	0.0019701382
Observations	10	10
Pearson Correlation	0.8238883847	
Hypothesized Mean Difference	0	
df	9	
t Stat	3.808103986	
P(T<=t) one-tail	0.002082612	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.004165224	
t Critical two-tail	2.262157163	

Figure 13. t-Test results for Efficiency



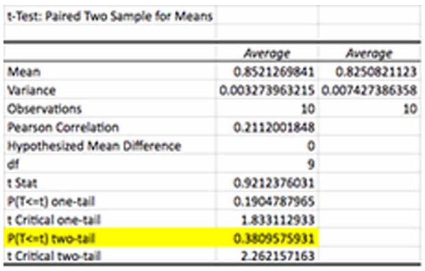
t-Test: Paired Two Sample for Means		
	Average	*Average*
Mean	0.09715254779	0.07918053314
Variance	0.0009328052056	0.0007717564736
Observations	10	10
Pearson Correlation	-0.6930225103	
Hypothesized Mean Difference	0	
df	9	
t Stat	1.058905061	
P(T<=t) one-tail	0.1586169814	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.3172339628	
t Critical two-tail	2.262157163	

- **Cognitive Load:** The recorded p-value is smaller than the alpha value, which means that Cognitive Load Null-HYP is not refuted and the mean is used to compare the two software products. The mean value of the cognitive load of Cyclos is greater than that of E-pay Suite, as shown in Figure 15. In other words, Cyclos has a significantly higher cognitive load than E-pay Suite. The cognitive load results of the t-Test are shown in Figure 15.

The results clearly show that Cyclos and E-pay Suite have similar performance in terms of efficiency, effectiveness and error safety. However, Cyclos outperforms E-pay Suite slightly when it comes to productivity and cognitive load.

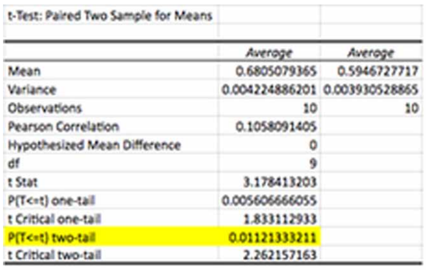
The questionnaire results revealed that the majority of participants were male and that most of them were educated at a PhD level. Most of them had experience with online banking, having used online banking services frequently for more than two years. In terms of user satisfaction, the questionnaire results showed that, in general, the participants were equally satisfied with both software

Figure 14. t-Test results For Error Safety



t-Test: Paired Two Sample for Means		
	Average	*Average*
Mean	0.8521269841	0.8250821123
Variance	0.003273963215	0.007427386358
Observations	10	10
Pearson Correlation	0.2112001848	
Hypothesized Mean Difference	0	
df	9	
t Stat	0.9212376031	
P(T<=t) one-tail	0.1904787965	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.3809575931	
t Critical two-tail	2.262157163	

Figure 15. t-Test results for Cognitive Load



t-Test: Paired Two Sample for Means		
	Average	*Average*
Mean	0.6805079365	0.5946727717
Variance	0.004224886201	0.003930528865
Observations	10	10
Pearson Correlation	0.1058091405	
Hypothesized Mean Difference	0	
df	9	
t Stat	3.178413203	
P(T<=t) one-tail	0.005606666055	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.01121333211	
t Critical two-tail	2.262157163	

products. However, they found it easier to navigate through E-pay Suite's interface. Nevertheless, the overwhelming majority of the participants would prefer to use Cyclos over E-pay Suite.

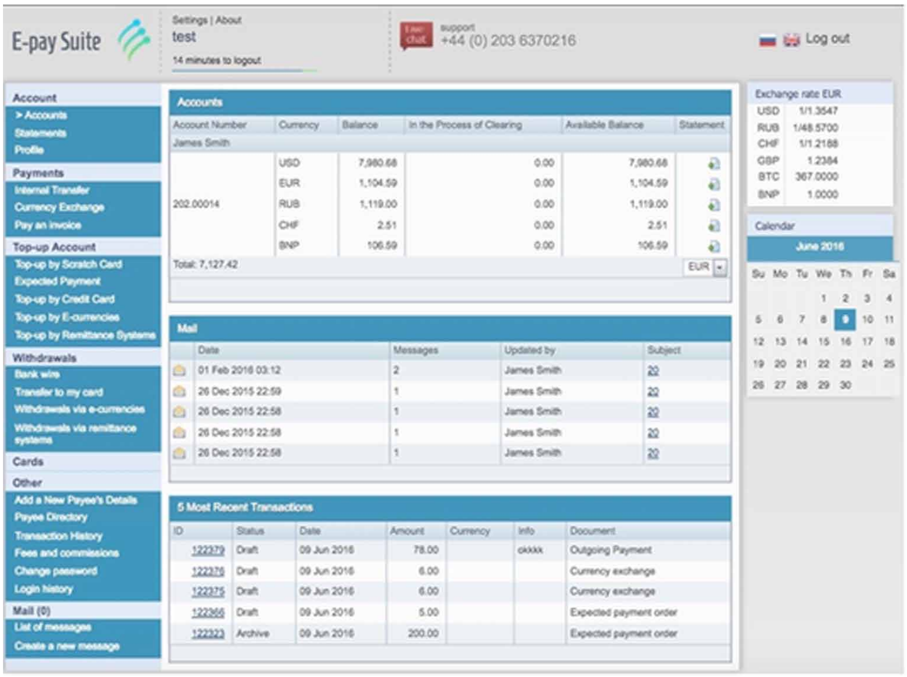
DISCUSSION

During the experiment, the authors observed that the participants faced some difficulties while performing tasks in both Cyclos and E-pay Suite. In this section, these difficulties are discussed in order to highlight weaknesses in the two software products. In addition, positive feedback from the participants about both products is noted.

The user interface of E-pay Suite displays menus and submenus on the left side of the window. At the center of the window, the content of each submenu is presented. Figure 16 shows the E-pay Suite interface layout.

Some participants noted that personal information is not organized logically, and some found it challenging to understand the meaning of the language icons in E-pay Suite, which they felt should be accompanied by suitable abbreviations. In addition, participants were confused about which button to click on to complete a money transfer: "Save" or "Sign". The terms "Submit", "Confirm'', or "Transfer" seemed to them to be more appropriate than "Sign" to describe this task. In addition, a list of frequent recipients is not provided for the user when entering the recipient's name. While it is easy to reach an account statement by clicking on a button in the account menu in E-pay Suite, the list of transactions is displayed in ascending order, whereas some of the participants would prefer to have them listed in descending order. Finally, in order to access the functions available in E-pay Suite, the user is required to click on the text describing a side menu item (i.e. the item's name), instead of clicking anywhere in the region of the item. Participants found this frustrating since this step requires precise clicking.

Figure 16. Graphical User Interface of E-pay Suite



Accounts

Account Number	Currency	Balance	In the Process of Clearing	Available Balance	Statement
James Smith					
	USD	7,980.68	0.00	7,980.68	
	EUR	1,104.59	0.00	1,104.59	
202.00014	RUB	1,119.00	0.00	1,119.00	
	CHF	2.51	0.00	2.51	
	BNP	106.59	0.00	106.59	

Total: 7,127.42

Exchange rate EUR

USD	1/1.3547
RUB	1/48.5700
CHF	1/1.2188
GBP	1.2384
BTC	367.0000
BNP	1.0000

Calendar

June 2016

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Mail

Date	Messages	Updated by	Subject
01 Feb 2016 03:12	2	James Smith	20
26 Dec 2015 22:59	1	James Smith	20
26 Dec 2015 22:58	1	James Smith	20
26 Dec 2015 22:58	1	James Smith	20
26 Dec 2015 22:58	1	James Smith	20

5 Most Recent Transactions

ID	Status	Date	Amount	Currency	Info	Document
122379	Draft	09 Jun 2016	78.00		okkkk	Outgoing Payment
122376	Draft	09 Jun 2016	6.00			Currency exchange
122375	Draft	09 Jun 2016	6.00			Currency exchange
122366	Draft	09 Jun 2016	5.00			Expected payment order
122323	Archive	09 Jun 2016	200.00			Expected payment order

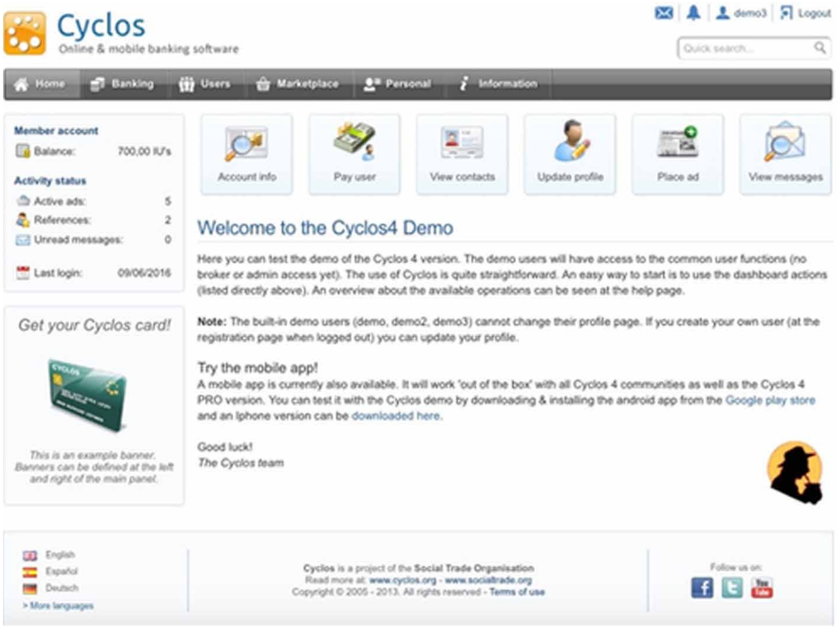
In contrast, the user interface of Cyclos offers two ways to access the functions provided, either from the menu bar or via dashboard icons. Each menu item or icon has a side menu (submenu) that provides related functions. The layout of the Cyclos user interface is shown in Figure 17.

On the Cyclos user interface, displaying the icons on the dashboard simplifies access to various functions; however, participants found that this made the home page appear crowded. Moreover, they found it difficult to remember how to access specific functions, given that there are two ways to access them. This is not an issue in E-pay Suite since all functions are displayed on the side of the window. In Cyclos, the names of menu items are not always clear (e.g. the terms "Personal" and "Information" were easily confused by participants when they requested access to their personal information). The participants struggled to save or print an account statement since the combined Save/Print button is quite small. These buttons are separate in E-pay Suite and easy to find. The Cyclos interface allows the user to change the language of the page via icons with the name of the language displayed, a feature that is not provided on the E-pay Suite interface. Even so, most of the participants were unable to find the language settings easily in Cyclos, since they are outside the window range. Another issue with Cyclos—a key one—was that the software does not allow the user to transfer a particular amount ($100 is the minimum amount that can be transferred). However, Cyclos displays search results for specific transactions immediately, as expected by the participants, while E-pay Suite does not.

CONCLUSION AND FUTURE WORK

The banking sector is a critical sector that deals with highly sensitive data. The accuracy and effectiveness of the software used in the sector are therefore critical issues. In this paper, a comprehensive set of available quality models and their application to OSS was proposed. The authors applied a new quality in use model (Alnanih, 2015), inspired by ISO/IEC 25010 (ISO/IEC

Figure 17. Graphical User Interface of Cyclos



Member account

Balance:	700,00 IU's

Activity status

Active ads:	5
References:	2
Unread messages:	0
Last login:	09/06/2016

Get your Cyclos card!

This is an example banner. Banners can be defined at the left and right of the main panel.

Welcome to the Cyclos4 Demo

Here you can test the demo of the Cyclos 4 version. The demo users will have access to the common user functions (no broker or admin access yet). The use of Cyclos is quite straightforward. An easy way to start is to use the dashboard actions (listed directly above). An overview about the available operations can be seen at the help page.

Note: The built-in demo users (demo, demo2, demo3) cannot change their profile page. If you create your own user (at the registration page when logged out) you can update your profile.

Try the mobile app!

A mobile app is currently also available. It will work 'out of the box' with all Cyclos 4 communities as well as the Cyclos 4 PRO version. You can test it with the Cyclos demo by downloading & installing the android app from the Google play store and an Iphone version can be downloaded here.

Good luck!
The Cyclos team

Cyclos is a project of the **Social Trade Organisation**
Read more at: www.cyclos.org - www.socialtrade.org
Copyright © 2005 - 2013. All rights reserved - Terms of use

25010:2011), through a series of phases that were designed to assess two online banking software applications: Cyclos (open source) and E-pay Suite (closed source). These phases included data gathering, data analysis and interpretation/evaluation of results. The results prove that the performance of Cyclos is comparable to that of E-pay Suite, based on a recent quality in use model. At the end of the experiment, the authors were able to conclude that Cyclos is not only as efficient and effective as E-pay Suite, but that it is more productive. Moreover, the results of the questionnaires filled out by the participants showed that, from a user standpoint, Cyclos works well, although the participants favored E-pay Suite for ease of navigation of its user interface. However, Cyclos is open source software, which means that its user interface is fully customizable and can easily be improved.

Proving that OSS products have the capabilities to perform as well as if not better than closed source software competitors indicates that there is great potential in open source technology. In the future, the authors plan to evaluate OSS from different fields and compare the results with those obtained from the banking sector. This will bring to light the reliability of open source technology. Another possible direction is to design a new quality in use model that fits the characteristics of both closed and open source software.

REFERENCES

Akbari, M., & Peikar, S. R. H. (2014). Evaluation of free/open source software using osmm model case study: Webgis and spatial database. *Advances in Computer Science: an International Journal*, *3*(5), 34–43.

Alnanih, R. (2015). *CON-INFO: A Context-based Methodology for Designing and Assessing the Quality of Adaptable MUIs in Healthcare Applications* [Doctoral dissertation]. Concordia University.

Aversano, L., & Tortorella, M. (2010, June). Evaluating the quality of free/Open source systems: A case study. *Proceedings of the International Conference on Enterprise Information Systems* (pp. 119-134). Springer.

Aversano, L., & Tortorella, M. (2011, June). Applying EFFORT for evaluating CRM open source systems. *Proceedings of the International Conference on Product Focused Software Process Improvement* (pp. 202-216). Springer. 10.1007/978-3-642-21843-9_17

Aversano, L., & Tortorella, M. (2013). Quality evaluation of floss projects: Application to ERP systems. *Information and Software Technology*, *55*(7), 1260–1276. doi:10.1016/j.infsof.2013.01.007

Bevan, N. (2001). Quality in use for all. In C. Stephanidis (Ed.), *User Interfaces for All: methods, concepts and tools* (pp. 353–368). CRC Press.

Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., & Merritt, M. (1978). *Characteristics of software quality*. TRW Series of Software Technology.

Canopus Innovative Technologies. (1992). E-pay Suite. Retrieved from http://epaysuite.com/

Creswell, J. W. (2013). *Qualitative Inquiry & Research Design: Choosing Among Five Approaches* (3rd ed.). Thousand Oaks, CA: SAGE Publications, Inc.

De Groot, A., Kügler, S., Adams, P. J., & Gousios, G. (2006, June). Call for quality: Open source software quality observation. *Proceedings of the IFIP International Conference on Open Source Systems* (pp. 57-62). Springer. 10.1007/0-387-34226-5_6

Dromey, R. G. (1995). A model for software product quality. *IEEE Transactions on Software Engineering*, *21*(2), 146–162.

Duijnhouwer, F. and Widdows, C. (2003). Open source maturity model.

Groven, A. K., Haaland, K., Glott, R., & Tannenberg, A. (2010, August). Security measurements within the framework of quality assessment models for free/libre open source software. In *Proceedings of the fourth european conference on software architecture: Companion volume* (pp. 229-235). ACM.

Hecht, L., & Clark, L. (2018). *Survey: Open Source Programs Are a Best Practice Among Large Companies*. The New Stack.

Houaich, Y. A., & Belaissaoui, M. (2015, February). Measuring the maturity of open source software. *Proceedings of the 2015 6th International Conference on Information Systems and Economic Intelligence (SIIE)* (pp. 133-140). IEEE. 10.1109/ISEI.2015.7358735

ISO. (2011). ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Retrieved from https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:en

McCall, J. A. (1977). *Factors in software quality. General Electric*. National Technical Information Service.

Nielsen, J. (2012). Usability 101: Introduction to usability. NN Group. Retrieved from http://www.nngroup.com/articles/usability-101-introduction-to-usability/

Nielsen, J., & Landauer, T. K. (1993) A Mathematical Model of the Finding of Usability Problems. *Proceedings of INTERCHI '93*. Academic Press.

Official Statistics of Finland (OSF). (2011) Use of information technology in enterprises. Retrieved from http://www.stat.fi/til/icte/2011/icte_2011_2011-11-24_tie_001_en.html

Popp, K. M. (Ed.). (2015). *Best Practices for commercial use of open source software: Business models, processes and tools for managing open source software*. BoD–Books on Demand.

Samoladas, I., Gousios, G., Spinellis, D., & Stamelos, I. (2008, September). The SQO-OSS quality model: measurement based open source software evaluation. *Proceedings of the IFIP International Conference on Open Source Systems* (pp. 237-248). Springer. 10.1007/978-0-387-09684-1_19

Social TRade Organisations. (1970). Cyclos. Retrieved from https://www.cyclos.org/

Soto, M., & Ciolkowski, M. (2009, October). The QualOSS open source assessment model measuring the performance of open source communities. *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement ESEM 2009* (pp. 498-501). IEEE. 10.1109/ESEM.2009.5314237

Soto, M., & Ciolkowski, M. (2009, September). The QualOSS Process Evaluation: Initial Experiences with Assessing Open Source Processes. *Proceedings of the European Conference on Software Process Improvement* (pp. 105-116). Springer. 10.1007/978-3-642-04133-4_9

Virzi, R. A. (1992). Refining the test phase of usability evaluation: How many subjects is enough? *Human Factors*, *34*(4), 457–468. doi:10.1177/001872089203400407

Wasserman, A. I., Pal, M., & Chan, C. (2006). Business readiness rating for open source. *Proceedings of the EFOSS Workshop*, Como, Italy. Academic Press.

Wasserman, T., & Das, A. (2007). Using FLOSSmole data in determining business readiness ratings.

Manar Abu Talib is teaching at the University of Sharjah in the UAE. Dr. Abu Talib's research interest includes software engineering with substantial experience and knowledge in conducting research in software measurement, software quality, software testing, ISO 27001 for Information Security and Open Source Software. Manar is also working on ISO standards for measuring the functional size of software and has been involved in developing the Arabic version of ISO 19761 (COSMIC-FFP measurement method). She published more than 40 refereed conferences, journals, manuals and technical reports, involved in more than 200 professional activities and sponsored research activities and supervised 30 capstone projects. She received the Best Teacher Award twice, the Exemplary Faculty Award in 2008 and 2010, the Google CS4HS Award in 2014, QCRI ArabWIC and Anita Borg Institute Faculty scholarships in 2015, outstanding University & Community Service Award in 2016 and Exemplary Leader Award in WiSTEM 2016. She was the Counselor of IEEE Student Branch at Zayed University, 2012-2013 and founder and former CEO of Emirates Digital Association for Women (EDAW111). She is the ArabWIC VP of Chapters in Arab Women in Computing Association (ArabWIC), an executive member in UAE IEEE Section, the Sharjah Google Developer Group Manager, the UAE representative for the COSMIC-FPP Education Committee and the International Collaborator to Software Engineering Research Laboratory in Montreal, Canada.

Areej Alsaafin is a master's student in computer science at the University of Sharjah, UAE. She received her B.A. of computer engineering from the University of Sharjah in 2015. Her main research interests are wireless sensor networks, data mining, open source software, and natural language processing.

Selma Manel Medjden is pursuing her master's degree in computer science at the University of Sharjah in the UAE. She graduated with a bachelor's degree in information systems and Software Engineering from the University of Sciences and Technology Houari Boumediene in Algeria in 2013. Her main research interests are machine learning, pattern recognition, and sign language recognition.