Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

PCyTI

UNIVERSIDAD AUTÓNOMA METROPOLITANA – IZTAPALAPA
DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

MODELO DE REFERENCIA PARA LA ESTIMACIÓN DE VALOR DEL PRODUCTO SOFTWARE DURANTE EL PROCESO DE DESARROLLO

Tesis que presenta
Oscar Jesús Castro López
Para obtener el grado de
Maestro en ciencias y tecnologías de la información

Asesores: Dra. Angelina Espinoza Limón
M.C. Alfonso Martínez Martínez

Jurado calificador:

Presidente: M.T.I.A. Elsa Ramírez Hernández
Secretaria: Dra. Angelina Espinoza Limón
Vocal: M.C. Luis Fernando Castro Careaga

México, D.F. Mayo 2013

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

PCyTI

UNIVERSIDAD AUTÓNOMA METROPOLITANA – IZTAPALAPA
DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

MODELO DE REFERENCIA PARA LA ESTIMACIÓN DE VALOR DEL PRODUCTO SOFTWARE DURANTE EL PROCESO DE DESARROLLO

Tesis que presenta
Oscar Jesús Castro López
Para obtener el grado de
Maestro en ciencias y tecnologías de la información

Asesores:	Dra. Angelina Espinoza Limón
	M.C. Alfonso Martínez Martínez

Jurado calificador:	
Presidente:	M.T.I.A. Elsa Ramírez Hernández
Secretaria:	Dra. Angelina Espinoza Limón
Vocal:	M.C. Luis Fernando Castro Careaga

México, D.F. Mayo 2013

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

PCyTI

REFERENCE MODEL FOR THE SOFTWARE PRODUCT VALUE ESTIMATION IN THE DEVELOPMENT PROCESS

A Thesis
Presented to
The Academic Faculty

by

Oscar Jesús Castro López

In Partial Fulfillment
of the Requirements for the Degree
Master in Science and Technology Information in the
Universidad Autónoma Metropolitana

Universidad Autónoma Metropolitana
Posgrado en ciencias y tecnologías de la información
May 2013

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

PCyTI

REFERENCE MODEL FOR THE SOFTWARE PRODUCT VALUE ESTIMATION IN THE DEVELOPMENT PROCESS

Approved by:

Dra. Angelina Espinoza Limón
Electrical Engineering Department
Universidad Autónoma Metropolitana-Iztapalapa (UAM-I)

M.Cs. Alfonso Martínez Martínez
Laboratorio de Investigación en Informática Médica (LIIM) Area de Procesamiento Digital de Señales e Imágenes Biomédicas
Departamento de Ing. Eléctrica
Universidad Autónoma Metropolitana-Iztapalapa (UAM-I)

M.T.I.A. Elsa Ramírez Hernández
Dirección de Innovación y Calidad
Praxis

M.Cs. Luis Fernando Castro Careaga
Área de Computación y Sistemas
Universidad Autónoma Metropolitana-Iztapalapa (UAM-I)

Date Approved: May 06th, 2013

I lovingly dedicate this thesis to Betty and my parents, for supporting me each step of the way, this work is as much mine as it is theirs.

ACKNOWLEDGEMENTS

I want to thank my advisors: Angelina Espinoza Limón and Alfonso Martínez Martínez, for all the time dedicated to my work, and for their generous guidance and mentoring.

To the approval commite: Elsa Ramírez Hernández and Luis Fernando Castro Careaga, who were of great assistance in taking the time to review and contribute to the content of my work.

To the PCyTI professors who shared their knowledge and experience.

To the Consejo Nacional de Ciencia y Tecnología (CONACyT) for the finantial support to conduct this thesis.

To my friends who have supported me throughout the process. I will always appreciate all they have done.

Thanks to all the members of my big family, specially to: Nereyda, Oscar, Cinthia, Fernanda and the newest member Giuliana.

Special thanks to Betty, for her unconditional love, her words of encouragement and the support she gave me, which have been fundamental for me to finish my thesis.

Contents

List of Tables

List of Figures

LIST OF ABBREVIATIONS

SMS	Systematic Mapping Study
RESVEP	Reference Model for the Software Product Value Estimation Process
IEEE	Institute of Electrical and Electronics Engineers
ACM	Association for Computing Machinery
ISO	International Organization for Standardization
SEI	Software Engineering Institute
SE	Software Engineering
PiD	Products in Development
PCDVE	Project Context Definition for the Value Estimation
QE	Quality Estimation
VQIA	Value Quality Indicators Assingment
VQIM	Value Quality Indicators Measurement
QC	Quality Calculation
FE	Function Estimation
CE	Cost Estimation
VE	Value Estimation
SQuaRe	Software Product Quality Requirements and Evaluation
EX	Expected Measure
ES	Estimated Measure
Q	Quality
VQI	Value Quality Indicator
RVQI	Ratio Value Quality Indicator
P	Portability
U	Usability
FS	Functional Suitability

M	Maintainability
S	Security
R	Reliability
PE	Performance Efficiency
ISPC	Instrument System for Preparatory Courses
UMC	User Management Component
V	Votes (in the UMC case study)
CP	Customer Priority (in the UMC case study)
AP	Architect Priority (in the UMC case study)
PQ	Product Q
FP	Function Points
MH	Man-Hours

RESUMEN

El valor de los productos software es un factor muy importante para posicionarlos en el competitivo mercado que existe hoy en día. En el mercado de productos software el elemento valor es tomado en cuenta cada vez mas, es importante señalar que el valor no es lo mismo que el precio, ya que aumentar o disminuir el precio no modifica el valor del producto, en cambio modifica la percepción del cliente o incentiva al cliente para comprar o no comprar un producto de software.

Los clientes hacen una gran inversión al adquirir productos de software, y cada vez son más consientes de qué es lo que están comprando, entonces los clientes se encuentran en búsqueda de una solución de software que ofrezca un alto valor para satisfacer sus necesidades a cambio de su dinero.

A su vez las compañías desarrolladoras de productos software también invierten grandes cantidades de dinero pero para desarrollar sus productos y posicionarlos en el mercado. Las compañías desarrolladoras que satisfacen las necesidades del cliente y a su vez ofrecen productos con un alto valor están en una mejor posición para competir y tener éxito en el mercado.

Sin embargo, un problema con el que se encuentran las compañías de software es que no tienen un sólido entendimiento del factor valor en sus productos de una manera cuantificable, y por ende estos productos no llenan sus expectativas económicas.

Una compañía puede comenzar el desarrollo de un producto de software y en el plan de proyecto éste es plasmado como una idea brillante e innovadora, pero a medida que el proceso de desarrollo se lleva a cabo, muy diversos problemas técnicos surgen y una validación del valor de dicho producto rara vez es realizada. Al final se obtiene un producto mediocre y sus posibilidades de éxito en el mercado son prácticamente

nulas.

Existe un enlace perdido entre la planeación de un producto de alto valor y el producto realmente desarrollado: el plan de producto no es reflejado en el producto final.

Para enlazar el valor del plan de producto con el producto final, es necesario que las empresas cuenten con conocimiento y métodos que ayuden a estimar y validar el valor del producto software. La estimación y validación permitiría a las empresas observar cómo el valor de sus productos varían entre aplicaciones y qué es lo que se ofrece al cliente.

Por lo tanto, el valor debe ser tomado en consideración cuando se usan técnicas de ingeniería de software para desarrollar productos software, de forma similar como el costo, planeación, recursos y otros aspectos importantes son considerados regularmente para el desarrollo de software. [3].

Es precisamente en las áreas de desarrollo de software en donde problemas técnicos están afectando el valor del producto lanzado al mercado. Por lo tanto, las compañías desarrolladoras necesitan métodos para estimar el valor del producto software durante el proceso de desarrollo, con el objeto de obtener datos cuantificables y visibilidad sobre el valor del producto software que será lanzado al mercado.

De acuerdo con diversos autores, el valor es altamente perceptual ([2], [6], [7]), por lo tanto hay que identificar los elementos que definen el valor de productos software. Estos elementos deben ser plenamente identificables y cuantificables.

Existen distintos enfoques para estimar el valor del producto software durante el ciclo de desarrollo, pero están principalmente enfocados a la fase de requerimientos de software, como Barney et al. en [2] en el cual los autores se refieren a una propuesta basado en valor de ingeniería de requerimientos, enfocándose en el proceso de crear valor a través de la selección de requerimientos para un lanzamiento de software. De manera similar, Ojala en [19] ofrece una propuesta para la evaluación de valor de productos principalmente a través de la priorización de requerimientos y atributos de calidad, asignando porcentajes a los atributos de calidad de acuerdo a la opinión de involucrados. Pero estas propuestas no están enfocadas a la estimación de valor de productos software. Estos no ofrecen visibilidad para la administración del valor, ni tampoco ofrecen datos cuantitativos acerca de la estimación de valor durante el proceso de desarrollo de software.

Se encontraron diferentes definiciones de valor, todas válidas en su contexto de investigación, pero ninguna especifica para valor de productos software. Por lo tanto en ésta tesis se propone una definición de valor, pero específica para productos de valor. La definición de valor propuesta considera los factores que influencian el valor en productos finales o intermedios. La definición esta basada en los trabajos relacionados de las conclusiones del estado del arte de ésta tesis (Sección 2.3). La definición que se propone es la siguiente:

Valor es una medida - por lo general en dinero, esfuerzo o intercambio, o en una escala comparativa - de bienes de software o servicios (conjunto de programas, procedimientos, algoritmos y su documentación) que satisfagan las necesidades, deseos y expectativas de los usuarios. Todos estos bienes o servicios son influenciados por los atributos de calidad del producto software.

Basados en ésta definición de valor especifica para productos software, se retoma la ecuación de Ojala en [19] la cual a su vez está basada en un estándar de SAVE [23], que incluye al factor de calidad. Por lo tanto, esta tesis propone utilizar la misma ecuación propuesta por Ojala 1 (introducida en la Sección 2.2.2), pero modificando ligeramente sus elementos para alinearlos a los objetivos de ésta tesis.

$$Valor = \frac{Funcionalidad + Calidad}{Costo} \tag{1}$$

En donde:

- **Funcionalidad** es el trabajo especifico que un diseño o ítem (producto, servicio o proceso) debe realizar (Ej. de trabajo: puntos función, necesarios para desarrollar un componente de software).

- **Calidad** es la necesidad, deseos y expectativas del usuario (Ej.: atributos de calidad de software como: usabilidad, portabilidad, seguridad, trazabilidad, etc.).

- **Costo** es la cantidad de esfuerzo necesario para desarrollar el producto, servicio o proceso (Ej.: el esfuerzo expresado en horas-hombre).

Como ya se ha mencionado anteriormente, el valor esta altamente relacionado a la percepción. Entonces, la identificación y cuantificación de los elementos que contribuyen a disminuir o aumentar el valor proveerán de un método formal y sistemático para obtener una percepción mas acercada a lo real en cuanto a valor de producto software.

Particularmente para productos software estos elementos de valor deben ser gestionados durante el proceso de desarrollo. Por lo tanto en ésta tesis se propone la definición de Indicadores de Valor del Producto Software (basada en [20]), los indicadores de valor son los elementos que definen al valor. La definición es la siguiente:

Es un medio para proporcionar información específica y cuantitativa sobre el estado o la condición del valor del producto de software.

Los indicadores son desarrollados basados en medidas cuantitativas o estadísticas de aquellos elementos que agregan/destruyen el valor de los productos de trabajo. En este sentido, los indicadores permiten obtener visibilidad sobre el valor que es realmente ofrecido al cliente.

Esta tesis propone un modelo de referencia para el proceso de estimación de valor (RESVEP), el cual ha sido desarrollado considerando los resultados del análisis del estado del arte (Capitulo 2) con respecto a la estimación de valor en el área de ingeniería de software. RESVEP es un marco abstracto que se compone de un conjunto de procedimientos (tales como actividades, productos de trabajo, sub-procesos, documentos, o tareas) que están relacionadas a través de un flujo de trabajo establecido.

El RESVEP describe el proceso para realizar una estimación de valor de productos de trabajo en un proyecto de software. El proceso de la estimación de valor del producto software esta dividido en las siguientes fases:

1. **La definición del contexto del proyecto para la estimación de valor (PCDVE).** Es de suma importancia conocer las circunstancias y datos que rodean el proyecto en donde se va a aplicar el RESVEP, debido a que el contexto del proyecto debe ser definido para establecer diversos aspectos necesarios para el proceso de estimación de valor del producto de desarrollo. Estos aspectos pueden ser el conjunto de indicadores, las mediciones y aplicación de ecuaciones. En este punto se definen fases (disciplinas o procesos, esto depende del modelo de proceso de desarrollo utilizado), actividades, involucrados y productos de trabajo.

2. **Estimación de calidad (QE).** Esta es la parte mas difícil de medir, ya que los atributos de calidad varían y dependen de la percepción de los involucrados en definir la calidad del producto. Los atributos de calidad que son complicados de cuantificar, por ejemplo: usabilidad, portabilidad, reusabilidad o eficiencia. Ésta tesis provee una propuesta para estimar la calidad del software, mayormente basada en *Software engineering-Software product Quality Requirements and Evaluation (SQuaRE)Quality model* [12]. La estimación de calidad se compone de los siguientes pasos:

   2.1 **Asignación de los indicadores de calidad de valor(VQIA).** En este paso se asignan un conjunto de indicadores de calidad a cada producto de trabajo. En este sentido los indicadores de calidad dependen de los atributos de calidad del tipo de producto de trabajo. El RESVEP se enfoca en definir indicadores de calidad para obtener mediciones de calidad (y por ende de valor). Estos indicadores son usados para identificar si el producto va en el camino correcto para alcanzar el valor esperado de acuerdo al plan de producto diseñado por el área de negocios. Los atributos de calidad afectan la calidad en general, y calidad tiene un alto impacto en el valor, por lo tanto en el contexto de esta tesis, los indicadores de calidad son considerados indicadores de valor.

2.2 **Medición de indicadores de calidad (VQIM).** En este paso se le asignan mediciones a cada indicador de calidad. Se realizan estas mediciones y se aplican las ecuaciones correspondientes para obtener datos cuantitativos. Las mediciones y ecuaciones están principalmente basadas en el estándar ISO 9126 [10].

2.3 **Calculo de Calidad (QC).** El calculo de calidad esta basado en el conjunto de indicadores de calidad que son asignados a un producto (Paso 2.1). Después de las mediciones y asignación de ecuaciones (Paso 2.2) para los indicadores de calidad del producto (de acuerdo con los atributos de calidad del producto), entonces la ecuación correspondiente es aplicada para obtener una medida cuantitativa para la calidad del producto estimado.

3. **Estimación de funcionalidad (FE).** Los datos de funcionalidad son obtenidos a través del plan de proyecto, en el cual la funcionalidad se define. Por lo tanto, para el propósito de calcular la Ecuación 1, ésta tesis recomienda utilizar las estimaciones plasmadas en el plan de proyecto, ya que este plan es diseñado en conjunto con los involucrados. En este sentido, el RESVEP considera la funcionalidad de un producto como funcionalidad esperada (la definida antes

de iniciar el desarrollo del producto) e implementada (la calculada al realizar la estimación de valor).

4. **Estimación de costo (CE).** Los datos de costo son establecidos por la compañía que desarrolla el producto, normalmente plasmados en el plan de proyecto. Esta es una medición que normalmente esta presente en las estimaciones de un proyecto. Para el propósito de aplicar la Ecuación 1, ésta tesis recomienda utilizar las estimaciones de costo del plan de proyecto, ya que son las mas cercanas a la realidad. El RESVEP utiliza el costo (estimado y real) para estimar el valor del producto.

5. **Estimación de valor (VE).** La última fase del RESVEP es la estimación del *valor estimado*, el cual es el valor con el que realmente cuenta un producto. Utilizando la Ecuación 1, la cual utiliza los indicadores de valor previamente establecidos -funcionalidad, calidad y costo- y calculados en los Pasos 2-4, la estimación de valor entonces es calculada.

Además ésta tesis provee de la información necesaria para aplicar el RESVEP en la Sección 3.3. Contiene una guía con actividades propuestas para aplicar el RESVEP, estas actividades vienen con una descripción y las roles de los involucrados en estas actividades. También incluye un conjunto de indicadores de calidad con sus medidas y ecuaciones para el proceso de diseño y el proceso de construcción descritos en el estándar ISO 12207 [11]. Además provee de información para calcular la calidad basada en el conjunto de indicadores seleccionados. También explica a detalle que datos son necesarios y cómo utilizar la ecuación para estimar el valor.

RESVEP es aplicado en dos casos de estudio, estos casos de estudio fueron realizados con datos reales. El RESVEP fue instanciado para cada uno de los casos de estudio. Durante las estimaciones, el evaluador revisó la documentación de los casos de estudio y se llevaron a cabo reuniones con los product managers de ambos equipos de desarrollo. Ambos casos de estudio fueron realizados siguiendo las guías propuestas en la Sección 3.3 y los pasos del RESVEP de la Sección 3.2.

Basado en los resultados de los casos de estudio, la utilidad de RESVEP fue identificada, se detectaron desviaciones en el proceso de desarrollo que afecta a los indicadores de valor, y principalmente información sobre calidad fue adquirida la cual afectará el valor de los productos cuando sean lanzados al mercado, todo esto surge del análisis cuantitativo de los resultados de los indicadores de valor.

INTRODUCTION

Software development is a large growing industry nowadays, the software products developing companies invest large amounts of money in creating their products and positioning them in a very competitive market, and it also occurs with every other product development area, offering a high-value becomes very important. Software customers tend to be very informed before acquiring any product because these products could either save or give problems to the company, and this situation will always affect the companies' finances. Thus, customers will prefer the product that fulfill all their requirements while offering a high value for their money, and for that customers will always make an extensive trade-off when buying a product.

However, a main problem with the released software products is that companies do not have a solid understanding of their products value situation, and, at the end these products do not fulfill the company economical expectations. Some companies might start with a very ambitious project or idea for a product but when the development process is performed the product ends as a mediocre product, failing to be competitive as the nowadays market requires. The problem is that these products are disproportionately expensive compared to the business level value that they produce. Even worse, companies draw funds and resources away from other higher value-producing opportunities [14], mainly because a deficient or lack of product value estimation. Thereafter, there is a clear missing link between the planning of a great product and the real development, since a supposedly good idea is not reflected in the final product. As Fujitsu states in [25]: *"theoretically, the amount invested in a particular business application should directly correspond with the amount of value the application contributes to the business"*.

In this sense, the software companies that provide products which at the same time satisfy the customer needs and offering a high-value are then more competitive in a value-consideration market as nowadays occurs. For reaching these goals, it is important that companies have the methods and knowledge to estimate and validate the software product value. For this it is important to recollect and analyze quantifiable data, and to identify the factors that affect the software product development. Quantifiable data would be of great support for the product manager to monitor the product value during the development process, and this information will allow managing and maybe change the development decisions in order to meet the business goals.

It is important to state that Value is a different concept than Price or Cost. Price is a financial reward for providing the product. Cost is the amount spent to develop the product. And Value is what the customers perceive as the worth of the product. The project of a product development can be successful in terms of cost (if it is finalized within its budget), but may fail to provide business value if it is not aligned on what the stakeholders want and the business area goals.

Thus, value must be taken in consideration when using software engineering techniques to develop software products, as cost, schedule, and other important aspects are often considered for development [3]. It is precisely in the software development tasks where technical issues are affecting the released products' value. Thus, companies need specific methods for estimating the software value during the development process, for getting a quantitative and real visibility about the value of the product that is going to be released to the market. However, value estimation is not an easy issue. It is easier to manage the development process of a new product with activity, effort or productivity metrics than product value metrics. There are some efforts for value estimation in software engineering, but they are mainly focused in the requirements phase such as Barney et al. in [2] in which authors address a value-based approach in *Requirements Engineering*, focusing on the process for creating product value through requirements selection for a software release. Similarly, Ojala addresses in [19] an approach for product value assessment mainly through requirements prioritization and quality attributes, and assigning percentages according to the stakeholders' opinion. The main found limitation of these approaches is that they are not specifically focused on software product value estimation. They do not offer visibility in the value management, nor quantitative figures about the value estimation during the software development process.

Another important effort for considering the value concept in the development tasks is provided by Boehm in [3]. Boehm brings the value to the software development projects, by emphasizing that each development activity has an implicit and important value. The estimation of this activities value is recommended to be used to the upper levels in companies, for getting more budget and position to the development area in the company. However, these approaches address the value estimation from the development process perspective instead of the development and released products perspective.

In this thesis a *Systematic Mapping Study* (SMS) is performed, academic and industrial papers were searched, and based on the SMS, an analysis of the state of the art in *Software Product Value Estimation* has been done. From this analysis this thesis concludes that there are important gaps in value estimation for software products during the whole development life cycle. This is the focus of this thesis approach; in which the software product value estimation focusing on the development process is addressed. In this thesis, a method to quantitatively obtain the value estimation of the development work products is provided, which directly affect the final product value. The main benefit pursued with the approach presented in this thesis is to provide the involved stakeholder in the value management, with a model

to estimate the real value of such final product before is released to the market. The aim is to provide a deep insight into value figures, to be used for identifying deviations and correct decisions which affect the expected final product value. For this thesis case studies were performed to evaluate and improve the estimation model in real industrial scenarios. This is very important to validate the model, since it is provided not only a theoretical model but also the model application into real product development scenarios.

This thesis provides interesting findings about the available literature on the software product value estimation and in general on the product value area. This thesis offers a detailed explanation step by step of the research methodology used (*Systematic Mapping Study*). The main steps of the research were: research scope and strategy, selection of primary studies, classification of studies and data extraction. Identifying and classifying the actual research on software product value estimation allows offering statistical facts and a detailed analysis about the state of the art, as well as its position in the software engineering area.

The developed estimation method is based on the value-indicators concept, these value indicators are central to the proposed approach as they give quantitative information about the software product value. Its development is based on international standards for establishing an agreed development processes and their generated development work products. Similarly, the value indicators are determined according to international standards in quality for software products. The process model to determine the value estimation is also provided. Case studies are also included to show the feasibility and usage of the approach for producing benefits in industry projects. The RESVEP was applied in two different environments, the first one was an internal school project developed by a team of students, the second was applied in a software company that its main business is to offer consulting services and software products, both branches highly related to software quality and process improvement.

The results of these case studies were also used in a simulation with the different aspects that define value to give conclusions and recommendations to the product managers. The feedback of both environments was very insightful to use and improve the estimation method.

Chapter 1 describes the thesis objectives, Chapter 2 it is the state of the art; with the description of the execution of a systematic mapping study and the related work to software product value estimation. Chapter 3 presents the Reference Model for Software Product Value Estimation (RESVEP) and a guideline to execute it. Chapter 4 contains the developed case studies applying the RESVEP, and finally Chapter 5 the main contributions of the thesis, the accomplishment of the objectives, limitations and future work.

Chapter I

THESIS OBJECTIVES

1.1 General Objective

To define a model to estimate the software product value in the development process, considering that this model must be capable of keeping the software value assigned by the business areas, through the production chain (lifecycle of software development).

1.2 Specific Objectives

To achieve the overall goal, the following specific objectives are proposed:

1. Investigate the actual research on software product value estimation.

2. To establish the elements that define software product value, those elements should be quantifiable to allow visibility of the value product through the software development process.

3. Propose a model to support the value estimation of the software product throughout the software production chain: from the definition of product value from the business areas, to the stages of software product development.

4. Establish the phases and activities of the software development on which the model would be centered.

5. Set up the requirements and activities needed to apply the software product value estimation model.

6. Elaborate a case study to prove the utility and feasibility of the model.

Chapter II

STATE OF THE ART

2.1 Systematic Mapping Study (SMS)

2.1.1 Research Methodology

The research presented in this thesis been performed following the guidelines for a Systematic Mapping Study (SMS), which is defined in [15] as follows:

"A broad review of primary studies in a specific topic area that aims to identify what evidence is available on the topic."

Following the SMS process the primary and secondary studies are used, which are defined in [15] as follows:

"Primary study. (In the context of evidence) An empirical study investigating a specific research question."

"Secondary study. A study that reviews all the primary studies relating to a specific research question with the aim of integrating/synthesizing evidence related to a specific research question."

This Section describes the research process, results and conclusions related to the Systematic Mapping Study about the Software Product Value area. The research methodology has been based on the guidelines for performing Systematic Mapping Studies stated in [16].

2.1.2 Research Scope, Search Strategy and Selection Criteria

- This systematic mapping study is motivated by the need to identify and classify primary studies on the Software Product Value topic.

- Research questions were defined (see Table 1) to set the scope of the research.

Table 1: Research questions for the SMS

ID	Question	Aim
RQ1	What is software product value?	Identify the meaning of the product value concept in software engineering.
RQ2	What type of estimation methods of software product value does exist?	Identify existing value estimation methods.
RQ3	What are the software companies experiences on product value assessment?	Get feedback on what the companies are doing in the field of value assessment of their products.
RQ4	What are the elements that create/add software product value?	Identify elements that create or add value to the software product.

Table 2: Search strings for the SMS

ID	String
SS1	Software product value
SS2	Product value
SS3	Value-based product
SS4	Value assessment
SS5	Value evaluation
SS6	Value estimation
SS7	Evaluation metrics
SS8	Value-based evaluation methods
SS9	Value evaluation experiences

- For the search strategy; search strings which are based on the research questions are used (stated in Table 2).

- As research resources the following digital libraries were considered: IEEE, ACM, Elsevier and Springer, as well as ISO standards, SEI documents and white papers from software companies.

- To select papers from the retrieved results the inclusion criteria stated in Table 3 was used.

- The exclusion criteria described in Table 4 is also considered, this is for refining the list of primary studies with high contribution.

Table 3: Inclusion criteria for the SMS

ID	Inclusion Criteria
I1	The paper has a product value definition.
I2	The paper has a software product value estimation method.
I3	Experiences of software companies in the product value area.

Table 4: Exclusion criteria for the SMS

ID	Exclusion Criteria
E1	The paper uses the word value but have no definition.
E2	The paper describes a method for estimating the product cost or it is non-related to the product value area.
E3	The paper describes a method for estimating the project and/or process instead of the product.

2.1.3 Selecting Primary Studies

Based on the search strings, a total of 52 papers were obtained (see Figure 1).

The selection process was as follows:

1. To read the abstracts and thus excluding those papers that did not seem to contribute according to the inclusion/exclusion criteria.

2. From the remaining papers the conclusion sections were reviewed, searching the extraction properties stated in Table 5, but that was not identified from the abstracts. Then a refined list of papers was produced.

3. After getting a refined list of papers with potential contribution, an analysis of all of them is performed, in search of the key elements that will contribute to the thesis goal. For this purpose, the inclusion and exclusion criteria are used.

After this selection process 36 papers of the total of 52 retrieved were of contribution to this research (see Figure 1).

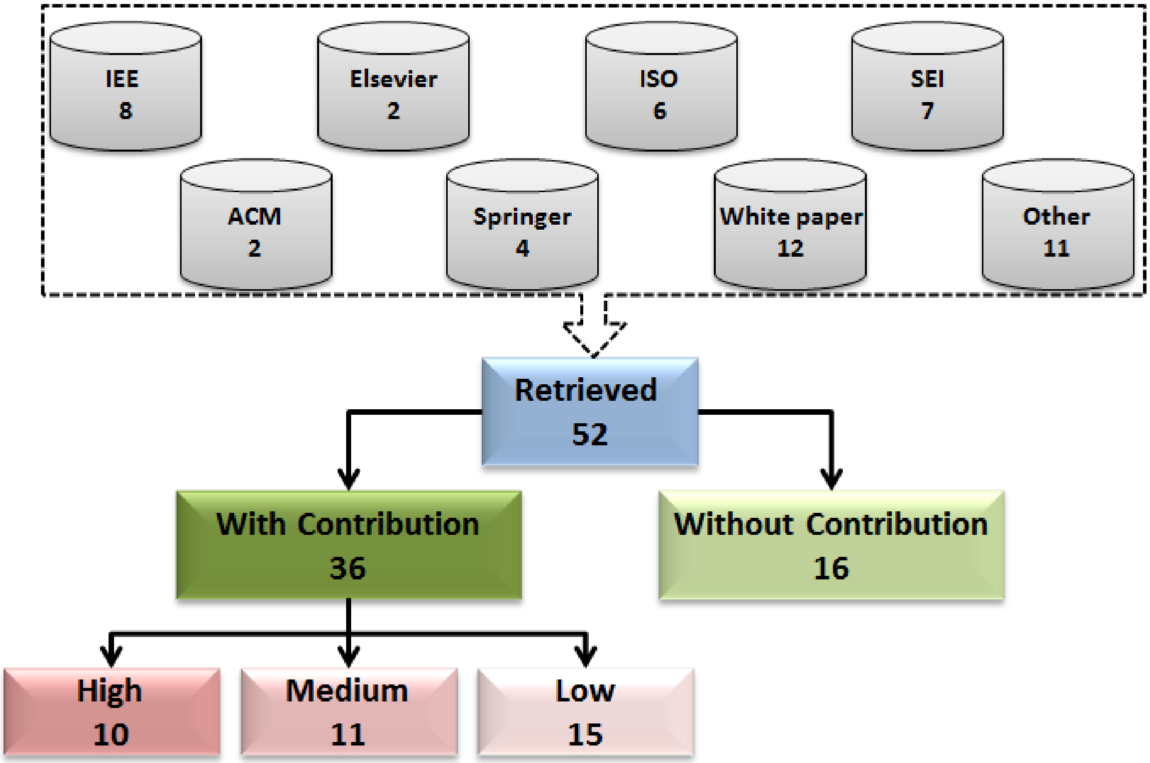


Figure 1: Papers classified by publisher and contribution

2.1.4 Classifying Selected Studies

The classification criteria is according to the level of contribution, and the type of publisher:

- The papers are classified according to their contribution to the thesis objectives (see Figure 1):

    1. High-level contribution papers have value assessment methods or important value concepts in the Software Engineering (SE) area.
    2. Medium-level contribution papers have value indicators and product value information no related to SE.
    3. Low-level contribution papers have value experiences, thoughts on the value importance that are software or non-software related.

Table 5: Extraction properties

ID	Property	Research question(s)
P1	Concept of value	RQ1
P2	Software product value	RQ1
P3	Value assessment or estimation methods	RQ2
P4	Indicators of value	RQ2, RQ3, RQ4

- A classification between academic and industry papers was performed in order to establish the importance of the value concept for software companies.
  - Academic papers: 25 (69.5%).
  - Industry papers: 11 (30.5%).

The research questions are addressed in Section 2.2; where findings of the SMS are reviewed.

2.1.5 Data Extraction

A set of properties is established in Table 5 for data extraction to address the research questions from the primary studies. The stated extracted properties help to obtain information and collect data from those primary studies.

2.1.6 SMS Conclusions

The reviewed papers of higher level contribution to this research have no more than 5 years of being published, which means there is a recent but increasing interest in software product value estimation. This area is getting more relevant due to the importance of the software product development companies, which is the main focus of this thesis. But also is getting more relevant to the software industry in general. Also software customers invest in the development of their products when there is not any available solution in the market, it is crucial to identify and justify the level of investment (costs) and the business impact (value) for the company.

It can be seen in Figure 1 that software product value is not a well-documented area. Not a lot of information specifically about the value of software products was found. Value-based Software Engineering is the former area in introducing the value concept into Software Engineering, in which is possible to find a lot of documentation about the value concept, but for the development tasks instead of the development products.

Although the academic research is bigger than the research done by software companies, some of the higher level contribution papers of the research came from white papers. It seems that the value subject is getting more important to the software development companies. This is because companies are business concerned and software product profits are expected, so they are more concerned about the value of their released products to the market and the real needs of the cost-benefit for the customers to differentiate themselves from competitors.

The related approaches for software product value estimation are mostly focused on the requirements selection and prioritization which are tasks of the software development process. The Ojala's work in [18] is the most closer to the thesis aim of research, since he already implemented the *Value study* based on the Value Standard in [23]. This standard is for products in general, but Ojala applied it to the software engineering area, and for software products. Thus, this approach gives a good perspective and feedback about the product assessments that are performed in practice. However, this study is not very exhaustive concerning to the value estimation, since this only measures quality attributes based on the stakeholders' votes.

2.2 Related Work

2.2.1 Definition of Software Product Value

2.2.1.1 Product Value Definition in Standards

Some of the officials standards reviewed in this research use the word *value*, but most of them do not offer a value definition, nor clarify the meaning or intention in the use of the concept. The only one standard that includes a value definition is the Value Methodology Standard and Body of Knowledge by SAVE International in [23]. The definition is as follows:

"Value is defined as a fair return or equivalent in goods, services, or money for something exchanged".

2.2.1.2 Product Value Definition in Business

From the Business Harvard School, Anderson and Narus define value from the business market approach in [1] as follows:

"Value in business markets is the worth in monetary terms of the technical, economic, service, and social benefits a customer company receives in exchange for the price it pays for a market offering".

Different concepts of value exist, and because of that Day and Crask in the Journal of Consumer Satisfaction, Dissatisfaction and Complaining Behavior (see [7]) enumerate some "tenets" , which are unproven principles that these authors state, and at least a few other authors coincide, and hold on to be true.

- No accepted definition of value exists.
- Value is a unique concept, but the term is often mistakenly interchanged with other concepts.
- Value is perceptual
- Value is situational and temporally determined.

- Consumers make tradeoffs when assessing value.

- Value is created by consumption or by possession.

- Multiple costs and benefits contribute to value.

Day and Crask also state that: "the greatest value derives from goods and services that are believed to yield the most benefits and require the least expenditure of consumer resources."

2.2.1.3 Value Definition in Software Engineering

Rönkkö et al in [22] defines value as follows:

"The economic concept of value is most commonly defined as the amount of money that a unit of goods or services is traded for. Utility, on the other hand, is all the good and desirable that is created by consuming a product or a service. Hence the concept of value in VBSE (Value-Based Software Engineering) is closer to economic utility than economic value".

Ojala defines value in [19] as:

"Value is a measure - usually in currency, effort or exchange or on a comparative scale - which reflects the desire to obtain or retain an item, service or ideal".

Value definition by Barney et al in [2]:

"Value constructions in economic theory are based on customer satisfaction, loyalty and re-purchasing behavior. By borrowing the economic theory, we address three aspects of value, namely product value, a customer's perceived value and relationship value".

The only definition which includes the *software product value* concept (which is also the SS1 and P2 of Section 2.1.2) comes from [2]:

"Product value is related to the product price and influenced by the quality attributes of the software product".

Barney et al, also address very important statements regarding software product value:

- The value of a product increases in direct proportion to its advantage over competitive products or decreases in proportion to its disadvantage.

- A customer's perceived value is the benefit derived from the product and is a measure of how much a customer is willing to pay for it, i.e. perceived value equal perceived benefits/perceived price, where perceived benefits and price are both measured relative to competing products.

- A customer's perceived value is influenced by his/her needs, expectations, past experiences, and culture. Relationship value is created through the social relationships between the software company and the customer. It exists through the product and customer's perceived value.

2.2.2 Related Approaches on Software Product Value Estimation

Authors offer different approaches to calculate value. Those differences can be seen in Table 6, however they use similar concepts. The Ojala's approach in [19] is very important, since this is the most related work to the thesis objectives that are regarding the value estimation on software products, and the inclusion of the quality concept into the product value. It is interesting to observe that the *quality* factor has a close relation to value; this is also considered in the Ojala's approach and the Barney et al.'s in [2].

One important and influential approach on value is provided in the *Value Standard of SAVE International* in [23]. This standard proposes a *value methodology*, which is applied to projects of several kind of products, It establishes the specific six-phase sequential Job Plan process and outlines the objectives of each of those phases. It does not standardize the specific activities that are used to accomplish each phase.

Table 6: Different approaches of value

Value Approach	**Explanation**	**Author**
$\frac{Function}{Resources}$	Function: Is measured by the performance requirements of the customer. Resources: They are measured in materials, labor, price, time, etc. required to accomplish that function	[23]
$Bo - Re$	Bo: Benefits obtained. Re: Resources expended.	[7]
$\frac{Worth}{Cost}$	Worth: The least cost to perform the required function (product, service or process), or the cost of the least cost functional equivalent. If possible can also be the worth in money, what customer sees in a product, service or process. Cost: The life cycle cost of the object, product, service or process (price paid or to be paid).	[19]
$\frac{Function + Quality}{Cost}$	Function: The specific work that a design/item (product, service or process) must perform. Quality: The owner's or user's needs, desires, and expectations. Cost: The life cycle cost of the product, service or process.	[19]
$\frac{Benefits}{Price}$	Benefits: Total benefits derived from the product. Price: The amount a customer is willing to pay.	[2]

The *value methodology* proposed by SAVE, is a systematic process that follows the *Job Plan*. Summarizing, the *value methodology* is applied by a multidisciplinary team to improve the value of a project through the analysis of functions. The *Job Plan* consists of the following sequential phases stated in [23]:

1. Information Phase. The team reviews and defines the current conditions of the project and identifies the goals of the study.

2. Function Analysis Phase. The team defines the project functions using a two-word active verb/measurable noun context. The team reviews and analyzes these functions to determine which need improvement, elimination, or creation to meet the project's goals.

3. Creative Phase. The team employs creative techniques to identify other ways to perform the project's function(s).

4. Evaluation Phase. The team follows a structured evaluation process to select those ideas that offer the potential for value improvement while delivering the project's function(s) and considering performance requirements and resource limits.

5. Development Phase. The team develops the selected ideas into alternatives (or proposals) with a sufficient level of documentation to allow decision makers to determine if the alternative should be implemented.

6. Presentation Phase. The team leader develops a report and/or presentation that documents and conveys the adequacy of the alternative(s) developed by the team and the associated value improvement opportunity.

The *Evaluation Phase* is particularly interesting to the thesis research, since this phase is concerned on the value estimation for the products generated in the development process, and not only for final products. Although the standard describes what must be done in the evaluation phase, there is no a specific method to estimate the value neither how to next improve such value.

One of the most important works is the Ojala's proposal in [19]. Which is based on the value standard of SAVE International [23]. In this approach Ojala implements SAVE into the Software Engineering field for assessing the value of the final software product (the one released to the market).

One of the key points, which is also a research concern, is the *Evaluation Phase* of the job plan. This is mainly because the subject of the thesis is concerned on the software value estimation for development products and not only for final products, as the Ojala's approach presented in [19].

Concretely, in the Ojala's approach the Evaluation Phase is addressed by implementing a value assessment study on software products, as it is explained in [18] as follows:

1. At the beginning of the evaluation phase the project team discusses the criteria for the evaluation of improvement ideas and decided criteria on quality attributes.

2. The project team members are asked to give a relative percentage (maximum 100%) for how important each criterion was for their project.

3. The project personnel calculates averages for all the criteria (Example: system stability 25%, safety 20%, optimized functioning 7.5%, ease of use 20%, maintainability 15%, and profitability 12.5%).

4. The project personnel gives points to each improvement proposal on a scale of one to six, where six indicated maximum points and one, minimum. The points allocated were multiplied by the calculated weighting percentages.

Another approach is by Cerdegren et al. in [6], in which a method called Products in Development (PiD) is proposed. PID is intended for integrating perceived customer value as a measure of performance during the development of new products. The PiD method requires a set of inputs that are assumed to be given, and they are:

- A set of n requirements
- An initial assessment of the perceived customer value for each of the n requirements
- A set of m phases or activity categories in the development value flow

The terminology for describing value in the PiD method in [6], is defined according to:

- The *Captured value* is the sum of the perceived customer value of the n requirements.
- The *Developed value* is the current value of the activities related to the n requirements for each of the m phases or activity categories in the development value flow.
- The *Developed value completed* is the minimum value of the m phases or activity categories of the developed value.

Given these assumptions and definitions the PiD method is described in [6] by the following steps:

Step 1: The Captured value is equaled to the value set in the business case. The Developed value and the Developed value completed are both set to 0. Step 1 is to be conducted as the development project is initiated.

Step N: The captured value is reassessed according to the changes in requirements from Step N-1. Requirements can be added and/or subtracted during the development. This is followed by updating the perceived customer value of the updated set of requirements. As the activities in the development project are continued until completion, the *Developed value* and the *Developed value completed* are updated accordingly.

The number (N) of steps of the PiD method depends on the complexity of the development project.

Similarly, Barney et al. [2] propose a method that is mainly focused and based on the requirements selection based on their value. This is performed through a series of activities that involve stakeholders and workshops for prioritizing and validating the system requirements. This approach does not estimate software product value during the development process, since the focus is on the value-based requirements selection. Therefore, it is not aligned with the main thesis research goals related to the value estimation on software products.

2.3 Conclusions

An SMS was performed, which resulted in 52 papers from the primary studies, from which 36 contributed to the thesis objectives. Although 36 papers were of contribution, only 10 had a high-level of contribution. This is the main reason for stating in this thesis that the Software Product Value Estimation topic is not thorough researched nor well-documented as it is nowadays published.

The aim of the SMS is not to obtain simple figures of extracted data from publications, but to tell something interesting about the status of the current research in the software product value area. It was also noticed that the industry research in this field is increasing. It seems that the software companies are more aware about the importance of value of their products. Since the companies' objective is to obtain profits, offering products with high-value increases the chance of success in the market.

A product is not an asset if its costs exceed the value that it delivers to the customers. Products must prove their value through metrics which satisfy the business objective of the company. The value of a product should be proven in the acquisition process to justify the development effort by itself, but also must be proven after such development to justify its adoption. One of the biggest issues is that measuring value is not a trivial problem. It is easier to manage the process of developing a new product with activity metrics than value metrics. Create high-value products should not be so complex, when the factors that create and destroy the value of a product are clearly identified, quantified and managed during the life cycle development.

Several of the reviewed authors agree ([2], [6], [7]) that value is highly related to perception, this statement is very interesting and it presents a challenge to the thesis research in finding a way to decipher this perception perhaps with quantitative elements that define value. This would be very helpful to get a closer perception to the real software product value.

In the reviewed papers, it has been found that quality is a key issue to understand the value of a software product; a high quality product has a higher probability of having a high value. As Barney states in [2]: "value is influenced by quality attributes of the software product". Thus quantification of these quality attributes will be very helpful in the value estimation of software products. This is an opportunity gap in software product value estimation, since there is a need by both academic and industry in getting a better understanding on this subject.

For software development companies the knowledge on the software product value estimation is fundamental, since this permits to manage the product value through the whole development life-cycle and then to assure a high value of the software products that are delivered to the market. Additionally, the value management in the development process will support making better product decisions, for instance: 1) concerning the product readiness to be released, 2) if the product needs to be modified to increase its value, or 3) if the product development should be abandoned when its value does not fulfill expectations and then it is not feasible for development.

Chapter III

A REFERENCE MODEL FOR THE SOFTWARE PRODUCT VALUE ESTIMATION

3.1 Definitions

3.1.1 Software Product Value Definition

In Section 2.2.1, several definitions have been reviewed for the value concept, all of them are valid in their own technical view or research context, however, none of them addresses entirely the research objectives of this thesis. The value concepts proposed in those works do not specifically consider factors that influence the value in intermediate software products (i.e. development work products). Therefore, in this thesis a value definition is proposed that is specific for software products. This definition considers the factors which influence the value in intermediate and/or final products. This definition is based on the related work and the state of the art conclusions of this thesis (Section 2.3), and it is stated as follows:

Value is a measure - usually in currency, effort or exchange, or on a comparative scale - of software (set of programs, procedures, algorithms and its documentation) goods or services that will meet the user's needs, desires, and expectations. All goods or services are being influenced by the quality attributes of the software product.

A main issue is how to estimate the software work product value, thus as it is reviewed in Table 6 (Section 2.2) there are several efforts for measuring this concept. The Ojala's equation [19] is based on the SAVE standard [23] and includes the quality factor in its value equation, that is defined for specifically measuring the software product value. Therefore, this thesis proposal states to use the same estimation Equation 2 (specified in Section 2.2.2), but slightly modifying its core components in order to meet this thesis objectives.

$$Value = \frac{Function + Quality}{Cost} \tag{2}$$

Where:

- **Function** is the specific work that a design/item (product, service or process) must perform (e.g.: function points, which are needed to accomplish a software component).

- **Quality** is the owner's or user's needs, desires, and expectations (e.g.: software quality attributes such as: usability, portability, security, traceability, etc.).

- **Cost** is the amount of effort for developing the product, service or process (example: the effort expressed in man-hours).

3.1.2 Software Product Value Indicators

According to the value definitions provided in Section 2.2.1, value is closely related to perception. Thus, the identification and quantification of the elements which contribute or destroy value will provide a systematic and formal method for getting a more real perception of the value measurement.

When a company starts a project of a new software product, there are certain business criteria to take into account that affects the product value. The business criteria is stated by a variety of specialists (business experts, engineers, marketers, project manager, product manager, etc...), because it covers different fields, not only software related experts. That group of specialists establishes the criteria that the product must meet to obtain success in the market. These business criteria must be translated into technical specifications for the product, which is where the software development process starts. The issue is to respect those technical specifications that the product must meet in order to be successful on the market. In Figure 2, on the left side there is an example of the referred business criteria and on the right side the technical specifications which are based on the business criteria. They are all related to value.

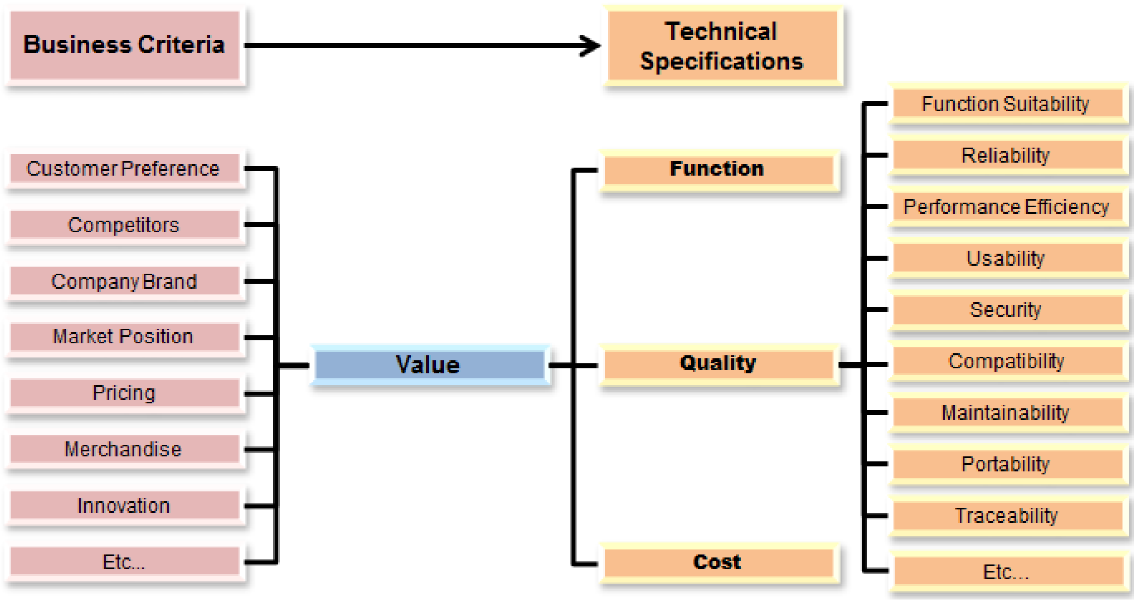


Figure 2: Relationship between business criteria and technical specifications that affect value

Particularly for software products, the value elements must be managed during the development process.

If the value management is not taken into account during the product life cycle, the product value could not meet the business objectives. For instance, if a company already has a product plan specifications (which must meet to obtain the desired value), but the company does not perform any validation of value through the development process, then it cannot know the status of the product value at any process (phase or discipline, depending on the adopted software development process), this leads not to quantitative know the value of the final product either (Figure 3 describes this situation).

It is very important to estimate and validate the software product value in the development process, the reason is that technical issues are strongly affecting the final product value, since the several development work products are used to build the final software product that is released to the market. Thereafter, such work products must be subject to value estimation according to specific value indicators.

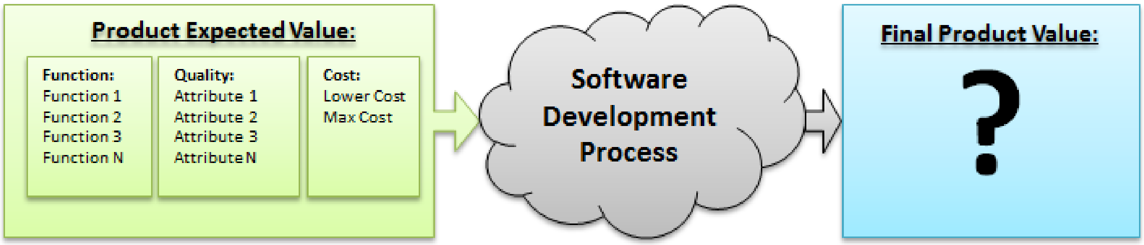


Figure 3: Transition of the value of a product from expected value to final value delivered

In Section 2.3 it is stated that the quality factor is closely affecting the value estimation, and this is happening also in software products. Thus the value indicators can be defined in terms of quality factors which are directly determining the value estimation. The aim is to get quantitative value metrics on the work products in development phases in which the business criteria are barely involved. In this phases, the business perspective (such as customer preferences, competitors' features, and market position) can be lost if several important quality factors are missing.

Then, in this thesis a *Software Product Value Indicator* definition is proposed (based on [20]) as follows:

It is a means for providing specific and quantitative information about the state or condition of the software product value.

Therefore, indicators are developed based on quantitative measurements or statistics of the development work products value. In this sense, indicators allow getting visibility about the real offered value to the customer.

Figure 4 illustrates the value indicators approach to define value. The software product has an intrinsic value, which is defined according to the software product value definition that is stated in Section 3.1.1. Value is defined by function, quality and cost; these elements that define value are actually value indicators. For some indicators there are sub-indicators, and all of these sub-indicators impact on the product value, therefore all of these are taken into account to estimate de product value.

Specifically, this thesis specially focuses on the estimation of the *Value Quality Indicators* that affect value. Although function and cost indicators impact on value, to research and propose about these other indicators is out of the scope of this thesis objectives. However, for performing the case studies in which the thesis proposal validation is carried out, it will be used third-party methods to estimate function and cost in specific software projects.

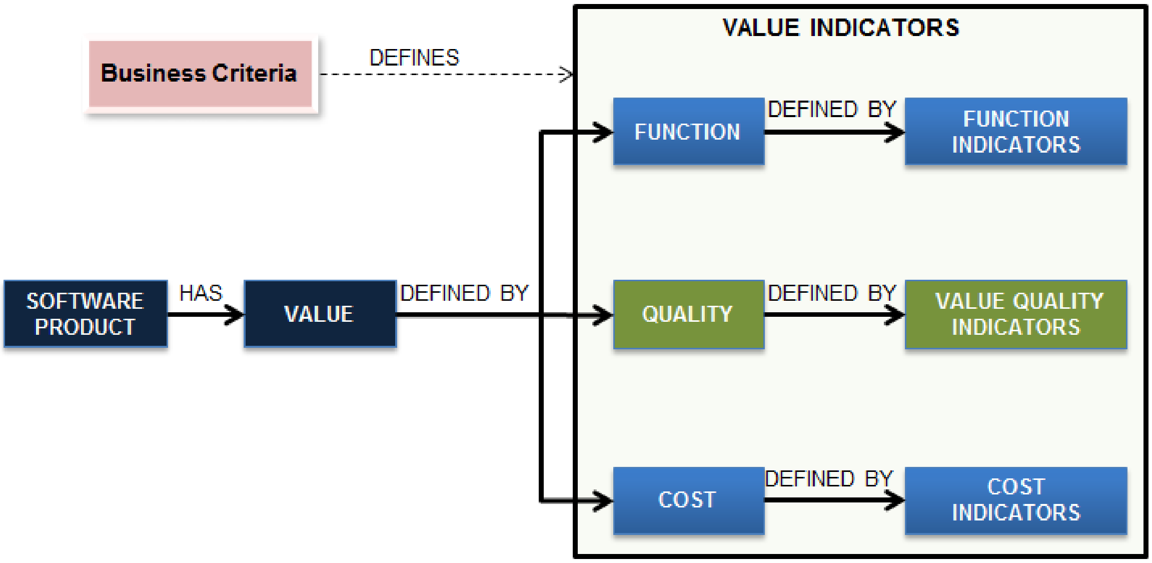


Figure 4: Value and its definition by the value indicators

3.2 Definition of the Reference Model for the Software Product Value Estimation Process

The reference model for the software product value estimation process (RESVEP) has been developed considering the analysis of the state of the art that is stated in

Chapter 2, regarding value estimation in the software engineering area. RESVEP is an abstract framework that comprises a set of defined process chunks (items such as activities, work products, sub-processes, documents, or tasks) that are linked through an established workflow. Figure 5 illustrates the reference model workflow.

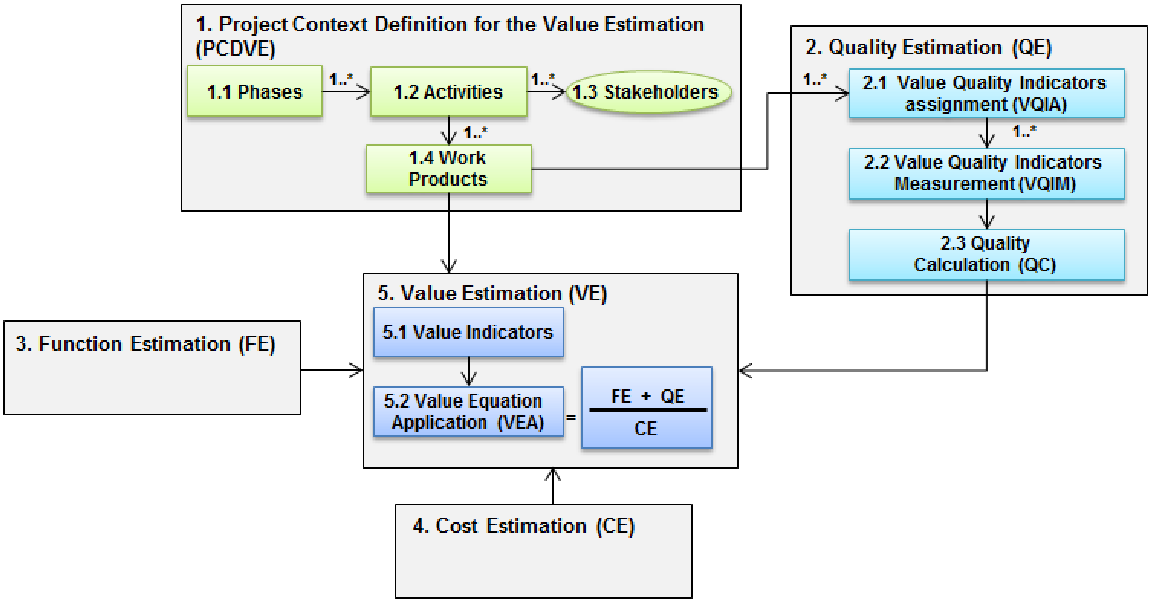

1. Project Context Definition for the Value Estimation (PCDVE)
1.1 Phases
1..*
1.2 Activities
1..*
1.3 Stakeholders
1..*
1.4 Work Products
2. Quality Estimation (QE)
1..*
2.1 Value Quality Indicators assignment (VQIA)
1..*
2.2 Value Quality Indicators Measurement (VQIM)
2.3 Quality Calculation (QC)
3. Function Estimation (FE)
5. Value Estimation (VE)
5.1 Value Indicators
5.2 Value Equation Application (VEA)
=
FE + QE
CE
4. Cost Estimation (CE)

Figure 5: Reference model to estimate the software value, based on value indicators for development work products

The reference model describes the process for performing a value estimation of the development work products in a software project. The software product value estimation process is divided in the following phases:

1. Project Context Definition for the Value Estimation (PCDVE).

2. Quality Estimation (QE).

3. Function Estimation (FE).

4. Cost Estimation (CE).

5. Value Estimation (VE).

The details of these phases are described as follows:

1. **Project Context Definition for the Value Estimation (PCDVE).** It is very important to know the circumstances or facts that surround the project of value estimation to apply the measurements and equations for that specific context. It also depends on the software development process, which is defined according to the following steps:

   1.1 **Phases.** The establishment of the phase, which is the stage of the development process where the estimation is going to be performed. The phases, also named disciplines or processes, depend on the software development framework (for example ISO 12207 [11], UP [13] and IEEE 1233-1998 [8]).

   1.2 **Activities.** Each phase has one or many activities, it is important to allocate the software development activity which belongs to the previous established phase. In this step, the activities are established based depending of the software development framework of the project context.

   1.3 **Stakeholders.** In the context of the RESVEP, the stakeholders are the involved persons of the development team to create and manage the work products. Each activity has their respective stakeholders, whose corresponding roles are established in the specifications of the software development framework used.

   1.4 **Work Products.** The work products are a deliverable or outcome that must be produced to complete a project and achieve its objectives. Activities can result in one or many work products. In this step the work products are established based on the software development framework of the project context. (example: ISO 12207[11], UP[13] and the *4+1 architecture view* [17]).

2. **Quality Estimation (QE).** This is the most difficult part to measure, since the quality attributes vary depending on the involved stakeholders' perception about the product quality. Quality attributes generally are very hard to measure, for example: usability, portability, reusability or performance. This thesis provides a proposal for estimating the software quality, mainly based on the *Software engineering-Software product Quality Requirements and Evaluation (SQuaRE) Quality model* [12]. The quality estimation is composed by the following steps:

    2.1 **Value Quality Indicators Assignment (VQIA).** In this step it is assigned a set of value quality indicators according to each work product and its corresponding type. In this sense, the value quality indicators depend on the quality attributes of each work product type. The reference model focuses on defining the quantifiable indicators to get measures of quality (and therefore value). These indicators are then used to identify if the product is on the right way to accomplish the expected value according to the product designed plan by the business side. Quality attributes affect quality, and quality has a crucial impact on value, thus in the context of this thesis, quality indicators are considered value indicators that in the context of this thesis they are called *Value Quality Indicators* (according to the definition for software product value indicator in Section 3.1.2).

    2.2 **Value Quality Indicators Measurement (VQIM).** In this step occurs the assignment of a measurement and the corresponding equation to get quantitative data for each value quality indicator. Measurement and equations are mainly based on the ISO 9126 [10].

    2.3 **Quality Calculation (QC).** The quality calculation is based on the set of the value quality indicators that are specified for a product (Step

2.1). After the measurements and equations are assigned (Step 2.2) for the product's value quality indicators (according to the product's quality attributes), then the corresponding equation is applied to get the quantitative measurement for the whole product quality.

3. **Function Estimation (FE).** The function figure is given by the development company through the project plan, in which is typically defined the software product functionality. Therefore, for the purpose of calculating the Equation 2, this thesis recommends to use the project plan's functionality estimation, since this is designed together with all the involved stakeholders. In this sense, this thesis' reference model will be considered a functionality size for a given work product (expected and implemented).

4. **Cost Estimation (CE).** The cost figure is established by the development company, through the project plan estimation. This is a measure that commonly every software project estimates, therefore for the purpose of calculating the Equation 2, this thesis recommends to use the project estimation which is the closest to the reality. The reference model uses the cost managed in the project management (both planned cost and real cost) to estimate the required work products' value.

5. **Value Estimation (VE).** The next and last phase in the RESVEP is the estimation of the estimated value, which is the real delivery value in by a development work product. Using the Equation 2 by Ojala [19], which uses the value indicators previously established - function, quality and cost- and calculated in Steps 2-4, the value estimation is then calculated.

3.3 Guidelines for Performing the Proposed Value Estimation Process

In this section, a set of recommendations is proposed in order to implement the RESVEP model (introduced in Section 3.2) for estimating the value of the work product in a given software project. Some of these recommendations have been identified during the case studies performance for validating the approach presented in this thesis.

The recommendations are presented in this section as a set of requirements and general activities, which are described in next sections.

3.3.1 Recommended Activities

This thesis provides as proposal a sequence of activities to perform the value estimation process of RESVEP described in Section 3.2. The activities are described in Table 7, with their respective description, stakeholders and the RESVEP steps that correspond to each activity.

3.3.2 Supporting the Value Quality Indicators Measurement and Quality Calculation

This section comprises the definition of a set of *Value Quality Indicators* specifically for the design and construction process of [11], to support the *Value Quality Indicators Measurement*, providing the value quality indicators with their measurement, equation and interpretation of result.

The purpose of the project context definition step is to identify the main activities, tasks and work products on which the project is located, in order to assign the corresponding value indicators to the involved work product in the value estimation process. These indicators are going to be defined as a general set of all the possible indicators that could be included in the estimation process. The project context and needs are going to determine both: 1) the subset of activities, stakeholders and

Table 7: Proposed activities for performing the value estimation process RESVEP (proposed in Section 3.2), the step description and involved stakeholders are included

Activities	**Description**	**Involved Stakeholders**	**RESVEP Steps**
1. Introduction to the software product value estimation model (RESVEP), based on value indicators.	The team of the company where the value estimation is going to be performed gets a brief explanation of the reference model and the activities to realize.	Evaluator, product manager, development team.	1. PCDVE
2. Gather knowledge about the project in which the RESVEP model will be applied.	The evaluator obtains knowledge about the project to be aware of the type of project it is, to set the context of the software product value estimation and its environment.	Product manager, evaluator.	
3. Delimitation of the work products to be evaluated under RESVEP.	On this activity the product manager decides which work product(s) are going to be estimated.	Product manager.	1.4 Work products
4. Delimitation of the value quality indicators to be used in the estimation with RESVEP.	Based on the work product(s) selection the product manager (advised by the evaluator) selects the set of quality attributes to be measured, a percentage from 0% to 100% is given to each quality attribute as the expected measure.	Product manager, evaluator.	2.1 VQIA
5. Application of the value indicator measurements and equations to obtain Quality for each work product.	At this stage the evaluator applies the measurements and equations to obtain quality, the evaluator could request assistance of the product manager to clarify measurements or doubts about the characteristics of the product. The results of the value quality indicators equations are verified with the expected measure given by the product manager on activity 4 to calculate the quality ratio.	Evaluator, product manager.	2.2 VQIM 2.3 QC
6. Verification of the function factor in Eq. 2.	A verification of the expected function is performed, every functionality expected is checked to get the real developed functions, to finally calculate the function ratio.	Product manager, evaluator.	3. FE
7. Verification of the development cost factor in Eq. 2.	A comparison of the expected cost to develop the product against the real cost to develop the work product is done, to calculate the cost ratio.	Product manager, evaluator.	4. CE
8. To estimate the value for each of the work products in development, using Eq. 2.	Using the value equation, an estimation of the value is performed using the ratio of function, quality and cost.	Evaluator.	5. VE
9. Interpretation of results and value elements that comprise it, using simulation techniques.	Based on the results of the value equation, an interpretation is conducted, in order to give qualitative findings of the value and its value indicators.	Evaluator.	

work products, and 2) the subset of value indicators related to such subset of work products.

This thesis defines a reference model that is applicable to be used through any discipline of the software development process. Therefore, in order to support the RESVEP step *2.2 Value Quality Indicators Measurement*, this thesis provides a set of *Value Quality Indicators* (including the corresponding measurements, equation and interpretation of the result) for the design process and construction process stated in the ISO standard 12207 [11].

The intention of providing this set of value quality indicators is to help the practitioners of this thesis model in the measurement of the value quality indicators, which is carried out in RESVEP step 2.2.

The set of indicators provided are mainly obtained from the quality model Software product *Quality Requirements and Evaluation (SQuaRE)* in [12] and the measurement and equations are obtained or based on the ISO 9126 [10].

The main reason to perform the estimation in these two processes, is due to the results of the estimation at these stages, there is still a chance to change and tune up the value attributes that are directly affecting the resulting estimating value before the product is released to the production process; or even better before the product is released into the market.

3.3.2.1 Design Process: Value Quality Indicators

Table 8 shows a set of eligible value quality indicators for the work products for the *Design Process* [11] of the software development process. The indicators are described with their respective measurement and equation.

Table 8: Value Quality Indicators definition, measurements and equation for the work products of the *Design Discipline*

Value Quality Indicator	**Measurement and Equation**
x1: Traceability to the requirements	$x1 = \frac{A}{B}$ x1 = Traceability to the requirements. A = Number of traceable design items confirmed in review. B = Number of items checked. Interpretation of the result: $0 <= x1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
S: Security	Equation of security: $S = \frac{s1+s2}{2}$
s1: Confidentiality	$s1 = 1 - \frac{A}{B}$ s1 = Confidentiality. A = Number of components that are not secure from having unauthorized disclosure of data or information, whether accidental or deliberate. B = Total number of components that handles data evaluated. Interpretation of the result: $0 <= s1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
s2: Integrity	$s2 = \frac{A}{B}$ s2 = Integrity. A = Number of components that prevents unauthorized access to, or modification. B = Total number of components that handles data evaluated. Interpretation of the result: $0 <= s2 <= 1$ The closer to 1, the better. Measurement of the standard [10].
M: Maintainability	Equation of Maintainability: $M = \frac{m1+m2+m3+m4+m5}{5}$
m1: Modularity.	$m1 = 1 - \frac{A}{B}$ m1 = Modularity. A = Number of components that have high impact when they are modified. B = Total number of components. Interpretation of the result: $0 <= m1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
m2: Reusability.	$m2 = \frac{A}{B}$ m2 = Reusability. A = Number of components that are reusable. B = Total number of components. Interpretation of the result: $0 <= m2 <= 1$ The closer to 1, the better. Measurement of the standard [10].

Continued on Next Page. . .

Table 8 – Continued

Value Quality Indicator	**Measurement and Equation**
m3: Analyzability.	$m3 = \frac{A}{B}$ m3 = Analyzability. A = Number of components that can be easy diagnosed for deficiencies or causes of failures in the component model. (Helped by comments, version, code standards, etc.) B = Total number of components. Interpretation of the result: $0 <= m3 <= 1$ The closer to 1, the better. Measurement of the standard: [10].
m4: Modifiability.	$m4 = \frac{A}{B}$ m4 = Modifiability. A = Number of components that can be modified without introducing defects or degrading performance. B = Total number of components. Interpretation of the result: $0 <= m4 <= 1$ The closer to 1, the better. Measurement of the standard [10].
m5: Testability.	$m5 = \frac{A}{B}$ m5 = Testability. A = Number of cases in which a component can be tested appropriately. B = Number of cases of component tests. Interpretation of the result: $0 <= m5 <= 1$ The closer to 1, the better. Measurement of the standard [10].
P: Portability	Equation of portability: $P = \frac{p1}{1}$
p1: Replaceability	$p1 = \frac{A}{B}$ p1 = Replaceability. A = Number of components that can replace another for the same purpose in the same environment. B = Total number of components in the component model. Interpretation of the result: $0 <= p1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
FS: Functional suitability	Equation of functional suitability: $FS = \frac{fs1+fs2}{2}$
fs1: Functional appropriateness	$fs1 = 1 - \frac{A}{B}$ fs1 = Functional appropriateness. A = Number of missing components or with errors detected in design evaluation. B = Number of components described in the requirements. Interpretation of the result: $0 <= fs1 <= 1$ The closer to 1, the better. Measurement of the standard [10].

Continued on Next Page...

Table 8 – Continued

Value Quality Indicator	**Measurement and Equation**
fs2: Accuracy	$fs2 = 1 - \frac{A}{B}$ fs2 = Accuracy. A = Number of inconsistent components detected in design evaluation. B = Number of components described in the requirements. Interpretation of the result: $0 <= fs2 <= 1$ The closer to 1, the better. Measurement of the standard [10].
U: Usability	Equation of usability: $U = \frac{u1+u2}{2}$
u1: Learnability	$u1 = \frac{A}{B}$ u1 = Learnability. A = Number of components whose purpose is correctly described in the design. B = Number of components evaluated. Interpretation of the result: $0 <= u1 <= N$ If equal to 1 or higher, the better. Mapped equation from the standard [10].
u2: Ease of use	$u2 = \frac{t1}{t2}$ u2 = Ease of use. t1 = Mean real-time taken to learn a component application correctly. t2 = Mean expected-time taken to learn a component application correctly. Interpretation of the result: $0 <= u2 <= N$ If equal to 1 or higher, the better. Mapped equation from the standard [10]. If more than one unit of software is evaluated, there will be a summation of t1 of each unit, and this also applies to t2.

3.3.2.2 Construction Process: Value Quality Indicators

In Table 9 there is a set of eligible value quality indicators for the work products for the *Construction Process* [11] of the software development process. The indicators are described with their respective measurements, equation and interpretation of results.

Table 9: Value Quality Indicators definition, measure and equation for the work products of the *Construction Process*

Value Quality Indicator	**Measurement and Equation**
U: Usability	Equation of usability: $U = \frac{u1+u2+u3+u4}{4}$

Continued on Next Page. . .

Table 9 – Continued

Value Quality Indicator	**Measurement and Equation**
u1: Learnability	$u1 = \frac{t1}{t2}$ u1 = Learnability. t1 = Mean expected-time taken to learn a software unit correctly. t2 = Mean real-time taken to learn a software unit correctly. Interpretation of the result: $0 <= u1 <= N$ If equal to 1 or higher, the better. Measurement of the standard [10]. If more than one unit of software is evaluated, there will be a summation of t1 of each unit, and this also applies to t2.
u2: Ease of use	$u2 = \frac{t1}{t2}$ u2 = Ease of use. t1 = Mean expected-time taken to use a software unit correctly. t2 = Mean real-time taken to use a software unit correctly. Interpretation of the result: $0 <= u2 <= N$ If equal to 1 or higher, the better. Mapped equation from the standard [10]. If more than one unit of software is evaluated, there will be a summation of t1 of each unit, and this also applies to t2.
u3: User error protection	$u3 = \frac{A}{B}$ u3 = User error protection. A = Number of user-interface units that protects users against making errors. B = Number of user-interface units evaluated. Interpretation of the result: $0 <= u3 <= 1$ The closer to 1, the better. Measurement of the standard [10].
u4: User interface aesthetics	$u4 = \frac{\frac{A_1+A_2+A_3+...+A_N}{B}}{10}$ u4 = User interface aesthetics. A_i = Rating assigned to the user interface (0 to 10) according to user satisfaction criteria. B = Number of user-interface units evaluated. Interpretation of the result: $0 <= u4 <= 1$ The closer to 1, the better. Mapped equation from the standard [10].
PE: Performance efficiency	Equation of performance efficiency: $PE = \frac{pe1+pe2}{2}$

Continued on Next Page...

Table 9 – Continued

Value Quality Indicator	**Measurement and Equation**
pe1: Time behavior	$pe1 = \frac{t1}{t2}$ pe1 = Time behavior. t1 = Time of response expected. t2 = Time of real response. Interpretation of the result: $0 <= pe1 <= N$ If equal to 1 or higher, the better. Measurement of the standard [10]. If more than one unit of software is evaluated, there will be a summation of t1 of each unit, and this also applies to t2.
pe2: Resource utilization	$pe2 = \frac{A}{B}$ pe2 = Resource utilization. t1 = Expected memory requirement for the software unit. t2 = Real memory requirement for the software unit. Interpretation of the result: $0 <= pe2 <= 1$ If equal to 1 or higher, the better. Measurement of the standard [10].
C: Compatibility	Equation of compatibility: $C = \frac{c1+c2}{2}$
c1: Co-existence	$c1 = 1 - \frac{A}{B}$ c1 = Co-existence. A = Number of functions that cannot co-exist with other independent software in a common environment sharing common resources without any detrimental impacts. B = Number of functions tested. Interpretation of the result: $0 <= c1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
c2: Interoperability	$c2 = \frac{A}{B}$ c2 = Interoperability. A = Number of functions that can exchange information and use the information that has been exchanged. B = Number of functions evaluated. Interpretation of the result: $0 <= c2 <= 1$ The closer to 1, the better. Measurement of the standard [10].
FS: Functional suitability	Equation of functional suitability: $FS = \frac{fs1+fs2}{2}$
fs1: Functional appropriateness	$fs1 = 1 - \frac{A}{B}$ fs1 = Functional appropriateness. A = Number of missing functions or with errors detected in evaluation. B = Number of functions described in design. Interpretation of the result: $0 <= fs1 <= 1$ The closer to 1, the better. Measurement of the standard [10].

Continued on Next Page...

Table 9 – Continued

Value Quality Indicator	**Measurement and Equation**
fs2: Accuracy	$fs2 = 1 - \frac{A}{B}$ fs2 = Accuracy. A = Number of inconsistent functions detected in evaluation. B = Number of functions described in design. Interpretation of the result: $0 <= fs2 <= 1$ The closer to 1, the better. Measurement of the standard [10].
S: Security	Equation of security: $S = \frac{s1+s2+s3+s4+s5}{5}$
s1: Confidentiality	$s1 = 1 - \frac{A}{B}$ s1 = Confidentiality. A = Number of functions that are not secure from having unauthorized disclosure of data or information, whether accidental or deliberate. B = Total number of functions that handles data evaluated. Interpretation of the result: $0 <= s1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
s2: Integrity	$s2 = \frac{A}{B}$ s2 = Integrity. A = Number of functions that prevents unauthorized access to, or modification. B = Total number of functions that handles data evaluated. Interpretation of the result: $0 <= s2 <= 1$ The closer to 1, the better. Measurement of the standard [10].
s3: Non-repudiation	$s3 = \frac{A}{B}$ s3 = Non-repudiation. A = Number of functions log file of their events so they can be proven to have taken place. B = Total of functions that need a log file. Interpretation of the result: $0 <= s3 <= 1$ The closer to 1, the better. Measurement of the standard [10].
s4: Accountability	$s4 = \frac{A}{B}$ s4 = Accountability. A = Number of functions that their actions can be traced uniquely to them in the log file. B = Total number of functions that need a log file to trace actions. Interpretation of the result: $0 <= s4 <= 1$ The closer to 1, the better. Measurement of the standard [10].

Continued on Next Page...

Table 9 – Continued

Value Quality Indicator	**Measurement and Equation**
s5: Authenticity	$s5 = \frac{A}{B}$ s5 = Authenticity. A = Resource or subject identification (Real percentage of veracity for assuring this identification). B = Resource or subject identification (Expected percentage of veracity for assuring this identification). Interpretation of the result: $0 <= s5 <= 1$ The closer to 1, the better. Measurement of the standard [10].
R: Reliability	Equation of reliability: $R = \frac{r1+r2+r3+r4}{4}$
r1: Maturity	$r1 = \frac{A}{B}$ r1 = Maturity. A = Desired system failures (reliability needs) in T. B = Current system failures in T. T = Time period (Time of the maturity test). Interpretation of the result: $0 <= r1 <= 1$ The closer to 1, the better. Measurement of the standard [10]. Special cases when a variable is equal to 0: 1. If $A = 0$ and $B = 0$, then $r1 = 1$. 2. If $A = 0$ and $B > 0$, then $r1 = 0$. 3. If $A > 0$ and $B = 0$, then $r1 = 1$.
r2: Availability	$r2 = \frac{t1}{t2}$ r2 = Availability. t1 = Time of the component being in an “up state” (operational and accessible when required). t2 = Time period (Time of the availability test). Interpretation of the result: $0 <= r1 <= 1$ The closer to 1, the better. Measurement of the standard: [10].
r3: Fault tolerance	$r3 = \frac{A}{B}$ r3 = Fault tolerance. A = Number of functions that operate as intended despite the presence of hardware or software faults. B = Number of functions evaluated. Interpretation of the result: $0 <= r1 <= 1$ The closer to 1, the better. Measurement of the standard: [10].

Continued on Next Page...

Table 9 – Continued

Value Quality Indicator	**Measurement and Equation**
r4: Recoverability	$r4 = \frac{A}{B}$ r4 = Recoverability. A = Number of functions that can recover data directly affected and re-establish the desired state of the system in the case of an interruption or a failure. B = Total number of functions that handles data to be evaluated. Interpretation of the result: $0 <= r1 <= 1$ The closer to 1, the better. Measurement from the standard: [10].
X1: Traceability to the requirements	$x1 = \frac{A}{B}$ x1 = Traceability to the requirements. A = Number of traceable software units confirmed in review. B = Number of items checked. Interpretation of the result: $0 <= x1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
P: Portability	Equation of portability: $P = \frac{p1+p2+p3}{3}$
p1: Adaptability	$p1 = \frac{Hi+Si}{2}$ p1 = Adaptability. 1. Hi = Hardware independence $Hi = 1 - \frac{A}{B}$ A = Number of functions of which tasks were not completed or not enough resulted to meet adequate levels during combined operating testing with environmental hardware. B = Total number of functions evaluated. Interpretation of the result: $0 <= Hi <= 1$ The closer to 1, the better. Measurement of the standard [10]. 2. Si = Software independence $Si = 1 - \frac{A}{B}$ A = Number of functions of which tasks were not completed or were not enough resulted to meet adequate level during combined operating testing with operating system software or concurrent application software. B = Total number of functions evaluated. Interpretation of the result: $0 <= Si <= 1$ The closer to 1, the better. Measurement of the standard [10].

Continued on Next Page. . .

Table 9 – Continued

Value Quality Indicator	**Measurement and Equation**
p2: Replaceability	$p2 = \frac{A}{B}$ p2 = Replaceability. A = Number of functions that can replace another for the same purpose in the same environment. B = Total number of functions evaluated. Interpretation of the result: $0 <= p2 <= 1$ The closer to 1, the better. Measurement of the standard [10].
p3: Installability	$p3 = \frac{A}{B}$ p3 = Installability. A = Number of environments where each function can be successfully installed and/or uninstalled. B = Number of environments specified in the quality scenarios. Interpretation of the result: $0 <= p3 <= 1$ The closer to 1, the better. Measurement of the standard [10].

3.3.2.3 Quality Calculation

After performing the step 2.2 of the RESVEP: *Value Quality Indicators Measurement* (VQIM), the following step is the *Quality Calculation* (QC), in which the assigned and measured value quality indicators are averaged in order to obtain quality. At the step 2.3 the *Expected Measure* (EX) is already defined by step 2.1 VQIA, and the *Estimated Measure* (ES) is obtained by the step 2.2 VQIM.

In order to obtain Quality, the following steps are performed:

1. The first part is to calculate the Ratio of each Value Quality Indicator (RVQI) that is assigned to the work product. The ratio is obtained by dividing the *Estimated Measure* by the *Expected Measure*, expressed in Equation 3.

2. The second part is to sum each ratio of the value quality indicators, the ratio results are from 0 to 1.0, and if the ratio is higher than 1.0, it is normalized to 1.0, although the exceeded result is taken into consideration for analysis and value conclusions.

3. The final part is to divide the summed result of the ratio of the value quality indicators with the number of indicators assigned to the work product.

$$RVQI = \frac{QI_{ES}}{QI_{EX}} \tag{3}$$

Where:

$RVQI$ = Ratio of the Value Quality Indicator.

QI_{ES} = Estimated Measure of the Value Quality Indicator.

QI_{EX} = Expected Measure of the Value Quality Indicator.

Equation 4 summarizes all the steps. This is done in order to obtain the *Ratio's* of the *Estimated Quality* and the *Expected Quality.* It is important to state the ratio of each value quality indicator, this way the results of the estimation of different *Value Quality Indicators* are not mixed.

$$Q = \frac{RVQI_1 + RVQI_2 + RVQI_3 + ... + RVQI_N}{N} \tag{4}$$

Where:

Q = Quality.

$RVQI$ = Ratio of the Value Quality Indicator.

N = Number of Value Quality Indicators assigned to the work product.

For example if the following value quality indicators define the quality of a product: Portability (P), Usability (U), Functional Suitability (FS), Maintainability (M) and Security (S). The Equation 5 should be applied in order to obtain Quality. Each *Estimated Measure* of the indicators is divided by the *Expected Measure*, the result of the divisions should be summed, and afterwards divided by the number of indicators.

$$Q = \frac{\frac{P_{ES}}{P_{EX}} + \frac{U_{ES}}{U_{EX}} + \frac{FS_{ES}}{FS_{EX}} + \frac{M_{ES}}{M_{EX}} + \frac{S_{ES}}{S_{EX}}}{5} \tag{5}$$

3.3.3 Supporting the Value Estimation

3.3.3.1 Required Data to the Value Estimation

The requirements to set up the value estimation process following the RESVEP are described in Table 10. These requirements should be obtained through performing the RESVEP steps 1 to 4.

Table 10: Required data to the *Value Equation Application (VEA)* of the selected *Work Product*

Value Element	**Requirement**
Data for Estimating the Function	**Expected Functionality.** Normally stated in the project plan, it is the expected quantitative measurement of the functional size of the work product.
	Developed Functionality. It is the measurement of the developed functional size of the work product at the time of the estimation. Normally function is expressed in function points or use case points.
Data for Estimating the Quality	**Expected Quality.** It is a set of quality attributes that are prioritized with a number from 0% to 100% of coverage. These could be obtained by the description of the product, its architecture or based on the product manager knowledge.
	Estimated Quality. It is obtained through the execution of the measurements and the equations of the value quality indicators selected according to the work product to estimate, and the equation application to obtain quality.
Data for Estimating the Cost	**Expected Cost.** It is normally stated in the project plan. The expected cost to develop the selected work product.
	Real Cost. A verification of the real cost to develop the selected work product.

3.3.3.2 Applying the Value Equation

With the data stated in Table 10 it is possible to use the Equation 2, proposed to obtain Value, and perform the step 5.2 (*Value Equation Application*) of the RESVEP. But it is necessary to obtain the ratio of every element of the value equation, this is to divide the *Estimated or Real Measurements* with the *Expected Measurements* of *Functionality*, *Quality* and *Cost*. Equations 6, 4 (in Section 3.3.2.3) and 7 describes how to obtain the ratio of each element of the value formula.

$$FunctionRatio = \frac{DevelopedFunctionality}{ExpectedFunctionality} \tag{6}$$

$$CostRatio = \frac{RealCost}{ExpectedCost} \tag{7}$$

With the ratio results of the value elements, the achieved percentage of each element can be obtained (multiplying it by 100). The value Equation 2 is applied with the ratio of each value element obtained by Equations 6, 4 and 7. This value result is a reference figure, which is going to give visibility on the value state of the estimated product. Equation 8 shows an example of the application of the value equation with every ratio element equal to 1.

$$Value = \frac{Function + Quality}{Cost} = \frac{1+1}{1} = \frac{2}{1} = 2 \tag{8}$$

The value numeric result does not give much information by itself, but it is a reference number to give a perspective on the differences between the expected value and the estimated value. For example the data of Equation 8 is the case on where the expected and estimated measurements of the data it is equal, therefore the expected value has been achieved. As it is mentioned before it is only a reference figure, in a different case where the expected value has not been achieved, this figure will change, and the interesting thing about RESVEP is that it offers information on the different elements (Value Indicators) that define value, therefore an extensive analysis of the value indicators can be performed in order to detect and correct problems in the development stage on where the products are being estimated.

Chapter IV

CASE STUDIES

In this chapter two case studies are presented. These case studies were performed using an instantiation of the reference model for the software product value estimation process. The two estimations were implemented with real-life data. During the estimations, the evaluator read several documents of the cases and ensured the findings with several interviews with different stakeholders involved in the project of the estimated products. Both case studies were performed following the guidelines proposed in Section 3.3 and the steps of the RESVEP in Section 3.2.

4.1 Case Study 1: ISPC System

4.1.1 ISPC: Introduction to RESVEP

A general explanation of the RESVEP was given to the product manager and the involved team members, the RESVEP steps, the activities, the information data and meetings required to develop the case study. Also the case study objectives were stated.

4.1.1.1 ISPC Case Study Objectives

General Objective. The general objective of this case study of the ISPC, was to apply the RESVEP and verify its feasibility. Also it was considered to obtain feedback from a case study with real-life data, to adjust or improve the indicators and measurements and the way it is measured. ISPC is a tailored product. However, it was very important to use it to the first RESVEP execution to prove its feasibility and use the results as feedback to improve the model.

Specific Objectives. The following objectives set elements which have to be obtained in the RESVEP execution:

1. To prove the feasibility of the RESVEP.

2. To obtain quantitative data of the value indicators.

3. To obtain the value state of the software product.

4. To change or adjust the value indicators and its measurements as required.

4.1.2 ISPC: Gather Knowledge About the Project

4.1.2.1 Background

Several universities face the challenge to measure the academic level of incoming students for their undergraduate programs in engineering and science. Specifically, at the Universidad Autónoma Metropolitana this is important for making decisions about the inclusion of low profile incoming students in preparatory courses. For this matter, a measurement instrument has been designed based on the institutional profile.

This instrument has been implemented as an exam including modules, such as mathematics, communication skills and problem solving. In this fashion, each question belongs to a particular profile section and it is assigned to a previously defined taxonomic level, which expresses a deep knowledge for answering. A particular instrument version is assigned to a 30 student group. Different people are involved in this process: students, preparatory courses coordinator, questions designers, reviewers, exam monitors and the results analyzer. Considering all the activities that each instrument needs (design, review, corrections, etc.), this process can be exhaustive, considering the 500 incoming students in each term. Therefore, the proposed solution is to support those activities with a software system, whose development started a year ago (of the value estimation date) with computer science students. A first system version, named Instrument System for Preparatory Courses (ISPC), is under development and some software components had been tested already. This case study use the main ISPC's components for applying the value estimation reference model presented in this thesis.

4.1.3 ISPC: Delimitation of the Work Products to Estimate

The name of the system for the case study is Instrument System for Preparatory Courses (ISPC), but only one big component of the system is going to be estimated, the *User Management Component* (UMC), which is going to be estimated for its release 1.0 at 66% of its development. The *user management* component at its release 1.0 (UMC 1.0) divides into the sub-components described in Table 11.

Table 11: Sub-components of the UMC 1.0 and its description

Sub-Component	**Description**
Add User's From File	Add multiple users through an excel file, the file should be verified to have a predefined format of the users data, and after verification save it to the database, preventing user data duplication.
Add User	Add single user verifying the data and without duplicating user data.
User Query	This sub-component comprises the execution of the query of users, but also comprises a user filter to find the desired user.
User Edit	On this sub-component the user data fields can be edited, through the user query sub-component, and also users can be deleted.

At the review time, the main development advance was in the design workflow activities, and according to the project plan, the UMC should already be implemented as a prototype.

4.1.3.1 Functions of ISPC 1.0

Table 12 describes a list of functions that the UMC 1.0 of the ISPC must provide; those functions according to the project plan are expected to be fully developed already. Also this list is 66% of the functions of UMC 1.0.

Table 12: Functions of the UMC 1.0

Sub-Component	**Function**
Add Users From File	Load and verify excel file
	Add users from file (through excel file)
Add User	Add new user
User Query	User filter
	Query execution
User Edit	Edit fields of user
	Delete user

4.1.4 ISPC: Project Context Definition

The development of the project of ISPC at the release 1.0 was on the initials stages of the software construction process, this case study was allocated between the completion of the software detailed design process and the beginning of the software construction process, the only expected work products to be completed are related to the User Management component, which was the one that is the subject of the value estimation of this case study. Table 13 describes the project context definition.

Table 13: Project context definition of the UMC 1.0 case study

Process	**Activity**	**Stakeholder**	**Work product**
Software Detailed Design [11]	Develop a detailed design for each software component of the software item [11]	Architect [13]	A detailed design of each software component, describing the software units to be built [11]
Software Construction [11]	Develop and document each software unit and database [11].	Implementer [11]	Software units defined by the design [11]

Figure 6 illustrates the instantiation of the RESVEP (see Figure 5), specific for the case study of the UMC 1.0.

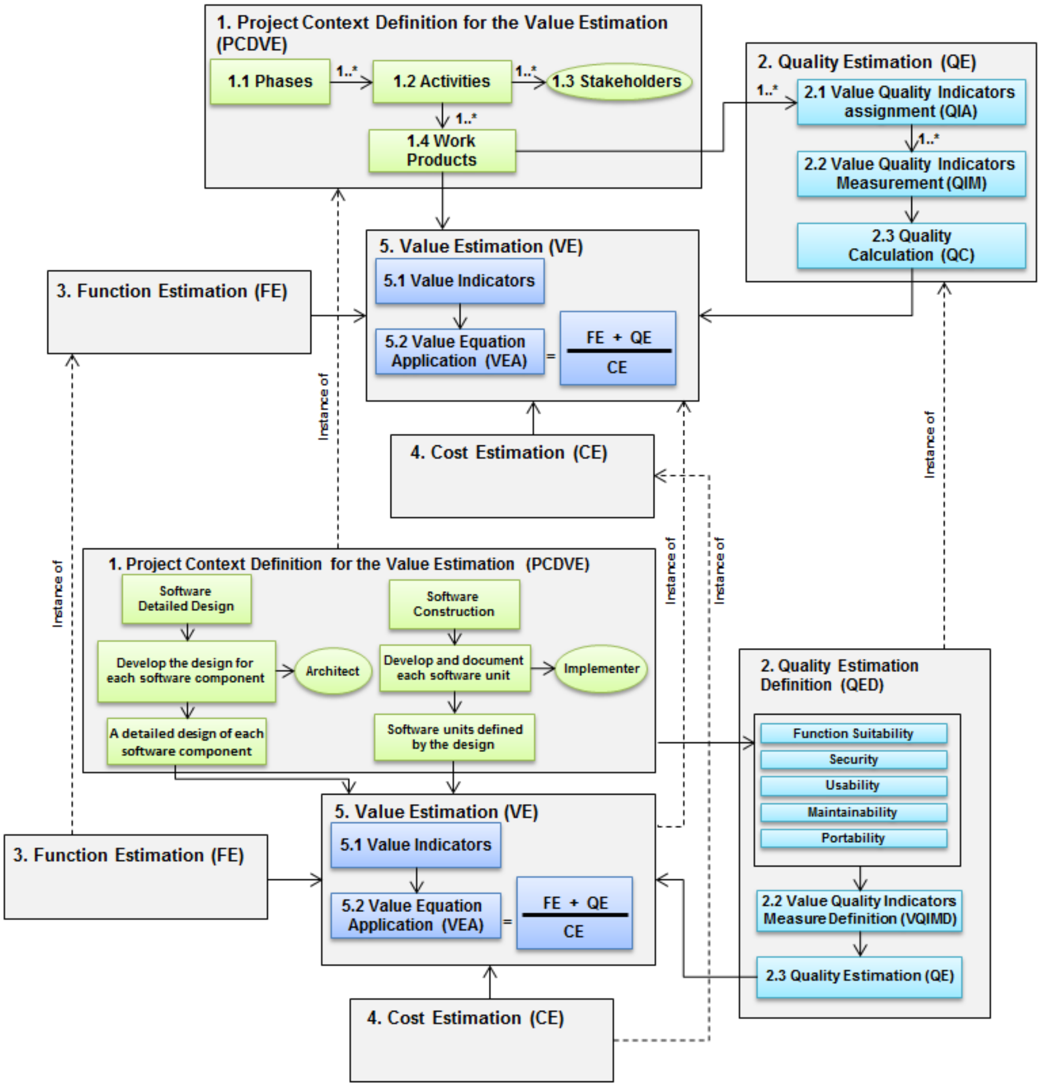

1. Project Context Definition for the Value Estimation (PCDVE)
1.1 Phases
1..*
1.2 Activities
1..*
1.3 Stakeholders
1..*
1.4 Work Products
2. Quality Estimation (QE)
1..*
2.1 Value Quality Indicators assignment (QIA)
1..*
2.2 Value Quality Indicators Measurement (QIM)
2.3 Quality Calculation (QC)
3. Function Estimation (FE)
5. Value Estimation (VE)
5.1 Value Indicators
5.2 Value Equation Application (VEA)
=
FE + QE
CE
4. Cost Estimation (CE)
Instance of
Instance of
Instance of
Instance of
Instance of
Instance of
1. Project Context Definition for the Value Estimation (PCDVE)
Software Detailed Design
Develop the design for each software component
Architect
A detailed design of each software component
Software Construction
Develop and document each software unit
Implementer
Software units defined by the design
2. Quality Estimation Definition (QED)
Function Suitability
Security
Usability
Maintainability
Portability
2.2 Value Quality Indicators Measure Definition (VQIMD)
2.3 Quality Estimation (QE)
3. Function Estimation (FE)
5. Value Estimation (VE)
5.1 Value Indicators
5.2 Value Equation Application (VEA)
=
FE + QE
CE
4. Cost Estimation (CE)

Figure 6: Instance of the reference model for software product value estimation process for the UMC 1.0 case study

4.1.5 ISPC: Delimitation of the Value Quality Indicators

This section comprises the step 2.1 of RESVEP, the *Value Quality Indicators Assignment (VQIA)*.

The architecture documentation was verified, which defined the value quality indicators assigned to the case study. The documentation reports the results of apply a method like Quality Attribute Workshop (QAW) to obtain quality attributes and their prioritization. They were considered in this work as Value Quality Indicators. To assign the expected measure to the value quality indicators the following steps were performed:

1. There were a total of 14 stakeholders in the architecture quality attributes prioritization. Each stakeholder had 6 votes, but they could only spend 3 at maximum per quality attribute. Also there was a prioritization from the customer and architecture design point of view (the V is used for votes).

2. A customer priority list was provided; if an attribute had high priority, it would get 6 points, with medium priority, it would get 4 points and with low priority, it would get 2 points (the CP is for customer priority).

3. An architect point of view priority list was provided; if an attribute had a high priority it adds 6 points, medium priority adds 4 points and low priority adds 2 points (the AP is for architect priority).

4. The total points are divided by 45 (42 because it is the maximum votes that an attribute can get plus a factor of 3 for the prioritization).

The equation to obtain the expected measure is $\frac{(V+CP+AP)}{45}$. Example for usability: V = 9, CP = 8 and AP = 8, therefore $\frac{9+8+8}{45} = 0.3777$, and the expected measure for usability is 0.37. Table 14 summarizes this exercise for each quality attribute.

Table 14: Quality attributes percentages of the UMC 1.0

Quality attributes	**Votes by Stakeholders (V)**	**Customer Priority (CP)**	**Architect Priority (AP)**	**Total (T=V+ CP+AP)**	**Assigned Measure (T/45)**
Functional Suitability					1.0
Usability	9	Medium	Medium	17	0.37
Portability	6	Medium	Medium	14	0.31
Maintainability	9	Medium	Low	15	0.33
Security	32	Medium	Low	38	0.84

Table 15 lists the value quality indicators that are going to be measured, to verify if they reach their expected measure.

Table 15: Value quality indicators under study for the UMC 1.0

Value Quality Indicator	**Expected Measure**
FS: Functional suitability	1.00
S: Security	0.84
U: Usability	0.37
M: Maintainability	0.33
P: Portability	0.31

4.1.6 ISPC: Value Quality Indicators Measurement and Quality Calculation

4.1.6.1 Value Quality Indicators Measurement (VQIM)

Functional suitability. The degree to which the product provides functions that meet stated and implied needs when the product is used under specified conditions. [12]

Functional Suitability has four sub-characteristics: x1, x2, x3, and x4, which are described in Table 16. Also, the measurement procedures for x1, x2, x3 and x4, are described in the Tables 17, 18 and 19. The evaluated sub-characteristics of functional suitability, and the equation of functional suitability are in Table 16.

Table 16: Sub-characteristics of functional suitability evaluated for the UMC 1.0

Identifier	**Sub-characteristics**	**Evaluation**
x1	Traceability to the requirements and design of the software item	See Table 17
x2	External consistency	See Table 18
x3	Internal consistency	See Table 19
x4	Appropriateness of coding methods and standards used	See Table 20
Equation of functional suitability: $FS = \frac{x1+x2+x3+x4}{4} = 0.375$		

Table 17: UMC 1.0: Traceability to the requirements and design of the software item

Characteristic	**Data**
Description	The product is only traceable if allows to identify and reference clearly the requirement it satisfies. [11]
Measurement and Equation	$$x1 = \frac{A}{B}$$ x1 = Traceability to the requirements and design of the software item. A = Number of traceable design items confirmed in review. B = Number of items checked. Interpretation of the result: $0 <= fs1 <= 1$ The closer to 1, the better. Measurement based on the standard [10].
Measurement Method	Documentation, requirements and design from the system were analyzed and contrasted with the developed functions of the user management component. Traceability tools or traceability matrix were requested.
Result	It was concluded that the 4 sub-components of UMC 1.0 were not traceable. Therefore: $$x1 = \frac{A}{B} = \frac{0}{4} = 0$$
Limitations	There were no limitations to evaluate this indicator.
Problems	There is a lack for a traceability tool, or a traceability matrix between requirements and design.

Table 18: UMC 1.0: External consistency

Characteristic	**Data**
Description	The level of consistency between the requirements and the design of the software. [11]

Continued on Next Page...

Table 18 – Continued

Characteristic	Data
Measurement and Equation	$$x2 = 1 - \frac{A}{B}$$ x2 = External consistency. A = Number of missing components detected in evaluation. B = Number of components described in the requirements of the software item. Interpretation of the result: $0 <= X2 <= 1$ The closer to 1, the better. Measurement based on the standard [10].
Measurement Method	A verification of the designed functions that the component must provide according to the requirements.
Result	There are 2 sub-components missing of the 4 evaluated. Therefore: $$x2 = 1 - \frac{A}{B} = 1 - \frac{2}{4} = 1 - 0.5 = 0.5$$
Limitations	There were no limitations to evaluate this indicator.
Problems	Missing sub-components: • User query. • User edit.

Table 19: UMC 1.0: Internal consistency

Characteristic	Data
Description	The level of consistency between the designed software units. [11]
Measurement and Equation	$$x3 = 1 - \frac{A}{B}$$ x3 = Internal consistency. A = Number of internal inconsistent software components detected in evaluation. B = Number of components described in the software item on the requirements. Interpretation of the result: $0 <= X3 <= 1$ The closer to 1, the better. Measurement based on the standard [10].
Measurement Method	A verification consistency between designed units was performed.
Result	There are 2 inconsistent components of the 4 evaluated. Therefore: $$x3 = 1 - \frac{A}{B} = 1 - \frac{2}{4} = 1 - 0.5 = 0.5$$
Limitations	There were no limitations to evaluate this indicator.
Problems	There is no consistency for 2 sub-components described in design. • User query. • User edit.

Table 20: UMC 1.0: Appropriateness of coding methods and standards used

Characteristic	Data
Description	The degree on which the software code and standards used on a product are appropriate. [11]

Continued on Next Page...

Table 20 – Continued

Characteristic	**Data**
Measurement and Equation	$$x4 = 1 - \frac{A}{B}$$ x4 = Appropriateness of coding methods and standards used. A = Number of components with incorrect coding methods and standards. B = Number of components evaluated. Interpretation of the result: 0 $<= X4 <=$ 1 The closer to 1, the better. Measurement based on the standard [10].
Measurement Method	A verification of the code of the user management sub-component's was performed.
Result	The lack of organization between folders and classes lowers the result of this aspect; but inside the classes the code standards are followed. Therefore: $$x4 = 1 - \frac{A}{B} = 1 - \frac{2}{4} = 1 - 0.5 = 0.5$$
Limitations	There were no limitations to evaluate this indicator.
Problems	Disorganization in the files of the folders is a problem to measure this indicator.

Summary of Functional Suitability Measurement. To obtain the functional suitability measurement result, the Equation (9) is applied. Figure 7 displays the results of functional suitability and its indicators.

$$FS = \frac{x1 + x2 + x3 + x4}{4} = \frac{0 + 0.5 + 0.5 + 0.5}{4} = 0.375 \tag{9}$$

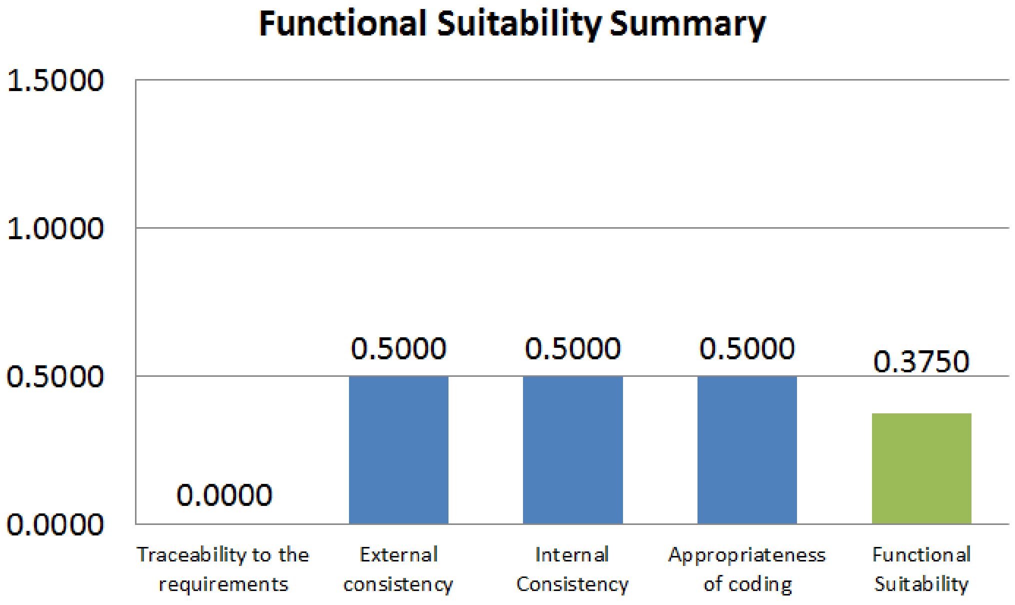


Figure 7: ISPC: Measurements summary of functional suitability and its sub-characteristics

Security. It is the degree of protection of information and data, so that unauthorized persons or systems cannot read or modify them, and authorized persons or systems are not denied the access to them. [12]

The evaluated sub-characteristics of security are s1, s2 and s3, which are described with the equation of security in Table 21. Also, the measurement procedures for s1, s2 and s3, are described in Tables 22, 23 and 24.

Table 21: Sub-characteristics of security evaluated for the UMC 1.0

Identifier	**Sub-characteristics**	**Evaluation**
s1	Confidentiality	See Table 22
s2	Integrity	See Table 23
s3	Authenticity	See Table 24
Equation of security: $S = \frac{s1+s2+s3}{3} = 0.8633$		

Table 22: UMC 1.0: Confidentiality

Characteristic	**Data**
Description	The degree of protection from unauthorized disclosure of data or information, whether accidental or deliberate [12]
Measurement and Equation	$$s1 = 1 - \frac{A}{B}$$ s1 = Confidentiality. A = Number of components that are not secure from having unauthorized disclosure of data or information, whether accidental or deliberate. B = Total components that handles data. Interpretation of the result: $0 <= s1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	A verification of the confidentiality of the user management component was performed.
Result	All of the sub-components functions have confidentiality on their design, every user has a defined role, permission, etc. Therefore: $$s1 = 1 - \frac{A}{B}$$ $$s1 = 1 - \frac{0}{4}$$ $$s1 = 1$$
Limitations	There were no limitations to evaluate this indicator.
Problems	There were no problems for this indicator.

Table 23: UMC 1.0: Integrity

Characteristic	**Data**
Description	The degree to which a system or component prevents unauthorized access to, or modification of computer programs or data. [12]
Measurement and Equation	$$s2 = \frac{A}{B}$$ s2 = Integrity. A = Number of components that prevents unauthorized access to, or modification. B = Total number component that handles data. Interpretation of the result: $0 <= s4 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	The design was verified to check if the sub-components prevent unauthorized access or modification.
Result	It was found that the system uses rules of validation. Therefore: $$s2 = \frac{A}{B} = \frac{4}{4} = 1$$
Limitations	There were no limitations to check this indicator.
Problems	Problems were not found for this indicator.

Table 24: UMC 1.0: Authenticity

Characteristic	**Data**
Description	The degree to which the identity of a subject or resource can be proved to be the one claimed. [12]

Continued on Next Page...

Table 24 – Continued

Characteristic	**Data**
Measurement and Equation	$$s3 = \frac{A}{B}$$ s3 = Authenticity. A = Resource or subject Identification (Real Percentage of veracity for assuring this identification). B = Resource or subject Identification (Expected Percentage of veracity for assuring this identification). Interpretation of the result: 0 <= $s5$ <= 1 The closer to 1, the better. Measurement of the standard [10].
Measurement Method	The system has a login, the measurement method was to verify if the *User Management* sub-components functions used the login to operate them.
Result	The session with a login works, but the only information that the authenticity mechanism implemented offers is the username. Therefore: $$s3 = \frac{A}{B} = \frac{50\%}{84\%} = 0.59$$
Limitations	There were no problems to evaluate this indicator.
Problems	Session works, but it does not offer more information except which user is online.

Summary of Security Measurement. To obtain the security estimated measure, the Equation (10) is applied. Figure 8 displays the results of security and its indicators.

$$S = \frac{s1 + s2 + s3}{3} = \frac{1 + 1 + 0.59}{3} = 0.8633 \tag{10}$$

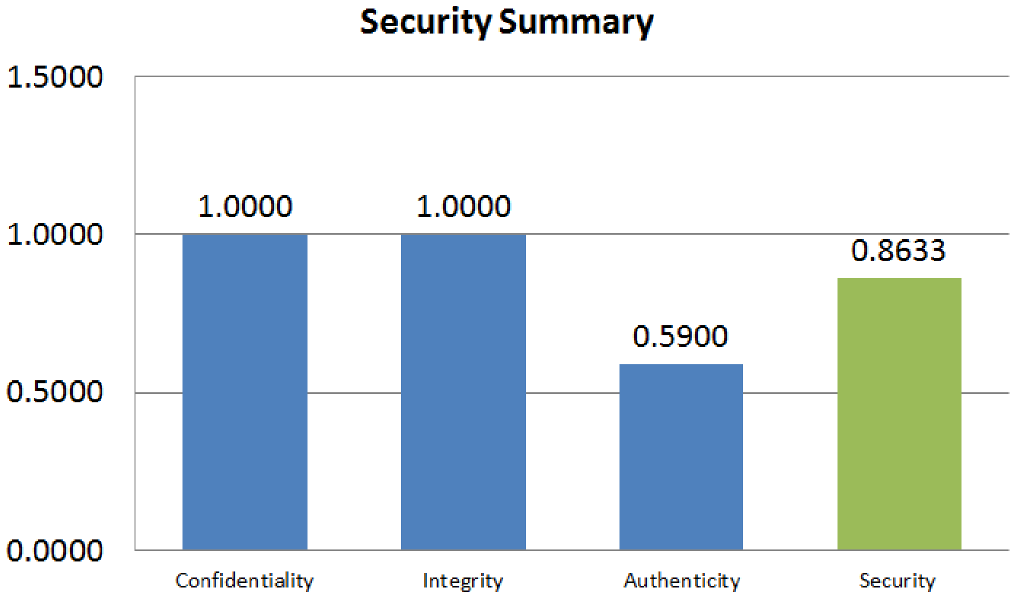


Figure 8: ISPC: Measurements summary of security and its sub-characteristics

Usability. The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. [12]

The evaluated sub-characteristics of usability are u1, u2, u3 and u4, which are described with the equation of usability in Table 25. Also, the measurement procedures for u1, u2, u3 and u4 are described in Tables 26, 27, 28 and 29.

Table 25: Sub-characteristics of usability evaluated for the UMC 1.0

Identifier	**Sub-characteristics**	**Evaluation**
u1	Learnability	See Table 26
u2	Ease of use	See Table 27
u3	User error protection	See Table 28
u4	User interface aesthetics	See Table 29
Equation of usability: $U = \frac{u1+u2+u3+u4}{4} = 0.414$		

Table 26: UMC 1.0: Learnability

Characteristic	**Data**
Description	The degree to which a product can be used by specified users to achieve specified learning goals with effectiveness, efficiency, safety and satisfaction in a specified context of use. [12]
Measurement and Equation	$$u1 = \frac{t1}{t2}$$ u1 = Learnability. t1 = Mean expected-time taken to learn a software unit correctly. t2 = Mean real-time taken to learn a software unit correctly. Interpretation of the result: $0 <= u1 <= N$ If equal to 1 or higher, the better. Measurement of the standard [10]. If more than one unit of software is evaluated, there will be a summation of t1 of each unit, and this also applies to t2.
Measurement Method	The expected average time was specified by the product manager according to product requirements. The actual average time was measured with the aid of a chronometer.

Continued on Next Page...

Table 26 – Continued

Characteristic	**Data**
Result	The mean expected-time to learn to use the 4 sub-components evaluated was 3 minutes per each component, which means for all components the mean expected time in second was 720, and the real mean-time to learn to use each component assessed was: - Add users from file = 480 seconds. - Add user = 180 seconds. - User query = 300 seconds. - User edit = 240 seconds The real mean time is equal to 1200 seconds. Therefore: $u1 = \frac{t1}{t2} = \frac{720}{1200} = 0.6$
Limitations	The evaluator's skills and perception influence this measurement method by the nature of the sub-characteristic of learnability.
Problems	The sub-component add group needed more time to learn, especially the part of adding users from excel file.

Table 27: UMC 1.0: Ease of use

Characteristic	**Data**
Description	The degree to which users find the product easy to operate and control [12]
Measurement and Equation	$u2 = \frac{t1}{t2}$ u2 = Ease of use. t1 = Mean expected-time taken to use a software unit correctly. t2 = Mean real-time taken to use a software unit correctly. Interpretation of the result: $0 <= u2 <= N$ If equal to 1 or higher, the better. Mapped equation from the standard [10]. If more than one unit of software is evaluated, there will be a summation of t1 of each unit, and this also applies to t2.
Measurement Method	The expected average time was specified by the product manager according to product requirements. The actual average time was measured with the aid of a chronometer.
Result	The mean expected-time to use the 4 components evaluated was 5 minutes (300 seconds), after the measurement the component resulted as follows: - Add users from file = 243 seconds. - Add user = 66 seconds. - User query = 110 seconds. - User edit = 121 seconds And the real mean time to use the functionalities was 308. Therefore: $u2 = \frac{t1}{t2} = \frac{300}{540} = 0.5556$
Limitations	The evaluator skills and perception influence this measurement method by the nature of the sub-characteristic of ease of use.
Problems	The part of add users from file is the most difficult part to understand.

Table 28: UMC 1.0: User error protection

Characteristic	**Data**
Description	The degree to which the system protects users against making errors [12]
Measurement and Equation	$$u3 = \frac{A}{B}$$ u3 = User error protection. A = Number of user-interface units that protects users against making errors. B = Number of user-interface units evaluated. Interpretation of the result: $0 <= u3 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	The sub-components that were evaluated had user input in the user interfaces, with the following criteria: 1. In text boxes: limit of accepted characters and special characters accepted.
Result	There were no user interfaces with user error protection. Therefore: $$u3 = \frac{A}{B} = \frac{0}{6} = 0$$
Limitations	There were no limitations to evaluate this indicator.
Problems	There are no user error protection in the developed interfaces, but also it is not stated on the design.

Table 29: UMC 1.0: User interface aesthetics

Characteristic	**Data**
Description	The degree to which the user interface enables pleasing and satisfying interaction for the user. [12]
Measurement and Equation	$$u4 = \frac{A}{B}$$ u4 = User interface aesthetics. A = Number of user-interface units that enables pleasing and satisfying interaction for the user B = Number of user-interface units evaluated. Interpretation of the result: $0 <= u4 <= 1$ The closer to 1, the better. Mapped equation from the standard [10].
Measurement Method	This indicator was evaluated totally by the preference of the evaluator regarding the aesthetics of the user interfaces.
Result	Only 3 user-interfaces could be evaluated of the expected 6 user-interfaces (user-interfaces measured: main interface, add new user interface and add users from file interface), the rest of the user-interfaces were incomplete. Therefore: $$u4 = \frac{A}{B} = \frac{3}{6} = 0.5$$
Limitations	There were no limitations to evaluate this indicator.
Problems	The following user-interfaces were incomplete: • User query interface. • User edit interface. • User filter interface.

Summary of Usability Measurement. To obtain the usability estimated measure, the Equation (11) is applied. Figure 9 displays the results of usability and its indicators.

$$U = \frac{u1 + u2 + u3 + u4}{4} = \frac{0.6 + 0.556 + 0 + 0.5}{4} = 0.414 \tag{11}$$

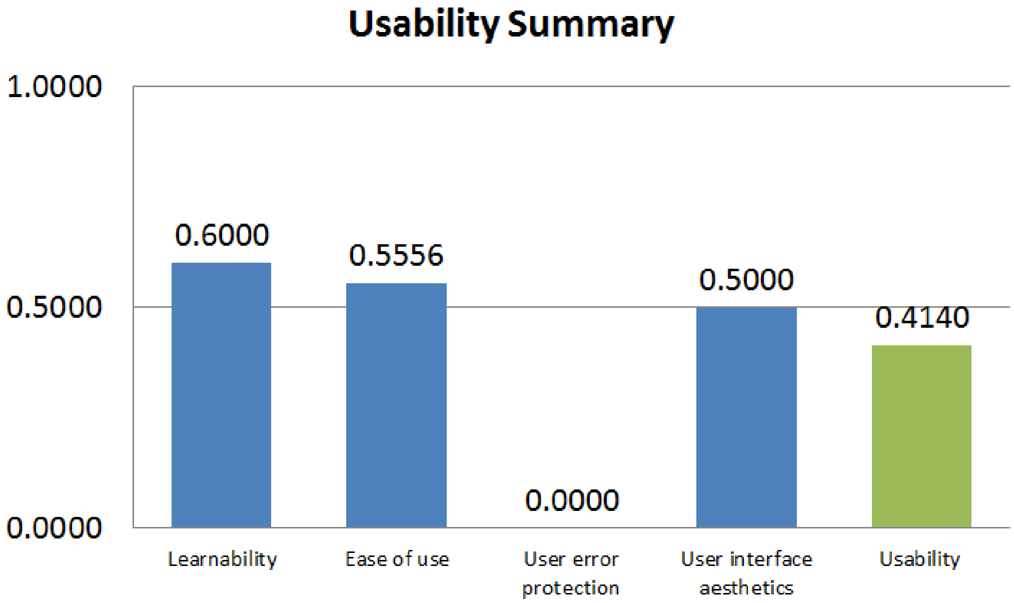


Figure 9: ISPC: Measurements summary of usability and its sub-characteristics

Maintainability. The degree of effectiveness and efficiency with which the product can be modified. [12]

The evaluated sub-characteristics of maintainability are m1, m2, m3, m4 and m5, which are described with the equation of maintainability in Table 30. Also, the measurement procedures for m1, m2, m3, m4 and m5 are described in Tables 31, 32, 33, 34 and 35.

Table 30: Sub-characteristics of maintainability evaluated for the UMC 1.0

Identifier	**Sub-characteristics**	**Evaluation**
m1	Modularity	See Table 31
m2	Reusability	See Table 32
m3	Analyzability	See Table 33
m4	Modifiability	See Table 34
m5	Testability	See Table 35
Equation of maintainability: $M = \frac{m1+m2+m3+m4+m5}{5} = 0.25$		

Table 31: UMC 1.0: Modularity

Characteristic	**Data**
Description	The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components. [12]
Measurement and Equation	$$m1 = 1 - \frac{A}{B}$$ m1 = Modularity. A = Number of components that have high impact when they are modified. B = Total number of components. Interpretation of the result: $0 <= m1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	Documentation of the system, architecture and relation between functions was analyzed.
Result	Of the 4 sub-components evaluated, 3 have a high impact when they are modified. Therefore: $$m1 = 1 - \frac{A}{B} = 1 - \frac{3}{4} = 0.25$$
Limitations	There were no limitations to evaluate this indicator.
Problems	The sub-components: add user, add users from file, and user edit have a high impact when they are modified.

Table 32: UMC 1.0: Reusability

Characteristic	**Data**
Description	The degree to which an asset can be used in more than one software system, or in building other assets. [12]
Measurement and Equation	$$m2 = \frac{A}{B}$$ m2 = Reusability. A = Number of components that are reusable. B = Total number of components. Interpretation of the result: $0 <= m2 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	Documentation and architecture of the system was analyzed to look out for reusable functions or components.

Continued on Next Page...

Table 32 – Continued

Characteristic	Data
Result	Of the 4 sub-components evaluated, 2 are reusable: add user and user query. Therefore: $$m2 = \frac{A}{B} = \frac{2}{4} = 0.5$$
Limitations	There were no limitations to evaluate this indicator.
Problems	There are no significant problems in this indicator, since some functions it is normal that they cannot be reusable.

Table 33: UMC 1.0: Analyzability

Characteristic	Data
Description	The ease with which the impact of an intended change on the rest of the software can be assessed, or the software product can be diagnosed for deficiencies or causes of failures in the software, or the parts to be modified can be identified. [12]
Measurement and Equation	$$m3 = \frac{A}{B}$$ m3 = Analyzability. A = Number of components that can be easy diagnosed for deficiencies or causes of failures in the component model. (Helped by comments, version, code standards, etc.) B = Total number of components. Interpretation of the result: $0 <= m3 <= 1$ The closer to 1, the better. Measurement of the standard: [10].
Measurement Method	Files, code were verified with the design of user management component.
Result	Of the 4 sub-components evaluated, 1 can be easy diagnosed for deficiencies. Therefore: $$m3 = \frac{A}{B} = \frac{1}{4} = 0.25$$
Limitations	There were no limitations to evaluate this indicator.
Problems	On the functions: add user, add group and user edit the code is not well documented and the files are not properly organized for its analyzability.

Table 34: UMC 1.0: Modifiability

Characteristic	Data
Description	The degree to which a product can be effectively and efficiently modified without introducing defects or degrading performance. [12]
Measurement and Equation	$$m4 = \frac{A}{B}$$ m4 = Modifiability. A = Number of components that can be modified without introducing defects or degrading performance. B = Total number of components. Interpretation of the result: $0 <= m4 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	Documentation and architecture of the system were analyzed.
Result	Of the 4 sub-components evaluated, 1 can be modified without introducing defects: User query. Therefore: $$m4 = \frac{A}{B} = \frac{1}{4} = 0.25$$
Limitations	There were no limitations to evaluate this indicator.
Problems	Almost all sub-components (except user query) are strongly coupled, therefore when they are modified have an big impact on each other.

Table 35: UMC 1.0: Testability

Characteristic	**Data**
Description	The ease with which test criteria can be established for a system or component and tests can be performed to determine whether those criteria have been met. [12]
Measurement and Equation	$$m5 = \frac{A}{B}$$ m5 = Testability. A = Number of cases in which a component can be tested appropriately. B = Number of cases of component tests. Interpretation of the result: 0 <= $m5$ <= 1 The closer to 1, the better. Measurement of the standard [10].
Measurement Method	The user management component was verified on order to look for a test environment.
Result	No test environment is available. Therefore m5 = 0: $$m5 = 0$$
Limitations	Could not be measured, due to lack an test environment.
Problems	There is no test environment for the user management component

Summary of Maintainability Measurement. To obtain the maintainability measurement result, the Equation 12 is applied. Figure 10 displays the results of maintainability and its indicators.

$$M = \frac{m1 + m2 + m3 + m4 + m5}{5} = \frac{0.25 + 0.5 + 0.25 + 0.25 + 0}{5} = 0.25 \qquad (12)$$

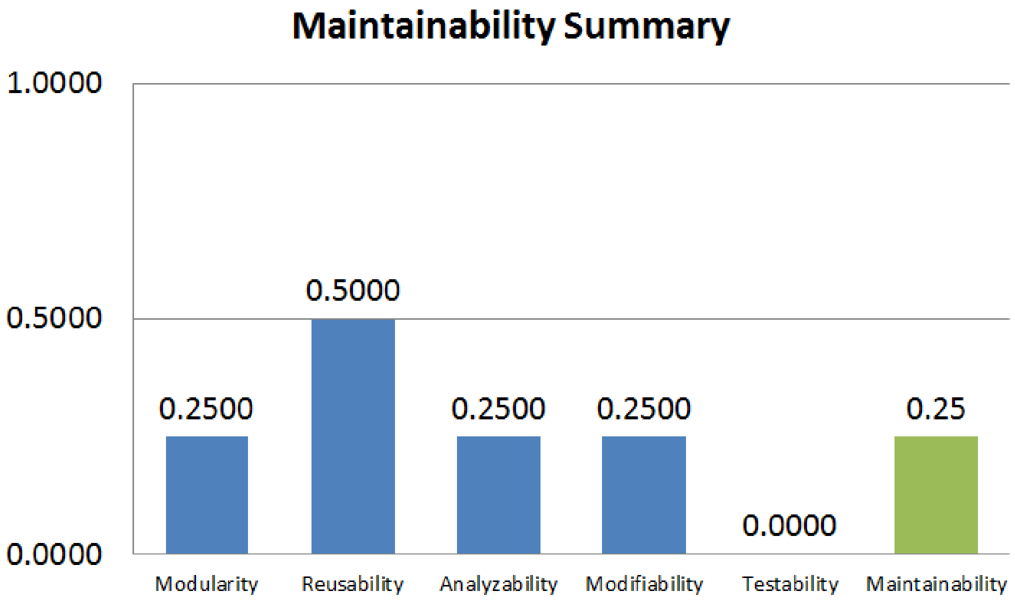


Figure 10: ISPC: Measurements summary of maintainability and its sub-characteristics

Portability. The degree to which a system or component can be effectively and efficiently transferred from one hardware, software or other operational or usage environment to another. [12]

The evaluated sub-characteristics of portability are p1 and p2 which are described with the equation of portability in Table 36. Also, the measurement procedures for p1 and p2 are described in Tables 37 and 38.

Table 36: Sub-characteristics of portability evaluated for the UMC 1.0

Identifier	**Sub-characteristics**	**Evaluation**
p1	Adaptability	See Table 37
p2	Replaceability	See Table 38
Equation of portability: $P = \frac{p1+p2}{2} = 0.5$		

Table 37: UMC 1.0: Adaptability

Characteristic	**Data**
Description	The degree to which the product can effectively and efficiently adapted for different specified hardware, software or other operational or usage environments. [12]
Measurement and Equation	$$p1 = \frac{Hi + Si}{2}$$ p1 = Adaptability. 1. Hi = Hardware independence $Hi = 1 - \frac{A}{B}$ A = Number of components of which tasks were not completed or not enough resulted to meet adequate levels during combined operating testing with environmental hardware. B = Total number components in the component model. Interpretation of the result: $0 <= Hi <= 1$ The closer to 1, the better. Measurement of the standard [10]. 2. Si = Software independence $Si = 1 - \frac{A}{B}$ A = Number of components of which tasks were not completed or were not enough resulted to meet adequate level during combined operating testing with operating system software or concurrent application software. B = Total number components in the component model. Interpretation of the result: $0 <= Si <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	The sub-components were tested in different computers (hardware environments). Since it is a web application the software independence test was performed through different web browsers that access the system.
Result	Hi = Hardware independence $Hi = 1 - \frac{A}{B} = 1 - \frac{0}{4} = 1$ Si = Software independence $Si = 1 - \frac{A}{B} = 1 - \frac{0}{4} = 1$ Therefore for adaptability: $p1 = \frac{Hi+Si}{2} = \frac{1+1}{2} = \frac{2}{2} = 1$
Limitations	There were no limitations to evaluate this indicator.
Problems	No significant problems were found for this indicator.

Table 38: UMC 1.0: Replaceability

Characteristic	**Data**
Description	The degree to which the product can be used in place of another specified software product for the same purpose in the same environment. [12]
Measurement and Equation	$$p2 = \frac{A}{B}$$ p2 = Replaceability. A = Number of components that can replace another for the same purpose in the same environment. B = Total number components in the component model. Interpretation of the result: $0 <= p2 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	A verification of the sub-components was performed, looking for interfaces or any device to replace sub-components easily.
Result	None of the sub-components is easily replaced. Therefore: $$p2 = \frac{A}{B} = \frac{0}{4} = 0$$
Limitations	There were no limitations to evaluate this indicator.
Problems	The functions and sub-components are not constructed as stated in design and they do not have an interface to easy connection and replacement.

Summary of Portability Measurement. To obtain the portability estimated measure, the Equation (13) is applied. Figure 11 displays the results of portability and its indicators.

$$P = \frac{p1 + p2}{2} = \frac{1 + 0}{2} = 0.5 \tag{13}$$

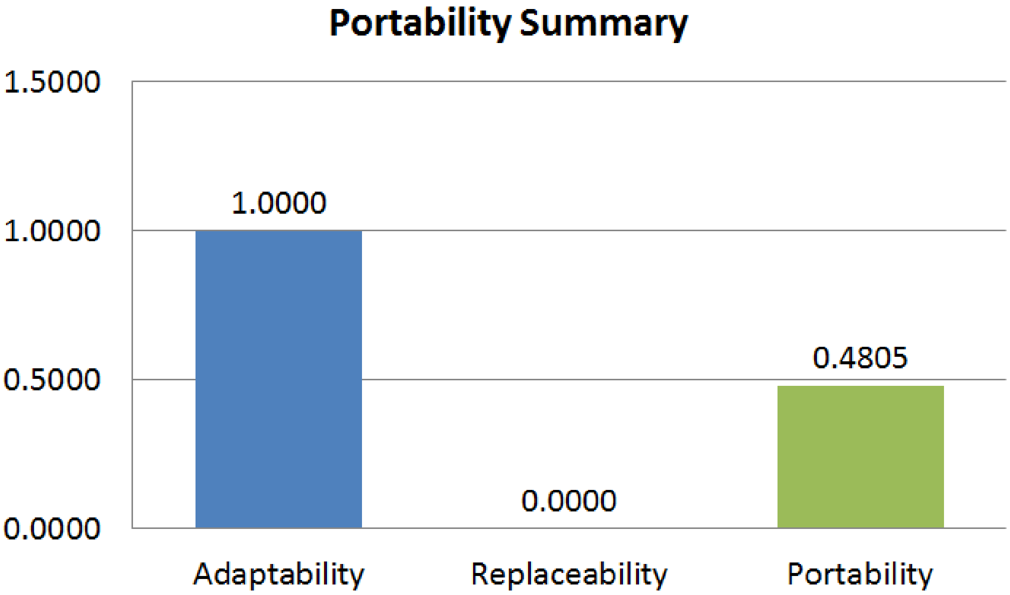


Figure 11: ISPC: Measurements summary of portability and its sub-characteristics

4.1.6.2 Quality Calculation (QC)

The quality estimation of the ISPC *User Management Component* is defined by the Equation 14, which averages the value quality indicators divisions of Estimated Measure (ES) by the expected measure (EX). The results of the ES by EX division are normalized to 1.0 if the ES is higher than the EX (as stated in Section 3.3.2.3, and discussed in Section 4.1.10.1).

$$Q = \frac{\frac{FS_{ES}}{FS_{EX}} + \frac{S_{ES}}{S_{EX}} + \frac{U_{ES}}{U_{EX}} + \frac{M_{ES}}{M_{EX}} + \frac{P_{ES}}{P_{EX}}}{5} \tag{14}$$

$$Q = \frac{\frac{0.3750}{1.0000} + \frac{0.8633}{0.8400} + \frac{0.4140}{0.3700} + \frac{0.2500}{0.3300} + \frac{0.5000}{0.3100}}{5}$$

$$Q = \frac{0.3750 + 1.0000 + 1.0000 + 0.7576 + 1.0000}{5} = \frac{4.1326}{5} = 0.8265$$

In Table 39 a summary of the results of the quality estimation, its ratio and exceeded or missed percentage. Figure 12 provides a graphic of the results of the indicators.

Table 39: Indicators with expected measures, estimated measurements, ratio and exceeded or missing percentage of the UMC 1.0

Indicator	**Expected measure**	**Estimated measure**	**Ratio**	**Exceeded or missed percentage from measures**
FS: Function suitability	1.0000	0.3750	0.3750	−62.50%
x1: Traceability to the requirements and design of the software item		0.0000		
x2: External consistency		0.5000		
x3: Internal consistency		0.5000		
x4: Appropriateness of coding methods and standards used		0.5000		
S: Security	0.8400	0.8633	1.0277	+2.33%
s1: Confidentiality		1.0000		
s2: Integrity		1.0000		
s3: Authenticity		0.5900		
U: Usability	0.3700	0.4140	1.1189	+4.4%
u1: Learnability		0.6000		
u2: Ease of use		0.5556		
u3: User error protection		0.0000		
u4: User interface aesthetics		0.5000		
M: Maintainability	0.3300	0.2500	0.7576	−8.0%
m1: Modularity		0.2500		
m2: Reusability		0.5000		
m3: Analyzability		0.2500		
m4: Modifiability		0.2500		
m5: Testability		0.0000		
P: Portability	0.3100	0.5000	1.6129	+19.00%
p1: Adaptability		1.0000		
p2: Replaceability		0.0000		

Continued on Next Page…

Table 39 – Continued

Indicator	**Expected measure**	**Estimated measure**	**Ratio**	**Exceeded or missed percentage from measures**
Averages	0.5700	0.4805	1.13756	−8.95%
Quality (ratio average, normalizing the ones that exceed 1.0**)**			0.8265	

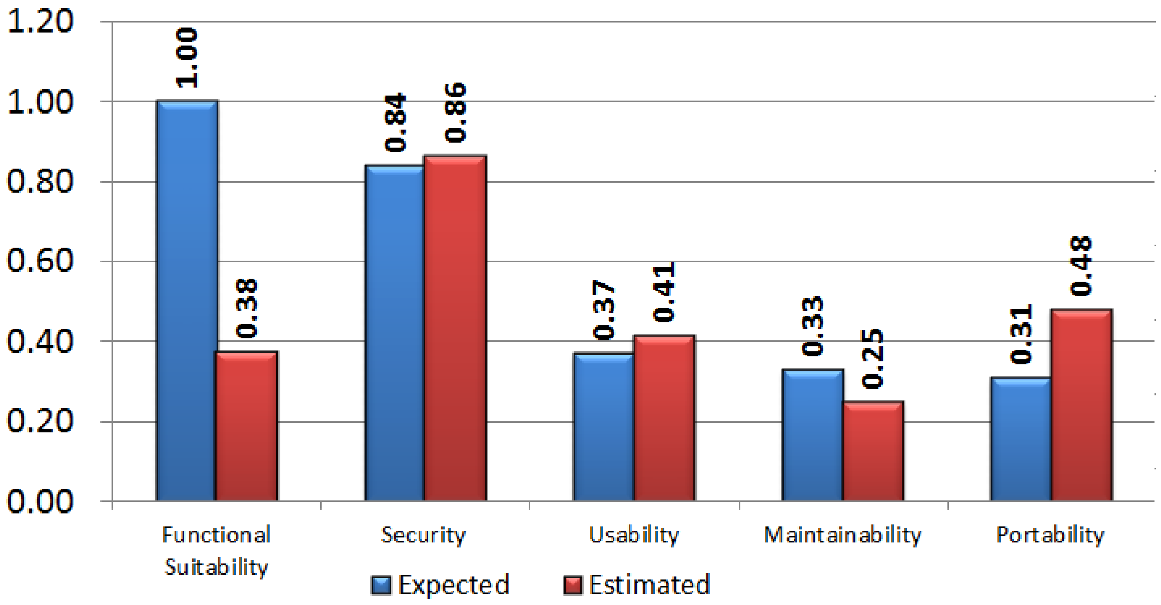


Figure 12: ISPC: Graphic of the value quality indicators

4.1.7 ISPC: Function Verification

4.1.7.1 Expected function

In this section, the expected functionality that the product must deliver against the real functionality that has been developed is explained. Table 40 lists the functions that the *User Management Component* of the *Instrument System for Preparatory Courses* must provide, and, the functions that must be fully developed at the Release 1.0. Table 40 also lists the adjusted function points obtained using the *Function Point Modeler* software [21], which is based on the *Function Point Counting Practices Manual* [9].

Table 40: List of functions of the UMC 1.0 with their adjusted function points, and the development information

Expected Functions	**Adjusted function points**	**Developed**
Add User From File		
Save group of users to database	4	Yes
Convert excel file	4	Yes
Add usrs from file interface	4	Yes
Excel file management	10	Yes
Add user		
Add new user	4	Yes
Add new user interface	4	Yes
User query		
User query interface	4	Yes
Query execution	4	No
Query result	4	No
User filter	4	Yes
User filter interface	4	Yes
User edit		
User edit interface	4	Yes
Edit user	4	No
Delete user	4	No
Util		
Main interface	4	Yes
User DB	10	Yes
Total adjusted function points	**76**	

As stated in Table 40 the expected function points for the ISPC at release 1.0 are: 76.

4.1.7.2 Developed function

The functions in Table 41 were not fully developed as expected. Therefore summarizing the function points of the functions that were not fully developed the result is 16 function points.

Table 41: UMC functions that were not fully developed at the release 1.0

Function	**Adjusted function points**
Query execution	4
Query result	4
Edit user	4
Delete user	4
Total not developed	16

Summarizing; the total expected function points were 76, but 16 of those function points were not fully developed. Therefore 60 function points were actually the real FP developed for the ISPC at the release 1.0. In Table 42 the data is summarized and the function ratio is described.

Table 42: Data of the UMC 1.0 function points and ratio

Results	**Data**
Expected function (EF)	76
Developed function (DF)	60
Function Ratio (DF/EF)	0.7895 (78.95%)

4.1.8 ISPC: Cost Verification

In the project documentation man-hours were estimated for each sub-component of the *User Management* component for the release 1.0 (Table 43).

Table 43: Man-hours estimated per sub-component of the UMC 1.0

Sub-Component	**Man/hours estimated**
Add users from file	10
Add user	10
User query	10
User edit	10
Total	**40**

According to the members of the development team, the real man-hours for the implemented *User Management* component at release 1.0 is 54. Therefore to obtain the ratio the real man-hours is divided by the estimated man-hours (developed on Equation 15).

$$Costratio = \frac{manhoursreal}{manhoursestimated} = \frac{54}{40} = 1.3 \tag{15}$$

Based on the Equation 15 the ratio cost is 1.3 (130%).

4.1.9 ISPC: Value Estimation

After the estimation of function, quality and cost for value estimation, Table 44 summarizes the results and the ratio of each element of the value equation.

Table 44: UMC 1.0 results of the elements of the value estimation equation

Elements from the value equation	**Expected (EX)**	**Estimated or real (ES)**	**Ratio (Es/Ex)**
Function	$76FP$	$60FP$	0.7895
Quality	100	82.65	0.8265
Cost	$40MH$	$54MH$	1.35

In Equation 18 (from [19]), the value of the component *User management* of ISPC at release 1.0 is calculated, based on the ratio of the different value indicators previously estimated (function, quality and cost).

$$Value = \frac{Function + Quality}{Cost} = \frac{0.7895 + 0.8265}{1.35} = 1.197 \qquad (16)$$

The value estimation result is 1.197. If it is taken into account that the value estimates of function, cost and quality should be 1, and applying the value equation, the value result should be 2, and the value obtained in Equation 16 is far of 2. Then, the expected value to the UMC was not achieved completely in the release 1.0. Figure 13 illustrates the difference between the value of *User Management* component and the reference value.

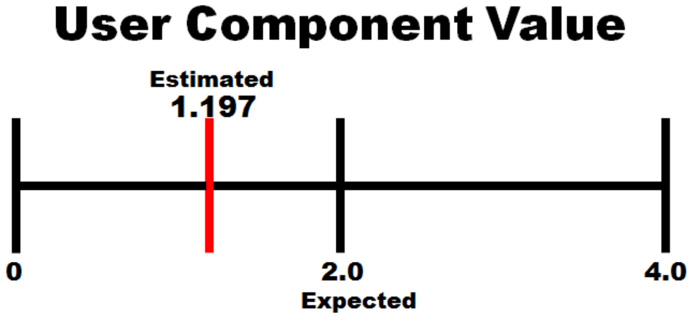


Figure 13: ISPC: Expected and estimated value

To obtain a different perspective, Table 45 describes the reached percentage of value and its elements.

Table 45: Estimated elements and their achieved percentages according to the expected measures of the UMC 1.0

Estimated elements	**Achieved percentage**
Function	78.95%
Quality	82.65%
Cost	135%
Value	59.85%

4.1.10 ISPC: Discussion

This section discusses the estimation of value and the value indicators of *User Management* component of the ISPC system at release 1.0.

4.1.10.1 Value Indicators Analysis

Function. The value indicator of function was very low on its estimation. It only reached a 78.95% of the expected functions to be developed at the release 1.0 of the *User Management* component, this is a red flag for the product manager and the development team; actions should be taken to solve this issue.

Cost. The cost for the development was exceeded by 35%, but also the function indicator was very low, this means that even that the functions were not fully developed the cost was higher than the expected for the entire functions. This means that there are a lot of problems with the development rate of the team, and special focus should be put in this.

Quality. According to the value quality indicators, quality was achieved at 82.65%, there are some problems with a few of the indicators. In Table 46 these problems are detailed indicator by indicator.

Table 46: Value quality indicators of the UMC 1.0, and their result discussion

Indicator	**Discussion**
Functional Suitability	This indicator was very low, mainly due to the functions that were not fully developed but were expected, as stated in the reviewed documentation.
Security	It reached the expected, one of the most important indicators for this product, had no problems in the estimations of its indicators.
Usability	There were no problems with this indicator, but if has to be taken into account that the expected measure was low.
Maintainability	The indicator of maintainability did not achieved what was expected but only by 8%, so there is not a lot of issues with this indicator, but some efforts should be made to achieve the expected measure.

Continued on Next Page...

Table 46 – Continued

Indicator	Discussion
Portability	The User Management component has no problems with portability, it achieved what was expected. This is mainly because the system is web based and normally portability is easy to achieve with this kind of applications.

Exceeded expected measures. Some of the value quality indicators estimation measure exceeded the expected measure, so the ratio of those indicators is higher than 1.0(100% in percentages). Those indicators that exceeded the expected measures are listed in Table 47.

Table 47: Value quality indicators of the UMC 1.0 and exceeded measure in percentages

Value quality indicators	Percentage
Portability	119.00%
Usability	104.4%
Security	102.33%

The only value quality indicator on which the estimated measure exceeds the expected considerably is portability, the other value quality indicators in Table 47 exceeded by a very small amount. There is very little from where to deviate efforts to the other indicators that were not achieved.

4.1.10.2 ISPC Value

Value reaches a 59.85% for the *User Component* of ISPC at release 1.0. This means that the developed product ended with a very low value, and decisions should be taken to modify the development process of the project to achieve the expected and wanted value. Function and cost have a high impact on the low value that the product has, the developed functions were low and costs were too high. Although quality is not the bigger problem for value it reached only 82.65%, so there is a lacking 17.35% that it must be addressed. Figure 14 illustrates the percentages of value and its indicators.

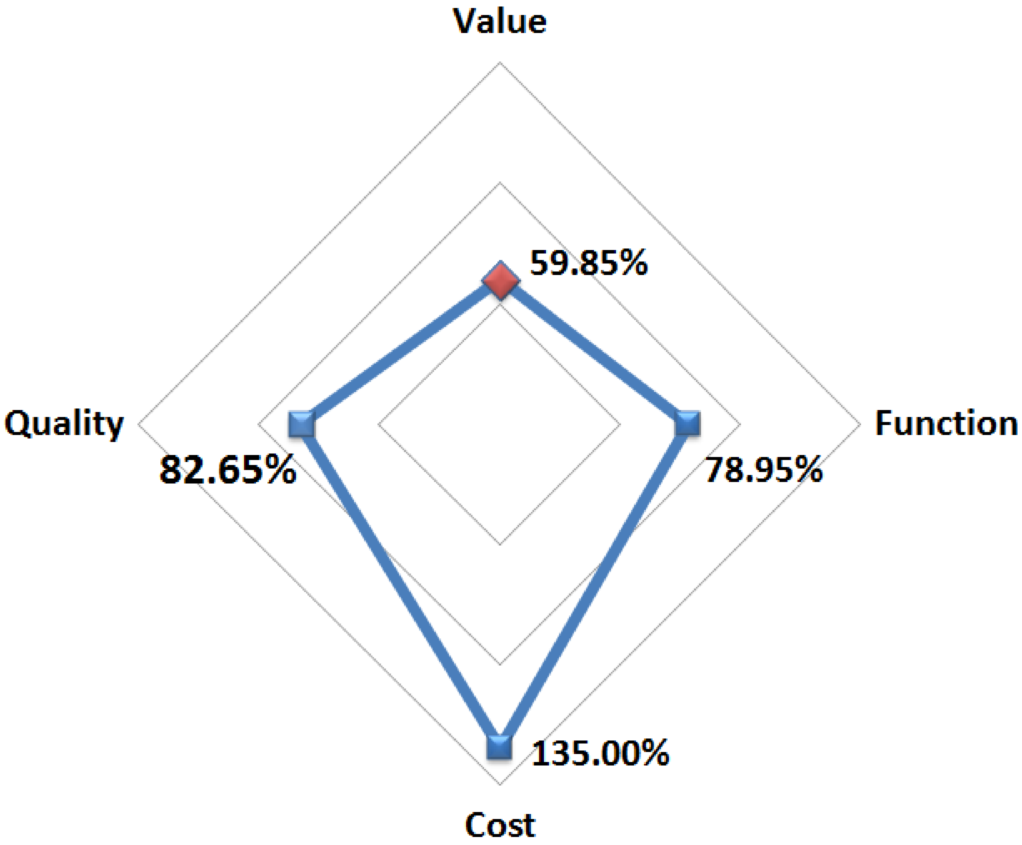


Figure 14: ISPC: Percentages of value and its indicators

4.1.11 ISPC: Limitations

The assurance of the value estimation for this case study totally depends on the quality of the documentation provided the interaction with the product manager, and the experience and perception of the evaluator.

Sufficient documentation was provided, but there was very little interaction with the product manager and the development team, which affected the study, a few interviews were made with one team member, but for more accuracy the study needs more feedback.

This is the first value estimation study that the evaluator executed which it is a limitation, taking into account the narrow perception for the execution, but valuable lessons and findings were done in the process, better planning and defined objectives together with the stakeholders should be added to future value estimations.

4.1.12 ISPC: Conclusions

This case study has proved to be very useful; it is the first execution of the reference model to estimate the software product value. Only a part of the system is on the value estimation but it gives a good perspective of the model application and it has proved its feasibility to apply it on real software projects.

Section 4.1.10 discusses the quantitative results of the value indicators, which are very interesting and offers a broad overview of the value indicators data to detect possible deviations of the software product development.

The results of the value estimation are very interesting for the development team; now with the provided data they are aware of the problems that they have on the *User Management Component*. The bigger problems are on function and cost, but also quality presents issues in the estimation, and it is where more details are provided in this case study. Based on the estimation results they should take it into account the data to modify the way they are working, in order to improve, so at the end they deliver a final product with the value they expect it should deliver to the customer.

After the application of the RESVEP to the UMC 1.0, minor changes have been performed to the Value Quality Indicators. Specially in the value quality indicator of Functional Suitability (FS), which in the case study is defined by indicators obtained by [11], but after an analysis it was defined that FS would be defined by the sub-indicators stated in the ISO 25010 [12].

4.2 Case Study 2: Product Q (PQ)

4.2.1 PQ: Introduction to RESVEP

A presentation was given to the product manager in order to explain the definition of the software product value, the steps of the RESVEP, the activities to perform the case study. Also, the documentation was required and the project manager participation. A session of questions and doubts was conducted, and the objectives of the case study were explained.

4.2.1.1 Objective of the PQ Case Study

The general objective of this case study on *Company A* was to apply the model to estimate software product value proposed in this thesis. The estimation process is performed using value indicators of the Product Q.

Also it was considered to obtain feedback from the industrial area of software development; this could measure the feasibility of the approach in terms of cost - benefits for software companies. Finally, getting feedback was established to complement and correct the measurements and equations of the value indicators.

Specific objectives. The specific objectives to achieve in this case study, which were validated at the end of the case study execution, are listed below:

1. To obtain quantitative data of the value indicators.
2. To obtain the value state of the software product.
3. To Estimate of the cost and/or effort that involves the application of the methodology, and how much it affects the cost of the software product to be evaluated.
4. To evaluate the usefulness of the value estimation reference model process for software companies in real projects.

4.2.2 PQ: Gather Knowledge About the Project

4.2.2.1 Company Background

Company A, is a software manufacturer and a provider of software consulting services, specializing in the design and construction of solutions for management of the portfolio and quality of applications, which help those responsible for information technology, based on the information collected and its management models, to make appropriate decisions for certification, maintenance, rationalization and modernization.

Company A offers products and services focused on improving the quality of software to other software development companies it is aware of the importance of evaluating their own software tools sold in the market. Particularly, some of the proprietary tools of *Company A* are oriented for this niche; it is assumed that there are similar tools in the highly competitive market, so it is a priority to *Company A* to develop their products with the highest quality standards.

To achieve this level of quality, *Company A* performs quality assessments to their own tools during the development cycle, being managed within the same company. Additionally, the company considers of great importance that a team outside itself evaluates their work as product developers. In the specific situation of this case study at *Company A*, this value estimation is focused on the value of products which include quality as a primary component of the concept of value, according to the reference model proposed in this thesis, which is based on the concept of value indicators.

4.2.2.2 Case Study Methodology

Resources Provided. The resources and documentation provided for this case study of *Product Q* is on the following list:

- Manual and list of functions and requirements: obtained with the release notes.
- High level architecture: scheme given by the product manager.

- Prioritization of quality attributes: defined in conjunction with the product manager.
- Function points obtained by a tool to calculate function points (function point modeler [21]).
- Cost: given by the product manager.
- Software units.

Activities and effort required. For the execution of the Product Q 1.0 case study for software product value estimation, it took 67.6 of total hours of work, (in Table 48 breaks down the effort to activities), about 5 hours per day, resulting in an approximate 13.5 workdays.

Table 48: Summary of activities and effort in hours for the value estimation at *Company A*

Activities	**Stakeholders**	**Hours required with the staff**	**Total effort in hours**
1. Introduction to the software product value estimation model methodology, based on indicators of value.	Evaluator, product manager, development team	1	1
2. Knowledge of the project, for which the model will be applied.	Product manager, evaluator	1	5
3. Delimitation of the products to be measured.	Product manager	1	5
4. Delimitation of the value indicators to be measured (quality attributes).	Product manager, evaluator	1	3
5. Application of the value indicator metrics to obtain Quality.	Evaluator, product manager	2	16.6
6. Verification of the implemented function (with the expected function to verify what percentage has been really developed).	Product manager, evaluator	1	10
7. Verification of the development cost of the products to measure.	Product manager, evaluator	1	7
8. Estimate the value of the software product in development, using the proposed approach.	Evaluator	-	10
9. Interpretation of results and value elements that comprise it, using simulation techniques.	Evaluator	-	10
Total of Hours		8	67.6

4.2.3 PQ: Delimitation of the Work Products to Estimate

The product manager together with the company team decided that the product to estimate would be Product Q (PQ). PQ is a software tool from *Company A*, which serves to define a quality model to evaluate software products. It is a web-type application, with visual windows and it is expected that the customer learn to use it in an intuitive manner, therefore usability is of great importance. PQ is in the final stages of development for the release 1.0 so it has not yet hit the market. PQ has been developed with the Agile methodology: *SCRUM* [24].

The release 1.0 of PQ (PQ 1.0) stems mainly of the analysis project of PQ done for *Product R 2.5.1* (proprietary tool of *Company A*, which is an integrated portal of quality that controls automatically each one of the different elements that are involved in the software life cycle and the susceptible software elements to be analyzed: source code, documentation, test scripts, data models, etc.).

PQ 1.0 includes the creation of a tool (or module) for managing quality models within *Product R* tool. The quality model entities that are managed in PQ 1.0 are:

- Indicators and metrics (single or aggregated).
- Rules.
- Forms.
- Primitive implemented rules (those from the connectors).
- Regulations and norms.
- Audits and checkpoints.

The development of *PQ 1.0* is determined by the functionality of *Product S* (a service of static code analysis in the cloud offered by *Company A*), and is conceived as a solution for managing and configuring *Product R* rule sets (*Company A* proprietary rules). Therefore, PQ functionality is limited by the functionality of *Product S*.

The restrictions of this first version are:

- Quality models are limited to pure rule sets configuration, i.e., rules with the overall confidence measure configured by default in the tool.
- The categories of rules are static, you cannot change or add, or edit, they correspond to the characteristics of the ISO 9126 [10] at first level.

4.2.3.1 Functions of PQ 1.0

The PQ 1.0 functions were obtained from the release note and the requirements, an identifier was assigned in conjunction with the Product Manager. In Table 49 such functionalities are listed and it indicates which ones were evaluated for this case study.

Table 49: PQ 1.0 functions

Identifier	**Description**	**Evaluated**
F001	Manage rules within the library.	✓
F002	Categorize into multilevel and multi-valued criteria.	✓
F003	Edit name, version, message, description, benefits, drawbacks, configuration parameters, examples, internal characteristics, priority and fix effort. • These fields can save and use in the library, leaving it personalized for each account. • The user can always restore the default distribution of the library rules.	✓
F004	Quick search of the name of the rules.	✓
F005	Feeding PQ rules library with PQ proprietary format defined for this purpose, which is the only thing that defines all required fields of the library and in multiple programming languages. • In this version is not performed impact analysis of new versions new quality models created.	✓
F006	Create versions of models.	✓
F007	Do not allow to change or edit categories in the ISO 9126 quality model.	✓
F008	Contemplate instantiation of current rules only for the model. • In each of the instances can only the next can be changed: priority, work effort and values of the characteristics of the rules.	✓
F009	Export quality model to the *Product R* format. • Currently only PQK exporting rule sets.	✓
F010	Deploy PQ on the server.	✗
F011	Advanced Search (search text fields).	✓
F012	Organize rules as a tree view.	✓
F013	Filter rules on the library.	✓
F014	Instantiate rules in the quality model.	✓

Continued on Next Page...

Table 49 – Continued

Identifier	Description	Evaluated
F015	Instantiate quality models.	✓
F016	Import of *Product R* quality models.	✗
F017	Filter rules in the quality model.	✓
F018	Organize rules in the quality model.	✓
F019	Edit: priority, repair effort and parameters of an instance of a rule in the model.	✓

4.2.4 PQ: Project Context Definition

According to the information already stated in Section 4.2.3 and Section 4.2.2, the project context definition is described in Table 50

Table 50: Project context of the PQ 1.0 case study

Process	Activity	Stakeholder	Work product
Software Construction [11]	Develop and document each software unit and database [11]	Implementer [11]	Software units defined by the design [11]

Figure 15 illustrates an instantiation of the reference model for the software product value estimation process (RESVEP), specific for the case study of PQ 1.0.

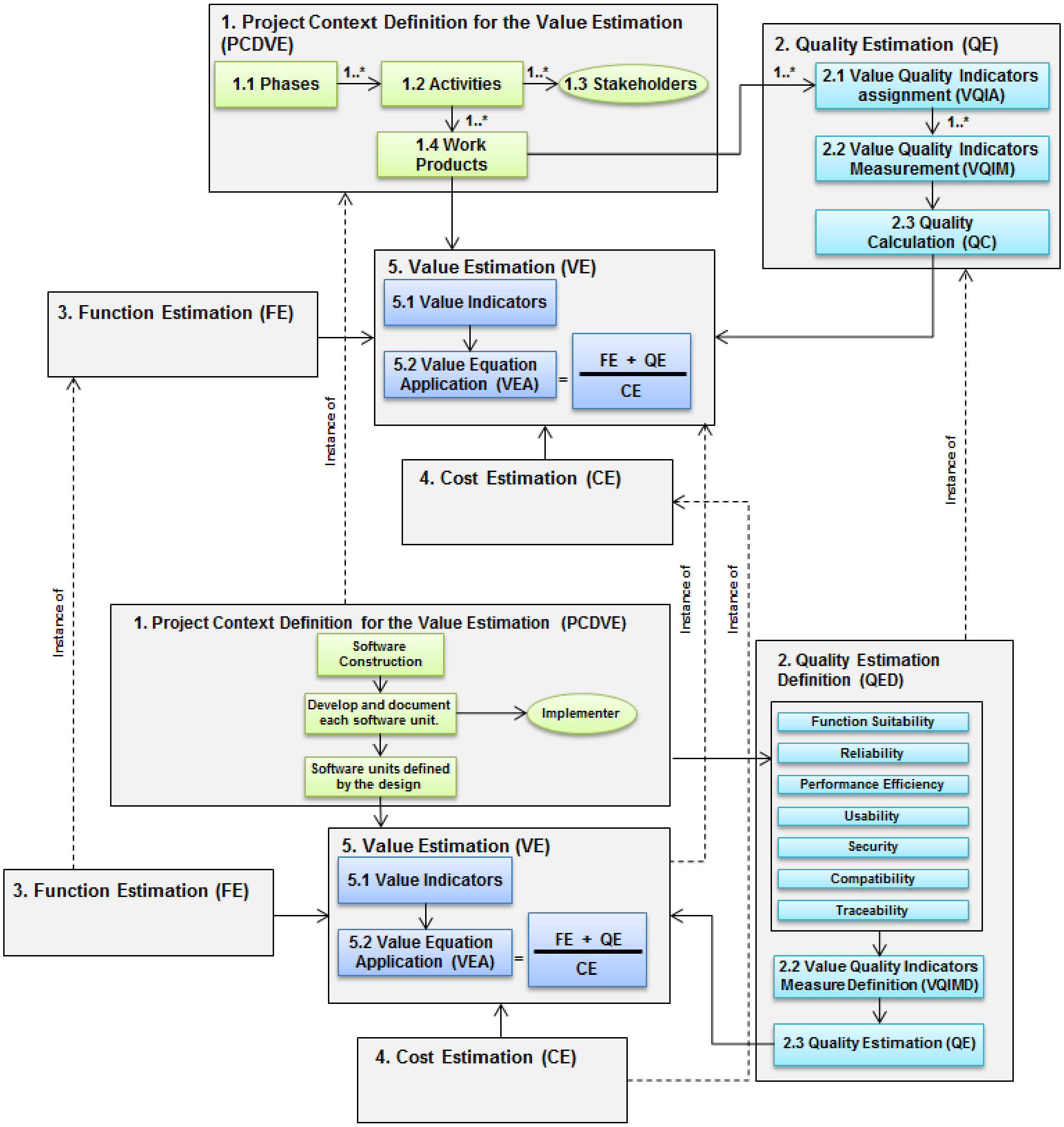

1. Project Context Definition for the Value Estimation (PCDVE)
1.1 Phases
1..*
1.2 Activities
1..*
1.3 Stakeholders
1..*
1.4 Work Products
2. Quality Estimation (QE)
1..*
2.1 Value Quality Indicators assignment (VQIA)
1..*
2.2 Value Quality Indicators Measurement (VQIM)
2.3 Quality Calculation (QC)
3. Function Estimation (FE)
5. Value Estimation (VE)
5.1 Value Indicators
5.2 Value Equation Application (VEA)
=
FE + QE
CE
4. Cost Estimation (CE)
Instance of
Instance of
Instance of
Instance of
Instance of
1. Project Context Definition for the Value Estimation (PCDVE)
Software Construction
Develop and document each software unit.
Implementer
Software units defined by the design
2. Quality Estimation Definition (QED)
Function Suitability
Reliability
Performance Efficiency
Usability
Security
Compatibility
Traceability
2.2 Value Quality Indicators Measure Definition (VQIMD)
2.3 Quality Estimation (QE)
3. Function Estimation (FE)
5. Value Estimation (VE)
5.1 Value Indicators
5.2 Value Equation Application (VEA)
=
FE + QE
CE
4. Cost Estimation (CE)

Figure 15: Instance of the reference model for software product value estimation process for the PQ 1.0 case study

4.2.5 PQ: Delimitation of the Value Quality Indicators

Value Quality indicators were assigned to PQ 1.0; this was done in consensus with the Product Manager. In Table 51 these indicators are listed, the expected measure (which PQ is expected to meet at the release 1.0) and the justification for the choice.

Table 51: Assignment of value quality indicators with measures expected to be delivered in the release 1.0 of PQ and justification of choice

Indicator	**Expected measure**	**Justification of choice**
U: Usability	0.9000	This product requires a maximum of usability, requires the user to learn to use it easily.
PE: Performance Efficiency	0.9000	The product will be mounted on a *Company A* server and users will access it via Web, therefore efficiency is needed for the case of having multiple users accessing the system simultaneously.
C: Compatibility	0.7000	PQ needs to coexist and be consistent in structure regardless of the Web browser the customers are using.
FS: Functional Suitability	0.7000	It is necessary a consistency between requirements and design with what was truly developed but, it may vary slightly because of the agile process that the team used.
S: Security	0.6000	Does not require a very high security, but a login is used to enter the system, and data is discriminated by accounts and users.
R: Reliability	0.5000	The level of this feature is medium, since there is a need of the data to be reliable, but is not high risk (money management, or health of persons).
X1: Traceability to the requirements	0.4000	This feature is low, by the type of development process.

4.2.6 PQ: Value Quality Indicators Measurement and Quality Calculation

4.2.6.1 Value Quality Indicators Measurement

The following describes the measurement of the value indicators of the PQ 1.0 case study.

Usability. The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. [12]

The measured sub-characteristics of usability are u1, u2, u3 and u4 and the equation of usability are in Table 52. The measurement procedures of u1, u2, u3 and u4 are in Tables 53, 54, 55 and 56.

Table 52: Sub-characteristics of usability evaluated for PQ 1.0

Identifier	**Sub-characteristics**	**Evaluation**
u1	Learnability	See Table 53
u2	Ease of use	See Table 54
u3	User error protection	See Table 55
u4	User interface aesthetics	See Table 56 and 57
Equation of usability: $U = \frac{u1+u2+u3+u4}{4} = 0.9083$		

Table 53: PQ 1.0: Learnability

Characteristic	**Data**
Description	The degree to which a product can be used by specified users to achieve specified learning goals with effectiveness, efficiency, safety and satisfaction in a specified context of use. [12]
Measurement and Equation	$$u1 = \frac{t1}{t2}$$ u1 = Learnability. t1 = Mean expected-time taken to learn a software unit correctly. t2 = Mean real-time taken to learn a software unit correctly. Interpretation of the result: $0 <= u1 <= N$ If equal to 1 or higher, the better. Measurement of the standard [10]. If more than one functionality of the software unit is evaluated, there will be a summation of t1 of each function, and this also applies to t2.
Measurement Method	The expected average time was specified by the product manager according to product requirements. The actual average time was measured with the aid of a chronometer.
Result	The mean expected-time to learn to use the software unit (15 functions) was $2,700$ seconds, and the real mean-time to learn to use the functions evaluated was $2,706$ seconds. Therefore: $$u1 = \frac{t1}{t2} = \frac{2700}{2706} = 0.9978$$
Limitations	The evaluator's skills and perception influence this measurement method by the nature of the sub-characteristic of learnability.
Problems	In the F006 functionality was very difficult to find the use, which was very hard to learn to use it, of 180 seconds expected it took 420 seconds to learn it.

Table 54: PQ 1.0: Ease of use

Characteristic	**Data**
Description	The degree to which users find the product easy to operate and control [12]
Measurement and Equation	$$u2 = \frac{t1}{t2}$$ u2 = Ease of use. t1 = Mean expected-time taken to use a software unit correctly. t2 = Mean real-time taken to use a software unit correctly. Interpretation of the result: $0 <= u2 <= N$ If equal to 1 or higher, the better. Mapped equation from the standard [10]. If more than one functionality of the software unit is evaluated, there will be a summation of t1 of each function, and this also applies to t2.
Measurement Method	The expected average time was specified by the product manager according to product requirements. The actual average time was measured with the aid of a chronometer.

Continued on Next Page...

Table 54 – Continued

Characteristic	Data
Result	The mean expected-time to use the software unit (15 functions) was 450 seconds and the mean real-time to use the functionalities was 308. Therefore: $$u2 = \frac{t1}{t2} = \frac{450}{308} = 1.4610$$
Limitations	The evaluator's skills and perception influence this measurement method by the nature of the sub-characteristic of ease of use.
Problems	The F005 and F013 functionalities got a mean real-time higher than the mean-expected time.

Table 55: PQ 1.0: User error protection

Characteristic	Data
Description	The degree to which the system protects users against making errors [12]
Measurement and Equation	$$u3 = \frac{A}{B}$$ u3 = User error protection. A = Number of user-interface units that protects users against making errors. B = Number of user-interface units evaluated. Interpretation of the result: 0 <= $u3$ <= 1 The closer to 1, the better. Measurement of the standard [10].
Measurement Method	The user-interface units that were evaluated had user input, with the following criteria: 1. In text boxes: limit of accepted characters and special characters accepted. 2. In buttons: acceptance of press button when they should be disabled.
Result	13 user interfaces were evaluated. And 7 user interfaces are protected against user errors, therefore: $$u3 = \frac{A}{B} = \frac{7}{13} = 0.5385$$
Limitations	In the release note it is not specified what kind of characters must support each functionality on which the user enters information, therefore the criteria taken into account was inferred according to the type of user input.
Problems	The user interface of the functionalities of filters (F013 and F017) and the organizer as tree view (F012) throw errors when used, there is no control in the text boxes on the type of characters and the amount of letters to enter, this may be planned to be developed in a future release, it is not clear in the documentation.

Table 56: PQ 1.0: User interface aesthetics

Characteristic	Data
Description	The degree to which the user interface enables pleasing and satisfying interaction for the user. [12]
Measurement and Equation	$$u4 = \frac{\frac{A_1+A_2+A_3+...+A_N}{B}}{10}$$ u4 = User interface aesthetics. A$_i$ = Rating assigned to the user interface (0 to 10) according to user satisfaction criteria. B = Number of user-interface units evaluated. Interpretation of the result: 0 <= $u4$ <= 1 The closer to 1, the better. Mapped equation from the standard [10].

Continued on Next Page...

Table 56 – Continued

Characteristic	**Data**
Measurement Method	User interfaces were evaluated, those related to the functionalities defined according to the following criteria taken into account for qualification: • Intuitive button design. • The use of the keyboard for fast functionality. • Color combination. • Number of clicks to execute the action.
Result	The results of individual ratings are displayed in Table 57, the sum of scores for the interfaces was 89, the total evaluated interfaces was 14. Therefore: $u4 = \frac{\frac{A_1+A_2+A_3+...+A_{14}}{B}}{10}$ $u4 = \frac{\frac{89}{14}}{10} = \frac{6.3571}{10} = 0.6357$
Limitations	This assessment is subject to the perception of the evaluator, several evaluators might repeat this exercise to obtain an average estimation.
Problems	Below are listed the problems identified in various user interfaces: • In the checkboxes when selecting a single sub-tree item, it appears to be selected as if sub-tree items of that item were selected. • There is a missing button to instantiate rules directly, for now the user can only instantiate a metric by drag and drop. • In the window of the rules instantiated at the quality model, the metrics that are added to the quality model on a visual table, and it does not allow to organize the items that are there (for example: by name or total) this only happens on the side of rules libraries (of the visual table). • To remove metrics from the quality model, the user must open another window and give too many clicks (4).

Table 57: Table of functionalities, user interface and their qualification of PQ 1.0

Function	**User interface**	**Qualification (A)**
All F00N	Main window.	7
F001, F005	Window library rules.	4
F018	Window quality model.	4
F012	Tree view of library rules.	8
F004, F011	Textbox search (library and model).	9
F001, F002, F018	Auxiliary window search (library and model).	6
F013, F017	Auxiliary window filter (library and model).	6
F003, F008	Window edit rules.	8
F002	Window classification criteria.	4
F015	Window setup.	5
F015	Window of versions.	3
F019	Window for quick edit of model rules.	8
F001	Window of rule information.	10
F014	Instantiate a rule.	7
	$(u4 = Sumofscores/Totalofinterfaces)/10$	0.6357

Summary of Usability Measurement. In Table 58, there is a breakdown of the usability evaluation, separated by sub-characteristics and functionalities evaluated. Figure 16 displays the results of usability and its sub-characteristics.

Table 58: Data of the usability evaluation of PQ 1.0, by functions and sub-characteristics

Function ID	**u1: Learnability**		**u2: Ease of use**		**u3: User error protection**		**u4: User interface aesthetics**
	t1	**t2**	**t1**	**t2**	**A**	**B**	-
F001	180	120	30	26	-		
F002	180	139	30	14	✓		
F003	180	84	30	25	✓		
F004	180	60	30	6	✗		
F005	180	300	30	35	-		
F006	180	420	30	12	✗		
F007	-	-	-	-	-		
F008	180	166	30	27	✓		
F009	180	80	30	27	✓		
F011	180	106	30	20	✗		
F012	180	66	30	7	✓		
F013	180	296	30	33	✗		
F014	180	144	30	13	✓		
F015	-	-	-	-	-		
F017	180	337	30	29	✗		
F018	180	199	30	18	✗		
F019	180	189	30	16	✓		
Results	Sum $t1$ = 2700	Sum $t2$ = 2706	Sum $t1$ = 450	Sum $t2$ = 308	$A = 7$	$B = 13$	See estimation in Table: 57.
Equation application	$u1 = \frac{t1}{t2} =$ 0.9978		$u2 = \frac{t1}{t2} =$ 1.4610		$u3 = \frac{A}{B} =$ 0.5385		$u4 = 0.6357$
U: Usability	$U = \frac{u1+u2+u3+u4}{4}$ $U = \frac{0.9978+1.4610+0.5385+0.6357}{4}$ $U = 0.9082$						

Notes:
Learnability and ease of use:
The measurements described in t1 (expected) and t2 (estimated) is time in seconds.
User error protection:
"✓" the function is protected against user errors.
"✗" the function is not protected against user errors.
"-" means that the function does not apply for this indicator or it could not be evaluated.

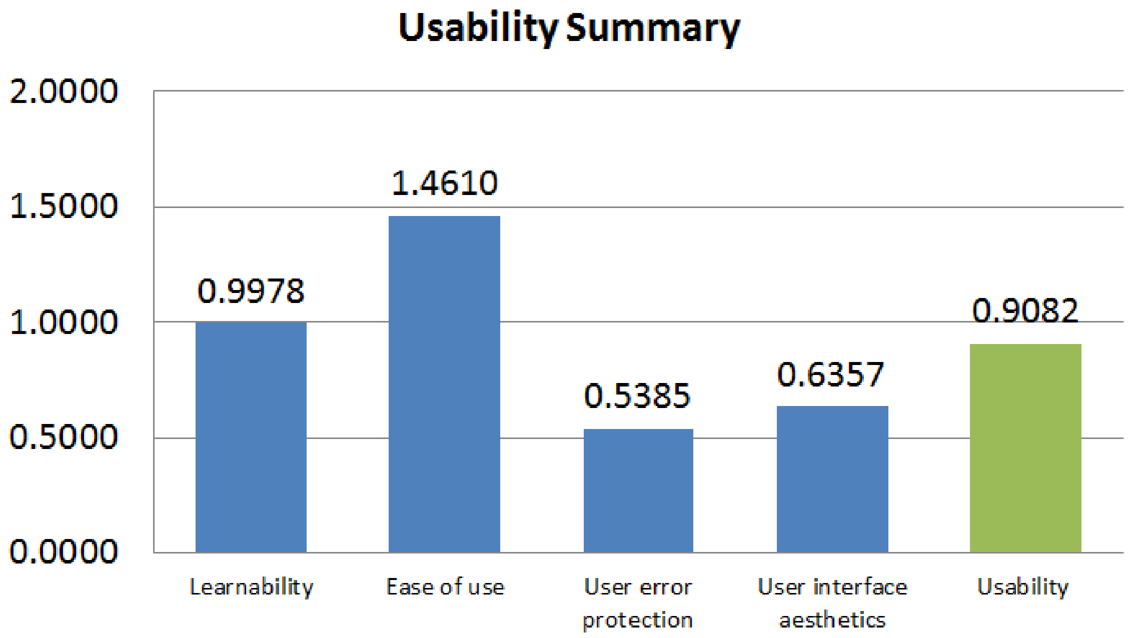


Figure 16: PQ 1.0: Measurement summary of usability and its sub-characteristics

Performance efficiency. The performance relative to the amount of resources used under stated conditions. [12]

The measured sub-characteristic of performance efficiency is pe1 and the equation of performance efficiency are in Table 59. The measurement procedures of pe1 is in Table 60.

Table 59: Sub-characteristics of performance efficiency evaluated for PQ 1.0

Identifier	**Sub-characteristics**	**Evaluation**
pe1	Time behavior	See Table 60
Equation of performance efficiency: $PE = \frac{pe1}{1} = 0.4706$		

Table 60: PQ 1.0: Time behavior

Characteristic	**Data**
Description	The response and processing times and throughput rates of a system when performing its function, under stated conditions in relation to an established benchmark. [12]

Continued on Next Page...

Table 60 – Continued

Characteristic	Data
Measurement and Equation	$pe1 = \frac{t1}{t2}$ pe1 = Time behavior. t1 = Time of response expected. t2 = Time of real response. Interpretation of the result: $0 <= pe1 <= N$ If equal to 1 or higher, the better. Measurement of the standard [10]. If more than one functionality of the software unit is evaluated, there will be a summation of t1 of each function, and this also applies to t2.
Measurement Method	The expected response time was specified by the product manager according to product requirements. The actual response time was measured with the aid of a chronometer.
Result	The sum of the expected response time for the functionalities tested (12) was 24 seconds, and the sum of the actual response time was 51. Therefore: $pe1 = \frac{t1}{t2} = \frac{24}{51} = 0.4706$
Limitations	Another sub-indicator of efficiency: resource utilization could not be evaluated due to complications to apply the measurements to the server memory that is hosting the system.
Problems	There are 7 functions that exceed the time behavior expected, it can be seen in detail in Table 61.

Summary of Performance Efficiency Measurement. In Table 61, there is a breakdown of the performance efficiency evaluation, separated by sub-characteristics and functionalities evaluated. Figure 17 displays the results of performance efficiency and its sub-characteristics.

Table 61: Data of the performance efficiency evaluation of PQ 1.0, by functions and sub-characteristics

Function ID	**pe1: Time behavior**	
	t1	**t2**
F001	-	-
F002	2	8
F003	-	-
F004	2	6
F005	2	6
F006	2	1
F007	-	-
F008	-	-
F009	-	-
F011	2	9
F012	2	6
F013	2	9
F014	2	2
F015	2	1
F017	2	1
F018	2	1
F019	2	1
Results	Sum $t1 = 24$	Sum $t2 = 51$

Continued on Next Page...

Table 61 – Continued

Function ID	**pe1: Time behavior**
Equation application	$pe1 = \frac{t1}{t2}$ $pe1 = \frac{24}{51}$ $pe1 = 0.4706$
PE: Performance Efficiency	$PE = \frac{pe1}{1}$ $PE = \frac{0.4705}{1}$ $PE = 0.4706$

Notes:
Time behavior:
The results described in t1 (expected) and t2 (estimated) is time in seconds.

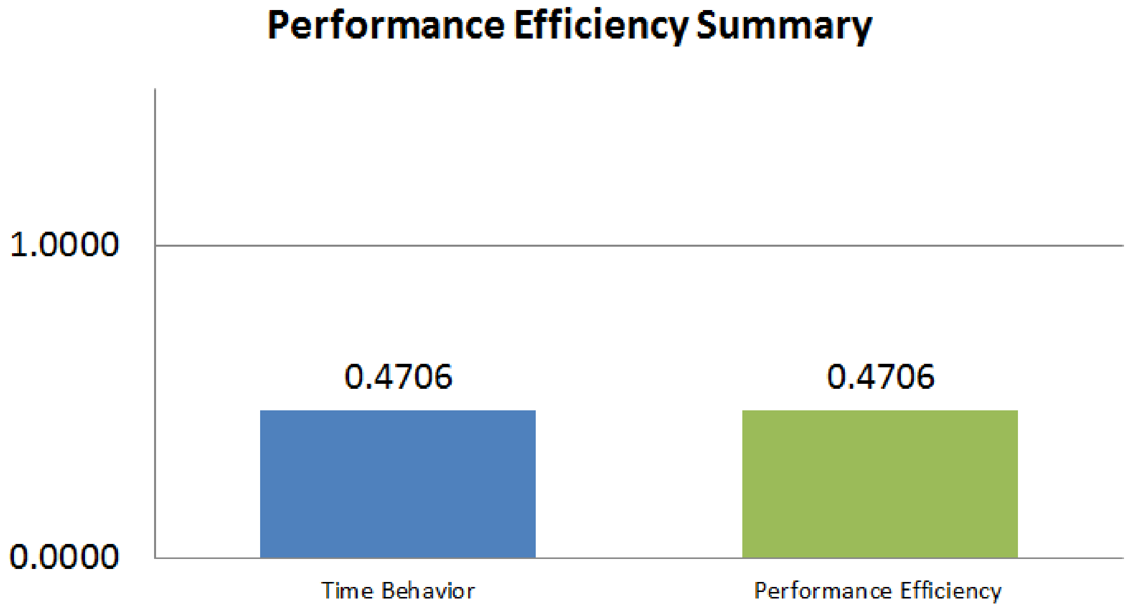


Figure 17: PQ 1.0: Measurement summary of performance efficiency and its sub-characteristics

Compatibility. The degree to which two or more systems or components can exchange information and/or perform their required functions while sharing the same hardware or software environment. [12]

The measured sub-characteristics of compatibility are c1 and c2 and the equation of compatibility are in Table 62. The measurement procedures of c1 and c2 are in Tables 63 and 64.

Table 62: Sub-characteristics of compatibility evaluated for PQ 1.0

Identifier	**Sub-characteristics**	**Evaluation**
c1	Co-existence	See Table 63
c2	Interoperability	See Table 64
Equation of compatibility: $C = \frac{c1+c2}{2} = 0.8750$		

Table 63: PQ 1.0: Co-existence

Characteristic	**Data**
Description	The degree to which the product can co-exist with other independent software in a common environment sharing common resources without any detrimental impacts. [12]
Measurement and Equation	$$c1 = 1 - \frac{A}{B}$$ c1 = Co-existence. A = Number of components that cannot co-exist with other independent software in a common environment sharing common resources without any detrimental impacts. B = Number of components tested. Interpretation of the result: $0 <= c1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	In accordance with the product manager coexistence was measured based on the browsers on which the product was tested, it is of interest to the product manager that PQ 1.0 components are evaluated in this area, because the product is used on the Web and is accessed via browsers. The evaluated *Browsers* were: • Internet explorer 8.076 • Mozilla firefox 13.0.1 • Safari 5.7.1 • Google chrome 20.0.1132.47
Result	Of the four browsers on which the tests were performed, errors were detected only in one of them: internet explorer, therefore it is not co-existent. The measurements for the components and the four browsers resulted as follows: $$c1 = 1 - \frac{\frac{A_{ie}}{B_{ie}} + \frac{A_{mf}}{B_{mf}} + \frac{A_s}{B_s} + \frac{A_{gc}}{B_{gc}}}{4}$$ $$c1 = 1 - \frac{\frac{17}{17} + \frac{0}{17} + \frac{0}{17} + \frac{0}{17}}{4} = 1 - 0.25 = 0.75$$ Note: • ie: Internet explorer • mf: Mozilla firefox • s: safari • gc: Google chrome
Limitations	The estimation of this indicator is much delimited to this particular case, because the product is web software accessed through browsers.

Continued on Next Page...

Table 63 – Continued

Characteristic	Data
Problems	The following problems were found in particular for the Internet Explorer browser: • The search buttons do not appear. • The search textbox is unformatted. • The tree view is not working until the second attempt to use it after loading the page.

Table 64: PQ 1.0: Interoperability

Characteristic	Data
Description	The degree to which two or more systems or components can exchange information and use the information that has been exchanged. [12]
Measurement and Equation	$$c2 = \frac{A}{B}$$ c2 = Interoperability. A = Number of functions that can exchange information and use the information that has been exchanged. B = Number of functions evaluated. Interpretation of the result: $0 <= c2 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	In accordance with the product manager; interoperability was measured only with the following functions which exchange information: • F003 • F008 • F009 • F019 These functions interchange information via XML files.
Result	Of the four functions evaluated, anomalies were not detected in the exchange of information. Therefore: $$c2 = \frac{A}{B} = \frac{4}{4} = 1$$
Limitations	Interoperability was assessed only among the company products; interoperability was not tested with products from another company.
Problems	No abnormalities were detected in this evaluation, but it must be emphasized that the evaluation was limited with the company own products.

Summary of Compatibility Measurement. In Table 65, there is a breakdown of the compatibility evaluation, separated by sub-characteristics and functionalities evaluated. Figure 18 displays the results of compatibility and its sub-characteristics.

Table 65: Data of the compatibility evaluation of PQ 1.0, by functions and subcharacteristics

Function ID	**c1: Coexistence**					**c2: Interoperability**	
	$\mathbf{A}_{ie}$	$\mathbf{A}_{mf}$	$\mathbf{A}_{s}$	$\mathbf{A}_{gc}$	**B**	**A**	**B**
F001	✗	✓	✓	✓		-	
F002	✗	✓	✓	✓		-	
F003	✗	✓	✓	✓		✓	
F004	✗	✓	✓	✓		-	
F005	✗	✓	✓	✓		-	
F006	✗	✓	✓	✓		-	
F007	✗	✓	✓	✓		-	
F008	✗	✓	✓	✓		✓	
F009	✗	✓	✓	✓		✓	
F011	✗	✓	✓	✓		-	
F012	✗	✓	✓	✓		-	
F013	✗	✓	✓	✓		-	
F014	✗	✓	✓	✓		-	
F015	✗	✓	✓	✓		-	
F017	✗	✓	✓	✓		-	
F018	✗	✓	✓	✓		-	
F019	✗	✓	✓	✓		✓	
Results	$A_{ie} = 17$	$A_{mf} = 0$	$A_s = 0$	$A_{gc} = 0$	$B = 17$	$A = 4$	$B = 4$
Equation application	$c1 = 1 - \frac{\frac{A_{ie}}{B} + \frac{A_{mf}}{B} + \frac{A_s}{B} + \frac{A_{gc}}{B}}{4}$ $c1 = 1 - \frac{\frac{17}{17} + \frac{0}{17} + \frac{0}{17} + \frac{0}{17}}{4}$ $c1 = 1 - 0.25 = 0.75$					$c2 = \frac{A}{B} = \frac{4}{4} = 1$	
C: Compatibility	$C = \frac{c1+c2}{2}$ $C = \frac{0.75+1}{2}$ $C = 0.875$						

Notes:

For coexistence:

- A_{ie}: Internet explorer
- A_{mf}: Mozilla firefox
- A_{s}: Safari
- A_{gc}: Google chrome

Coexistence and interoperability:
"✓" if the indicator approved the criteria of their respective measurement.
"✗" if the indicator do not approved the criteria of their respective measurement.
"-" means it does not apply or could not be measured.

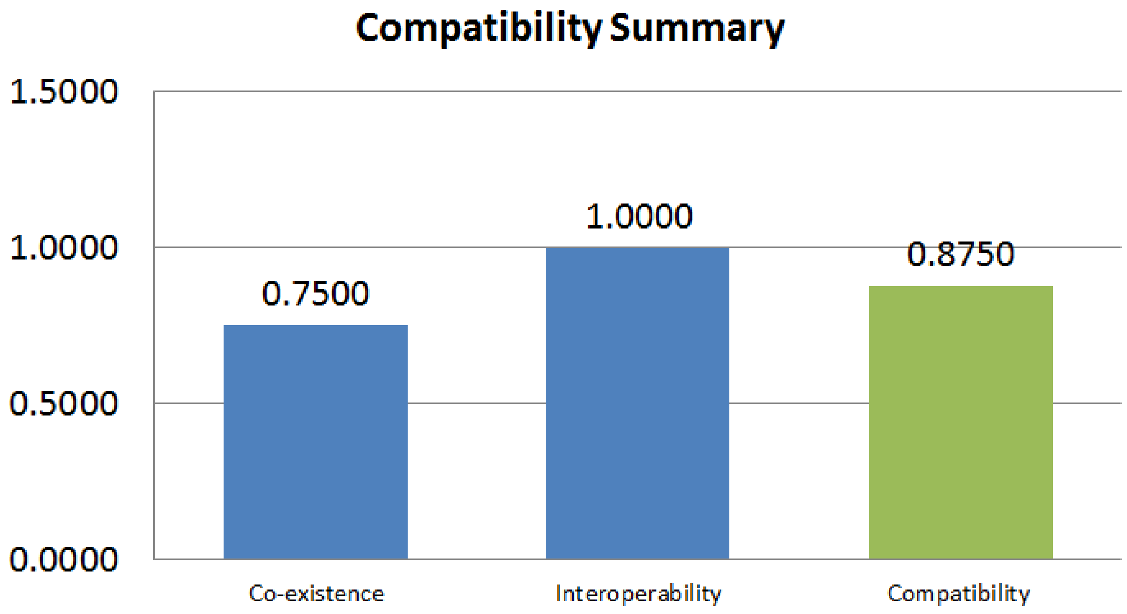


Figure 18: PQ 1.0: Measurement summary of compatibility and its sub-characteristics

Functional suitability. The degree to which the product provides functions that meet stated and implied needs when the product is used under specified conditions. [12]

The measured sub-characteristics of functional suitability are fs1 and fs2 and the equation of functional suitability are in Table 66. The measurement procedures fs1 and fs2 are in Tables 67 and 68.

Table 66: Sub-characteristics of functional suitability evaluated for PQ 1.0

Identifier	**Sub-characteristics**	**Evaluation**
fs1	Functional appropriateness	See Table 67
fs2	Accuracy	See Table 68
Equation of functional suitability: $FS = \frac{fs1+fs2}{2} = 0.8236$		

Table 67: PQ 1.0: Functional appropriateness

Characteristic	**Data**
Description	The degree to which the set of functions is suitable for specified tasks and user objectives. [12]
Measurement and Equation	$$fs1 = 1 - \frac{A}{B}$$ fs1 = Functional appropriateness. A = Number of missing functions or with errors detected in evaluation. B = Number of functions described in requirements. Interpretation of the result: $0 <= fs1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	Each function was checked to confirm its existence and verify if the functionality performed correctly or with errors.
Result	Two problems were found while verifying the functions, of the total that were tested (17). Therefore: $$fs1 = 1 - \frac{A}{B} = 1 - \frac{2}{17} = 1 - 0.1176 = 0.8824$$
Limitations	There were no limitations for the evaluation of this indicator.
Problems	F0013 and F0017 functionalities have errors when using filters (rules and instantiated rules), and the buttons are disabled, the user has to close and open the window again to use the buttons again.

Table 68: PQ 1.0: Accuracy

Characteristic	**Data**
Description	The degree to which the product provides the correct results with the needed degree of precision. [12]
Measurement and Equation	$$fs2 = 1 - \frac{A}{B}$$ fs2 = Accuracy. A = Number of inconsistent functions detected in evaluation. B = Number of functions described in requirements and design. Interpretation of the result: $0 <= fs2 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	Verification was made for each function to verify that they execute what was specified in the requirements.
Result	There were 4 inconsistencies found of 17 functionalities evaluated. Therefore: $$fs2 = 1 - \frac{A}{B} = 1 - \frac{4}{17} = 1 - 0.2353 = 0.7647$$
Limitations	There were no limitations for the evaluation of this indicator.
Problems	The following problems were found: • F012: When trying to display the library rules in a tree view at the first time it appears blank. • F013: The window of library rules filter have errors in their filter selection buttons, and sometimes do not do what is expected, for example when clicking the button to select all or select none, it does not execute any action. • F017: The filter windows of the instantiated rules in the quality model, has the same issues that F003. • F018: It does not allow organizing rules by its name, type or priority.

Summary of Functional Suitability Measurement. In Table 69, there is a breakdown of functional suitability evaluation, separated by sub-characteristics and functionalities evaluated. Figure 19 displays the results of functional suitability and its sub-characteristics.

Table 69: Data of the functional suitability evaluation of PQ 1.0, by functions and sub-characteristics

Function ID	**fs1: Functional appropriateness**		**fs2: Accuracy**	
	A	**B**	**A**	**B**
F001	✓		✓	
F002	✓		✓	
F003	✓		✓	
F004	✓		✓	
F005	✓		✓	
F006	✓		✓	
F007	✓		✓	
F008	✓		✓	
F009	✓		✓	
F011	✓		✓	
F012	✓		✗	
F013	✗		✗	
F014	✓		✓	
F015	✓		✓	
F017	✗		✗	
F018	✓		✗	
F019	✓		✓	
Results	Sum $A = 2$	Sum $B = 17$	Sum $A = 4$	Sum $B = 17$
	$fs1 = 1 - \frac{A}{B}$ $fs1 = 1 - \frac{2}{17}$ $fs1 = 0.8824$		$fs2 = 1 - \frac{A}{B}$ $fs2 = 1 - \frac{4}{17}$ $fs2 = 0.7647$	
FS: Functional Suitability	$FS = \frac{fs1+fs2}{2}$ $FS = \frac{0.8824+0.7647}{2}$ $FS = 0.8235$			

Notes:
Functional appropriateness:
"✓" functions where there are not problems.
"✗" functions where nonexistent or with errors.
Accuracy:
"✓" functions with no inconsistencies.
"✗" functions with inconsistencies.
"-" functions that could not be evaluated for that indicator.

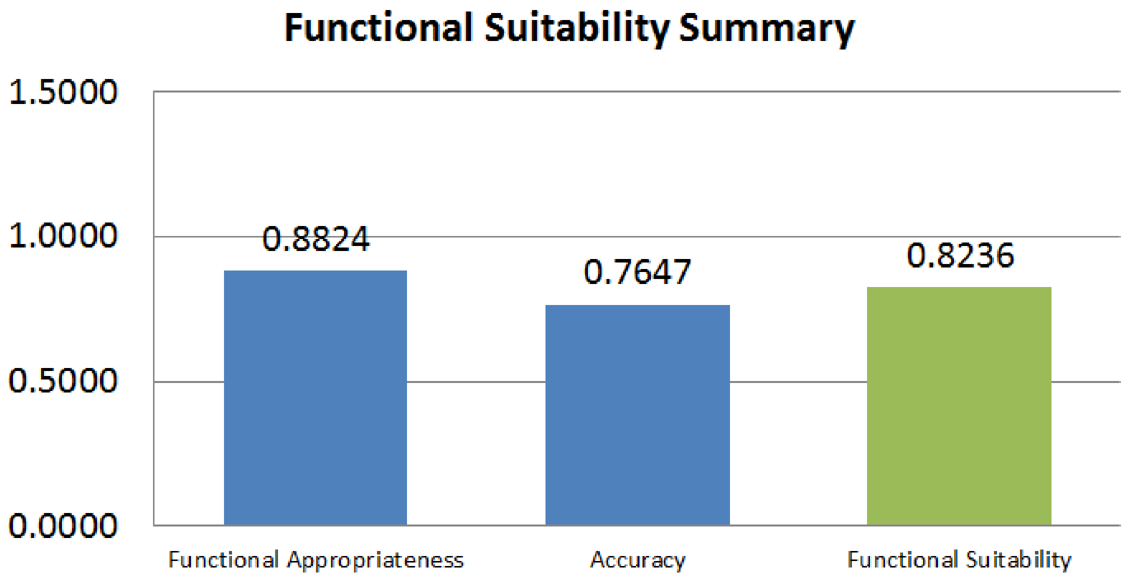


Figure 19: PQ 1.0: Measurement summary of functional suitability and its sub-characteristics

Security. The degree of protection of information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them. [12]

The measured sub-characteristics of security are s1, s2 and s3 and the equation of security are in Table 70. The measurement procedures of s1, s2 and s3 are in Tables 71, 72 and 73.

Table 70: Sub-characteristics of security evaluated for PQ 1.0

Identifier	**Sub-characteristics**	**Evaluation**
s1	Non-repudiation	See Table 71
s2	Accountability	See Table 72
s3	Authenticity	See Table 73
Equation of security: $S = \frac{s1+s2+s3}{3} = 0.6667$		

Table 71: PQ 1.0: Non-repudiation

Characteristic	Data
Description	The degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later. [12]
Measurement and Equation	$$s1 = \frac{A}{B}$$ s1 = Non-repudiation. A = Number of functions that have a log file of their events so they can be proven to have taken place. B = Total of functions that need a log file. Interpretation of the result: 0 <= $s3$ <= 1 The closer to 1, the better. Measurement of the standard [10].
Measurement Method	Log files were checked in search of information of the actions that are stored in the system.
Result	In the log files it was found that events recorded functionalities that manage data (8) to be held in the system. Therefore: $$s1 = \frac{A}{B} = \frac{8}{8} = 1$$
Limitations	There were no limitations to check this indicator; the log file was given by the product manager for review.
Problems	There were no problems for this indicator, although the format of the log files can be improved to facilitate the clarity of reading.

Table 72: PQ 1.0: Accountability

Characteristic	Data
Description	The degree to which the actions of an entity can be traced uniquely to the entity. [12]
Measurement and Equation	$$s2 = \frac{A}{B}$$ s2 = Accountability. A = Number of functions that their actions can be traced uniquely to them in the log file. B = Total number of functions that need a log file to trace actions. Interpretation of the result: 0 <= $s4$ <= 1 The closer to 1, the better. Measurement of the standard [10].
Measurement Method	Log files were verified to search the entity that realized an action.
Result	In the log files it was found that the events of the functions that manage data (8) and the entity that made such event are saved. Therefore: $$s2 = \frac{A}{B} = \frac{8}{8} = 1$$
Limitations	There were no limitations to check this indicator; the log file was given by the product manager for review.
Problems	Problems were not found for this indicator, although the format of the log files can be improved to facilitate clarity of reading.

Table 73: PQ 1.0: Authenticity

Characteristic	Data
Description	The degree to which the identity of a subject or resource can be proved to be the one claimed. [12]

Continued on Next Page...

Table 73 – Continued

Characteristic	**Data**
Measurement and Equation	$$s3 = \frac{A}{B}$$ s3 = Authenticity. A = Resource or subject Identification (Real Percentage of veracity for assuring this identification). B = Resource or subject Identification (Expected Percentage of veracity for assuring this identification). Interpretation of the result: $0 <= s5 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	Log files were verified to search the author of the realized actions.
Result	In the log files the author of actions realized was not founded, a 100% of veracity to assure this identification was expected, but after the evaluation the conclusion is that the veracity has 0% of real percentage. Therefore: $$s3 = \frac{A}{B} = \frac{0}{100} = 0$$
Limitations	There were no limitations to check this indicator; the log file was given by the product manager for review.
Problems	The author of the recorded actions in the log file was not found.

Summary of Security Measurement. In Table 74, there is a breakdown of security evaluation, separated by sub-characteristics and functionalities evaluated. Figure 20 displays the results of security and its sub-characteristics.

Table 74: Data of the security evaluation of PQ 1.0, by functions and sub-characteristics

Function ID	**s3: Non-repudiation**		**s4: Accountability**		**s5: Authenticity**	
	A	**B**	**A**	**B**	**A**	**B**
F001	-		-		-	
F002	-		-		-	
F003	✓		✓		✗	
F004	-		-		-	
F005	✓		✓		✗	
F006	✓		✓		✗	
F007	-		-		-	
F008	✓		✓		✗	
F009	✓		✓		✗	
F011	-		-		-	
F012	-		-		-	
F013	-		-		-	
F014	✓		✓		✗	
F015	✓		✓		✗	
F017	-		-		-	
F018	-		-		-	
F019	✓		✓		✗	
Results	$A = 8$	$B = 8$	$A = 8$	$B = 8$	$A = 0$	$B = 100$

Continued on Next Page...

Table 74 – Continued

Function ID	**s3: Non-repudiation**	**s4: Accountability**	**s5: Authenticity**
Equation application	$s3 = \frac{A}{B}$ $s3 = \frac{8}{8}$ $s3 = 1$	$s4 = \frac{A}{B}$ $s4 = \frac{8}{8}$ $s4 = 1$	$s5 = \frac{A}{B}$ $s5 = \frac{0}{100}$ $s5 = 0$
S: Security	$S = \frac{s3+s4+s5}{3}$ $S = \frac{1+1+0}{3}$ $S = 0.6667$		

Notes:

Non-repudiation:
"✓" functions that accomplish the non-repudiation measure.
"✗" functions that does not accomplish the non-repudiation measure.
"-" functions where this indicator is not applied.
Accountability:
"✓" functions that accomplish the accountability measure.
"✗" functions that does not accomplish the accountability measure.
"-" functions where this indicator is not applied.

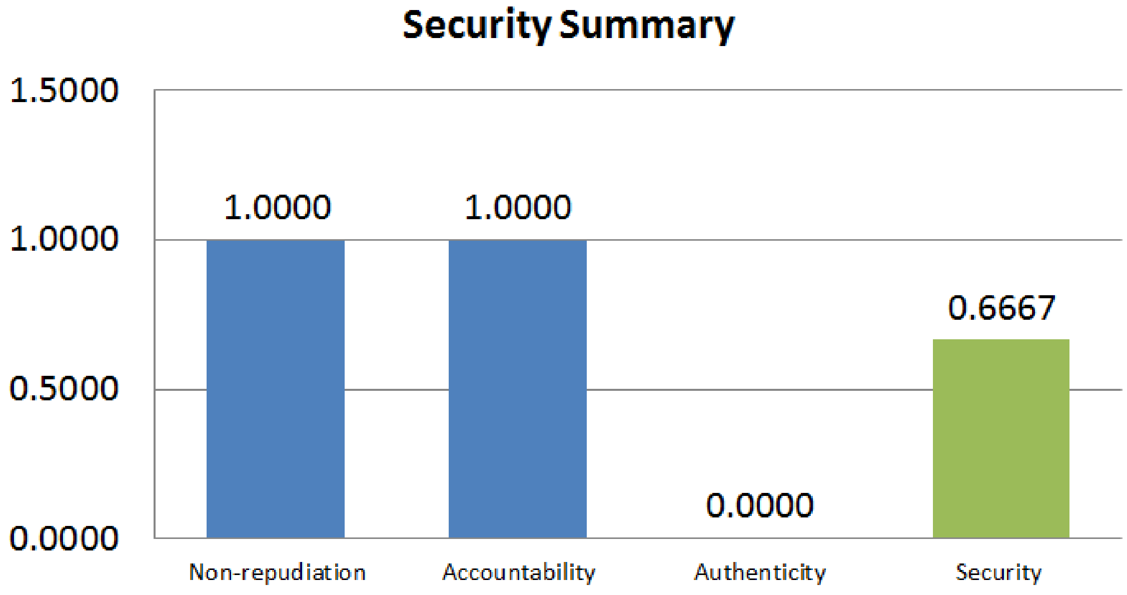


Figure 20: PQ 1.0: Measurement summary of security and its sub-characteristics

Reliability. The degree to which a system or component performs specified functions under specified conditions for a specified period of time. [12]

The measured sub-characteristics of reliability are r1, r2, r3 and r4 and the equation of reliability are in Table 75. The measurement procedures of r1, r2, r3 and r4 are in Tables 76, 77, 78 and 79.

Table 75: Sub-characteristics of reliability evaluated for PQ 1.0

Identifier	**Sub-characteristics**	**Evaluation**
r1	Maturity	See Table 76
r2	Availability	See Table 77
r3	Fault tolerance	See Table 78
r4	Recoverability	See Table 79
Equation of reliability: $R = \frac{r1+r2+r3+r4}{4} = 0.2500$		

Table 76: PQ 1.0: Maturity

Characteristic	**Data**
Description	The degree to which a system meets needs for reliability under normal operation. [12]
Measurement and Equation	$$r1 = \frac{A}{B}$$ r1 = Maturity. A = Desired system failures (reliability needs) in T. B = Current system failures in T. T = Time period (Time of the maturity test). Interpretation of the result: $0 <= r1 <= 1$ The closer to 1, the better. Measurement of the standard [10]. Special cases when a variable is equal to 0: 1. If $A = 0$ and $B = 0$, then $r1 = 1$. 2. If $A = 0$ and $B > 0$, then $r1 = 0$. 3. If $A > 0$ and $B = 0$, then $r1 = 1$.
Measurement Method	The system was tested during 273 minutes (4 hours, 33 minutes), the expected failures according to the needs of the system was 0, but there were 2 test failures.
Result	From the measurement method the following data was obtained: $A = 0$, $B = 2$ and $T = 273$ minutes, then the equation is applied: $$r1 = \frac{A}{B} = \frac{0}{2} = 0$$
Limitations	There were no limitations for this indicator.
Problems	The system had 2 failures with a "message error 500", and the site had to be restored to access the application again.

Table 77: PQ 1.0: Availability

Characteristic	**Data**
Description	The degree to which a system or component is operational and accessible when required for use. [12]

Continued on Next Page...

Table 77 – Continued

Characteristic	**Data**
Measurement and Equation	$$r2 = \frac{t1}{t2}$$ r2 = Availability. t1 = Time of the component being in an "up state" (operational and accessible when required). t2 = Time period (Time of the availability test). Interpretation of the result: 0 <= $r1$ <= 1 The closer to 1, the better. Measurement from the standard: [10].
Measurement Method	The system was tested during 273 minutes (4 hours, 33 minutes), time is counted when the application was available despite errors in the system operation.
Result	The measurement method gives the next information: $t1$ = 273 minutes, $t2$ = 273 minutes, with this data the equation is applied: $$r2 = \frac{t1}{t2} = \frac{273}{273} = 1$$
Limitations	There were no limitations to evaluate this indicator.
Problems	Despite the failures (two failures) after updating the website the system was normalized and continued to run smoothly.

Table 78: PQ 1.0: Fault tolerance

Characteristic	**Data**
Description	The degree to which a system or component operates as intended despite the presence of hardware or software faults. [12]
Measurement and Equation	$$r3 = \frac{A}{B}$$ r3 = Fault tolerance. A = Number of functions that operate as intended despite the presence of hardware or software faults. B = Number of functions evaluated. Interpretation of the result: 0 <= $r1$ <= 1 The closer to 1, the better. Measurement of the standard: [10].
Measurement Method	Documentation was analyzed, architecture, and it was discussed with the product manager to verify if PQ 1.0 had some kind of fault tolerance.
Result	According to the analysis and discussion of this indicator, the conclusion is that the system (17 functions evaluated) had no fault tolerance mechanisms, therefore: $$r3 = \frac{A}{B} = \frac{0}{17} = 0$$
Limitations	For the evaluation of this indicator the cooperation of the product manager was needed for not being explicit in the documentation fault tolerance mechanisms.
Problems	At this stage the PQ release 1.0 does not contemplate fault tolerance.

Table 79: PQ 1.0: Recoverability

Characteristic	**Data**
Description	The degree to which the product can recover the data directly affected and re-establish the desired state of the system in the case of an interruption or a failure. [12]

Continued on Next Page...

Table 79 – Continued

Characteristic	Data
Measurement and Equation	$$r4 = \frac{A}{B}$$ r4 = Recoverability. A = Number of functions that can recover data directly affected and re-establish the desired state of the system in the case of an interruption or a failure. B = Total number of functions that handles data to be evaluated. Interpretation of the result: $0 <= r1 <= 1$ The closer to 1, the better. Measurement from the standard: [10].
Measurement Method	Documentation was analyzed, architecture, and it was discussed with the product manager to verify if PQ 1.0 has some kind of data recovery mechanism.
Result	According to the analysis and discussion of this indicator, the conclusion is that the functions that handle data (8 functions) do not count with mechanisms of data recovery. Therefore: $$r4 = \frac{A}{B} = \frac{0}{8} = 0$$
Limitations	For the evaluation of this indicator the cooperation of the product manage was needed, because there was not any explicit documentation of data recovery mechanisms.
Problems	At this stage the PQ release 1.0 does not contemplate recoverability of data.

Summary of Reliability Measurement. In Table 80, there is a breakdown of reliability evaluation, separated by sub-characteristics and functionalities evaluated. Figure 21 displays the results of reliability and its sub-characteristics.

Table 80: Data of the reliability evaluation of PQ 1.0, by functions and sub-characteristics

Function ID	**r1: Maturity**		**r2: Availability**		**r3: Fault tolerance**		**r4: Recoverability**	
	t1	**t2**	**t1**	**t2**	**A**	**B**	**A**	**B**
F001					✗		-	
F002					✗		-	
F003					✗		✗	
F004					✗		-	
F005					✗		✗	
F006					✗		✗	
F007					✗		-	
F008					✗		✗	
F009					✗		✗	
F011					✗		-	
F012					✗		-	
F013					✗		-	
F014					✗		✗	
F015					✗		-	
F017					✗		-	
F018					✗		-	
F019					✗		✗	
Results	$A = 0$	$B = 2$	$t1$ = 273 Minutes	$t2$ = 273 Minutes	$A = 0$	$B = 17$	$A = 0$	$B = 8$

Continued on Next Page. . .

Table 80 – Continued

Function ID	r1: Maturity	r2: Availability	r3: Fault tolerance	r4: Recoverability
	$r1 = \frac{A}{B}$ $r1 = \frac{0}{2}$ $r1 = 0$	$r2 = \frac{t1}{t2}$ $r2 = \frac{273}{273}$ $r2 = 1$	$r3 = \frac{A}{B}$ $r3 = \frac{0}{17}$ $r3 = 0$	$r4 = \frac{A}{B}$ $r4 = \frac{0}{8}$ $r4 = 0$
R: Reliability	$R = \frac{r1+r2+r3+r4}{4}$ $R = \frac{0+1+0+0}{4}$ $R = 0.25$			

Notes:

Fault tolerance:
"✓" functions that perform as planned despite the presence of hardware failures or software.
"✗" functions that do not have fault tolerance.
Recoverability:
'✓" functions that can recover and restore data directly affected and restore the desired state of the system in case of a disruption or failure.
"✗" functions that handle data and does not have recoverability mechanisms.
"-" functions where this indicator does not apply.

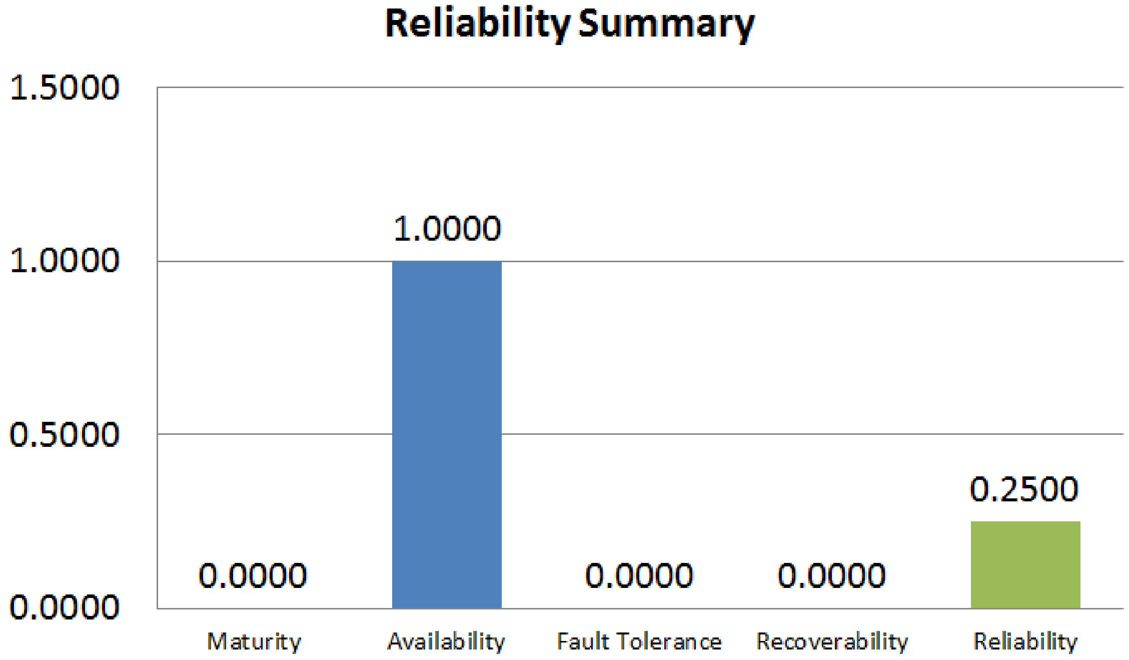


Figure 21: PQ 1.0: Measurement summary of reliability and its sub-characteristics

Traceability to the requirements. The information of the measurement procedure of traceability to the requirements is in Table 81 and Table 82.

Table 81: PQ 1.0: Traceability to the requirements

Characteristic	Data
Description	The product is only traceable if allows to identify and reference clearly the requirement it satisfies. [11]

Continued on Next Page...

Table 81 – Continued

Characteristic	**Data**
Measurement and Equation	$$x1 = \frac{A}{B}$$ x1 = Traceability to the requirements. A = Number of traceable software functions confirmed in review. B = Number of functions checked. Interpretation of the result: $0 <= x1 <= 1$ The closer to 1, the better. Measurement of the standard [10].
Measurement Method	Documentation, requirements from the system were analyzed and contrasted with the developed functions, and traceability tools or traceability matrix were requested.
Result	It was concluded that the 17 functions were not traceable. Therefore: $$x1 = \frac{A}{B} = \frac{0}{17} = 0$$
Limitations	There is no documentation to evaluate this indicator.
Problems	A requirements document is available, but there is no requirement management tool or a traceability matrix.

Summary of Traceability Measurement. In Table 82, there is a breakdown of the traceability evaluation, separated by sub-characteristics and functionalities evaluated.

Table 82: Data of the traceability evaluation of PQ 1.0, by functions and sub-characteristics

Function ID	**X1: Traceability to the requirements**	
	A	**B**
F001	✗	
F002	✗	
F003	✗	
F004	✗	
F005	✗	
F006	✗	
F007	✗	
F008	✗	
F009	✗	
F011	✗	
F012	✗	
F013	✗	
F014	✗	
F015	✗	
F017	✗	
F018	✗	
F019	✗	
Results	$A = 0$	$B = 17$
Traceability	$X1 = \frac{A}{B}$ $X1 = \frac{0}{17}$ $X1 = 0$	

Notes:
"✓" functions with traceability.
"✗" functions without traceability.

4.2.6.2 Quality Calculation

To obtain the estimated quality of PQ 1.0 the Equation 17 is applied on which the estimated measure (ES) is divided by the expected measure (EX) of the value quality indicators, and at the end the result is divided with the total number of evaluated indicators. It must be taken into notice that the ratio of the indicators is normalized to 1.0 when the estimated measure is higher than the expected measure (which leads to a ratio higher than 1.0).

$$Q = \frac{\frac{U_{ES}}{U_{EX}} + \frac{PE_{ES}}{PE_{EX}} + \frac{C_{ES}}{C_{EX}} + \frac{FS_{ES}}{FS_{EX}} + \frac{S_{ES}}{S_{EX}} + \frac{R_{ES}}{R_{EX}} + \frac{X1_{ES}}{X1_{EX}}}{7} \tag{17}$$

$$Q = \frac{\frac{0.9083}{0.9000} + \frac{0.4706}{0.9000} + \frac{0.8750}{0.7000} + \frac{0.8236}{0.7000} + \frac{0.6667}{0.6000} + \frac{0.2500}{0.5000} + \frac{0.0000}{0.4000}}{7}$$

$$Q = \frac{1.0000 + 0.5229 + 1.0000 + 1.0000 + 1.0000 + 0.5000 + 0.0000}{7} = \frac{5.3010}{7} = 0.7176$$

Table 83 lists a summary of the results of the quality estimation, its ratio and exceeded or missed percentage.

Table 83: Value quality indicators of PQ 1.0 with expected measures, estimated measurements, ratio and exceeded or missing percentage

Indicator	**Expected measure**	**Estimated measure**	**Ratio**	**Exceeded or missed percentage from measures**
U: Usability	0.9000	0.9083	1.0092	+0.83%
u1: Learnability		0.9978		
u2: Ease of use		1.4610		
u3: User error protection		0.5385		
u4: User interface aesthetics		0.6357		

Continued on Next Page...

Table 83 – Continued

Indicator	**Expected measure**	**Estimated measure**	**Ratio**	**Exceeded or missed percentage from measures**
PE: Performance Efficiency	0.9000	0.4706	0.5229	−42.94%
pe1: Time behavior		0.4706		
C: Compatibility	0.7000	0.8750	1.2500	+17.50%
c1: Co-existence		0.7500		
c2: Interoperability		1.0000		
FS: Function suitability	0.7000	0.8236	1.1765	+12.36%
fs1: Functional appropriateness		0.8824		
fs2: Accuracy		0.7647		
S: Security	0.6000	0.6667	1.1111	+6.67%
s1: Non-repudiation		1.0000		
s2: Accountability		1.0000		
s3: Authenticity		0.0000		
R: Reliability	0.5000	0.2500	0.5000	−25.00%
r1: Maturity		0.0000		
r2: Availability		1.0000		
r3: Fault tolerance		0.0000		
r4: Recoverability		0.0000		
X1: Traceability to the requirements	0.4000	0.0000	0.0000	−40.00%
Averages	0.5714	0.5706	0.7957	−20.43%
Quality (ratio average, normalizing the ones that exceed 1.0)			0.7176	

Some of the estimated measures of the value quality indicators, exceeded the expected measure, this can be seen in Figure 22.

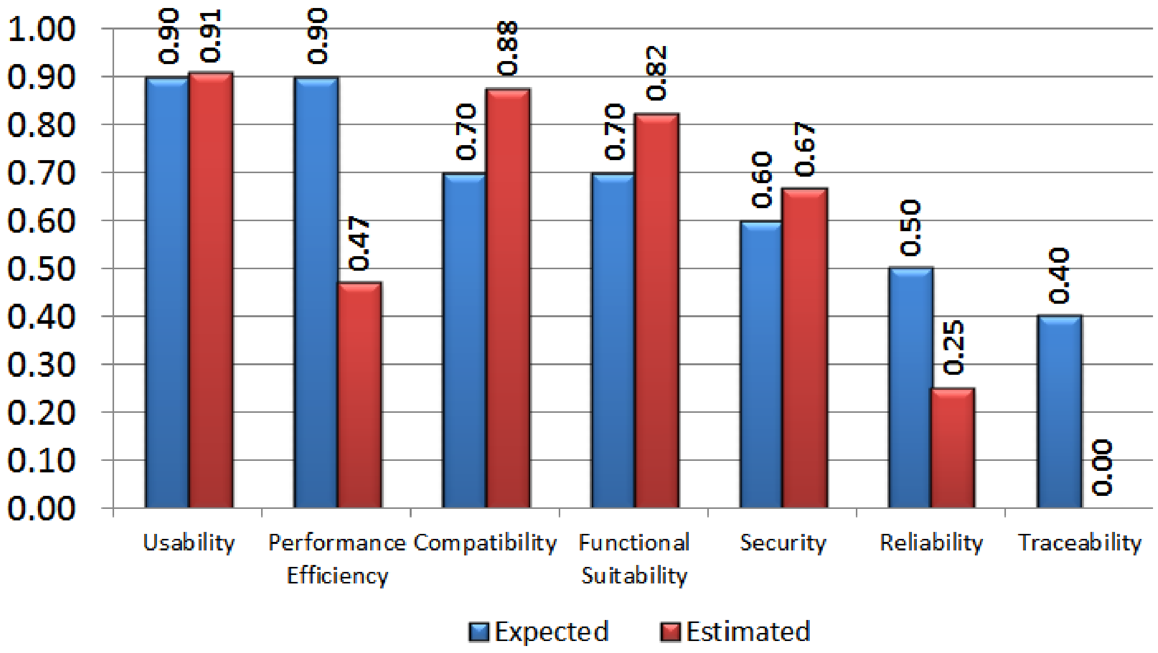


Figure 22: Value quality indicators expected measure vs. estimated measure (without normalizing the estimated measures)

4.2.7 PQ: Function Verification

4.2.7.1 Expected Function

An exercise was conducted to calculate the function points (FP) from the description of the requirements of PQ 1.0 and the described functionalities. For this, the tool *Function Point Modeler*[21] was used to calculate the function points for each functionality listed in Table 49 (regardless of F010 and F016 functionalities, which was not possible to evaluate). The results can be viewed in Table 84.

Table 84: Results of the expected function of PQ 1.0

Results	**Data**
Unadjusted function points	183
Adjustment factor	1.01
Expected adjusted function points (from multiplying unadjusted function point with the adjustment factor)	185

4.2.7.2 Developed Function

From the total of expected function points, a checklist with the developed functions at the moment of the evaluation was performed, it was found that all of the functions were developed, but some of them had failures and errors, which means an extra effort of work to repair them. In Table 85 those functions are listed with their respected function points.

Table 85: Functions of PQ 1.0 that are not fully developed

Functionality	**Function points'**
F013 Filter rules on the library	6.1
F017 Filter rules in the quality model	6.1
F012 Organize rules as a tree view	3
F018 Organize rules in the quality model	3
Total	18.2

Summarizing; the total adjusted function points not properly developed are 18.2. To obtain the real developed function points a subtraction is performed: total adjusted function points expected minus adjusted function points not properly developed. The results of the function estimation and function ratio are in Table 86.

Table 86: PQ 1.0 function data summary.

Results	**Data**
Function points expected	185
Function points developed	166.8
Function ratio	0.90164 (90.16%)

4.2.8 PQ: Cost Verification

The cost estimates and information was given by the project. The cost data is displayed in Table 87.

Table 87: Summary of cost data for PQ 1.0

Results	**Data**
Estimated effort for the release 1.0 of PQ specifications.	500 Man-hours
Real effort of the development of PQ 1.0.	500 Man-hours
Cost ratio. (Estimated cost divided by real cost)	1
The ratio cost is: 1, *that means the team dedicated exactly* 100% *of the estimated to the development effort.*	

4.2.9 PQ: Value Estimation

Table 88 is a summary of the information obtained from the expected measures, estimated measures and the calculated ratio of the different indicators that define the value (function, quality and cost).

Table 88: Summary of the elements from the value equation for the PQ 1.0 value estimation

Elements from the value equation	**Expected (EX)**	**Estimated or real (ES)**	**Ratio (Es/Ex)**
Function	$185FP$	$166.8FP$	0.9016
Quality	100	71.76	0.7176
Cost	$500HH$	$500HH$	1

In equation 18 (from [19]), the value is calculated based on the ratio of the different value indicators previously estimated.

$$Value = \frac{Function + Quality}{Cost} = \frac{0.9016 + 0.7176}{1} = 1.6192 \tag{18}$$

In a hypothetical case that the estimated measures are equal to the expected measures, when calculating the value (Equation 18), the result would be equal to 2.00, this value is taken as reference as seen in Figure 23 to compare expected with estimated. For *PQ 1.0* the value result is equal to 1.6192, based on the reference value it reached 80.96% of the expected value, according to the estimation of the software product value.

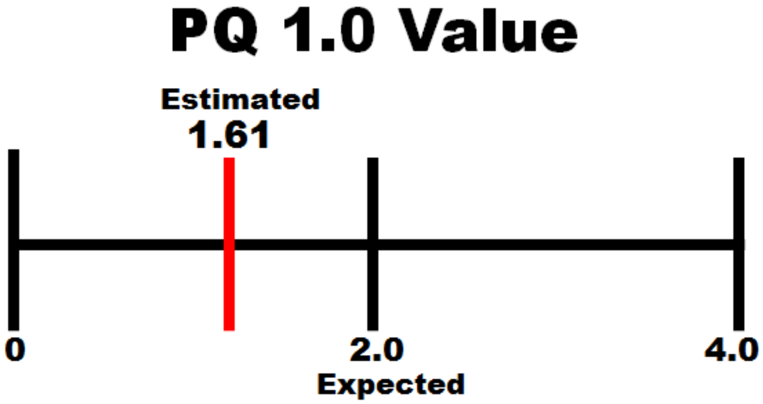


Figure 23: Expected and estimated value of PQ 1.0

Table 89 lists the elements and their percentage achieved in accordance with the measures that were expected of them, this gives a broad overview of the current value of *PQ 1.0*.

Table 89: Elements of the value equation and their achieved percentages according to the expected measures of PQ 1.0

Estimated elements	**Achieved percentage**
Function	90.16%
Quality	71.76%
Cost	100%
Value	80.96%

Based on the results, productivity data can be obtained:

- Function points by man-hour: $166.8/500MH = 0.3336$ FP x MH.
- Percentage of quality by estimated man-hour: $71.76/500MH = 0.14352$ Q x MH.

4.2.10 PQ: Discussion

This section discusses the results of the estimated value of *PQ 1.0*.

4.2.10.1 Value Indicators Analysis

Function. It reached a 90% of the expected functionality, the function results were very close to achieving the expected functionality, this does not mean that they have not been developed in part some features, in fact all the features were developed, the problem is that some of these features are not fully developed and they have some errors which would lead to an extra development effort.

Cost. The initial expected cost was 420 MH, but because of a project decision 80 MH were added, this does not mean that it has exceeded the initial cost, since at the same time that the man hours were increased, some features of functionality were also increased, thus taking 500 MH as the initial cost, that means it is the expected cost. On the other hand the man hours that were used to develop *PQ 1.0* were 500 MH, this means that they dedicated exactly the estimated cost, but as explained in the discussion of functionality, the expected functionality was not fully achieved with the projected cost.

Quality. For quality a 71.76% was achieved, but a further analysis of the value indicators must be performed, in which several details can be found to explain the quality result, in Table 90 these data is discussed.

Table 90: Value quality indicators of PQ 1.0 and their result discussion

Indicator	**Discussion**
Usability	A few problems on usability were detected, especially on the sub-characteristics of protection against user errors and user interfaces.
Performance Efficiency	It was one of the lower indicators, emphasis should be placed on this aspect, as it was one of the indicators that were expected higher, and the indicator: time of response was the only indicator evaluated.

Continued on Next Page...

Table 90 – Continued

Indicator	Discussion
Compatibility	It went well evaluated, but it must be noticed that coexistence was evaluated from Internet browsers, and interoperability is limited only to the company own products, not tested with other products.
Functional Suitability	This aspect is under control, there are only some inconsistencies in some features but they are minimal.
Security	Of the three sub-characteristics of security evaluated (non-repudiation, accountability and authenticity), only authenticity had problems, because in the log files the identity or subject of who performs the action is not saved.
Reliability	It had a good result on availability, but maturity was very low, because according to the product manager there should not be any failures, but in the evaluation tests there were a couple of them. At the moment for the release 1.0, fault tolerance or recoverability is not covered for the functions that handle data.
Traceability to the requirements	There is clear documentation on the list of system requirements, but they do not have any tools for requirements management and traceability or traceability matrix, this indicator does not contribute to the quality, the result is 0.

Exceeded Expected Measures. The result of the estimation of some value quality indicators exceeded the expected measurement, that is to say when the ratio is obtained with the expected measure and the estimated measure, exceeded 1.0(100.00% in percentage), these indicators that their results exceeded are displayed in Table 91.

Table 91: Value quality indicators of PQ 1.0 with their exceeded percentages

Value Quality indicators	Percentage
Compatibility	125.00%
Functional Suitability	117.65%
Security	111.11%
Usability	100.92%

The quality is achieved at 71.76%, this calculation has already been done previously in Section 4.2.9 for which the exceeded amount of the estimated indicators is discarded to stick to the expected quality of the product, since although the indicators that exceed the expected measure are contributing to the quality, they distort the value estimation result of the product, it seems useful to maintain a balance between the expected and estimated, and for this particular case there is a significant deviation. It can be analyzed clearly in Figure 24.

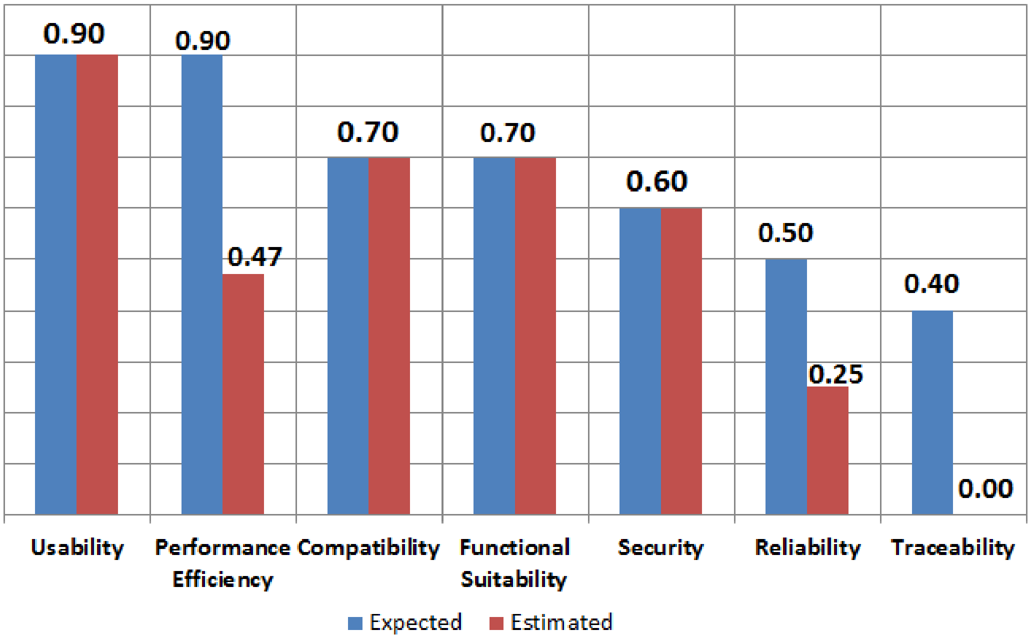


Figure 24: Value Quality Indicators and their expected measure against the estimated measure (normalizing the estimated measure)

Indicators where the estimated measure exceeds the expected measure cannot be ignored, it has been explained that they distort the result of quality and therefore value, but it is data that reflects the reality of that indicator. With the data of the exceeded estimated measures the *product manager* can analyze quantitatively to where and from where should divert efforts, for instance: diverting efforts of the indicators listed in Table 91, and these efforts focus them to increase performance efficiency and reliability.

4.2.10.2 PQ 1.0 Value

In Section 4.2.9 the value estimation can be seen in detail, which reaches a total of 80.96%, that is below of what was expected of *PQ 1.0*. The cost indicator is the only that its result is equal to what is expected, so it is under control. Although the functionality is not achieved as expected, it is worth mentioning that it was by a very low percentage (missed around 10%). The value quality indicator is the one that is affecting mostly the value of the product, since it reached only 71.76%, but it is the one that has more information and more scope to take decisions and change elements to increase the value of *PQ 1.0*.

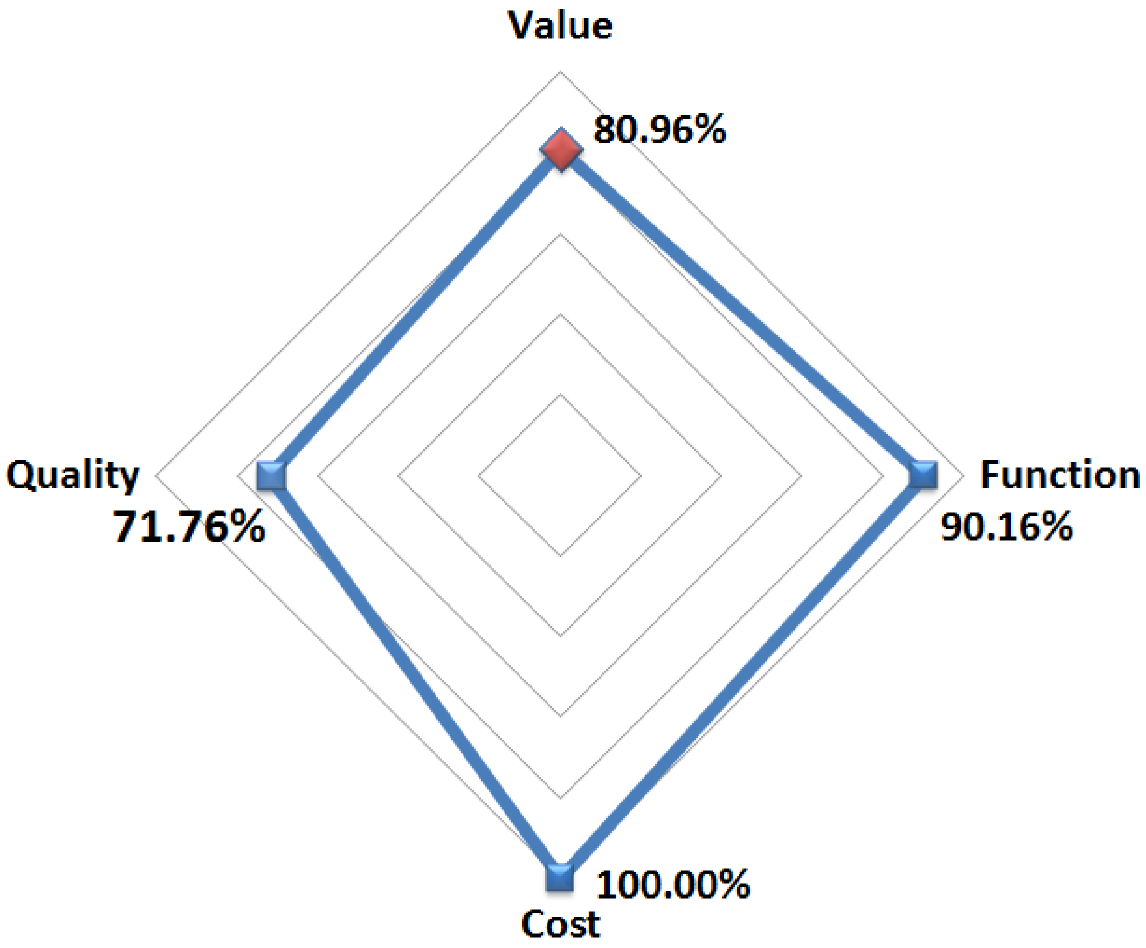


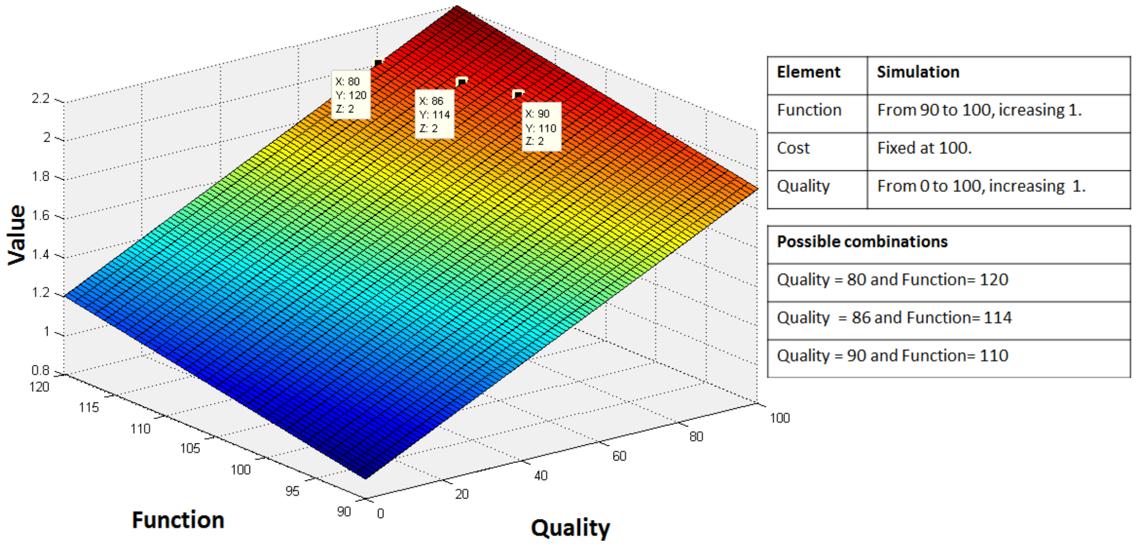
Figure 25: Results of value in percentages and the elements that define value (*PQ 1.0*)

4.2.10.3 Simulations

Since the desired product value is not achieved, it is entirely up to the *product manager* and the company, to make the necessary changes to achieve the desired value of the product, there are a lot of indicators to change, and all these indicators impact the final product value, in this section there will be some suggestions or options of how these changes impact the value.

Simulations were performed with the obtained data; the following are some figures of those simulations with a brief explanation.

Figure 26 it is a simulation varying function and quality, with cost fixed, in this figure some combinations that achieve the desired value can be seen (examples, as there are more combinations of those shown in the figure), for this case is necessary to increase both the quality and function.

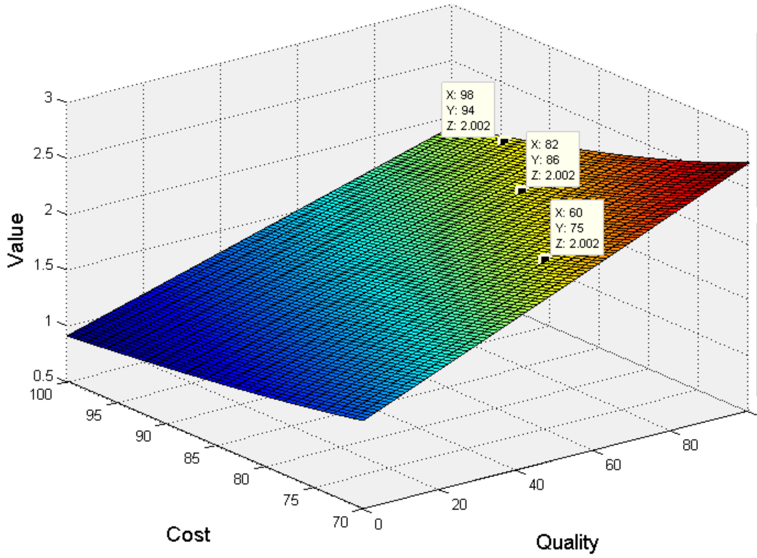


Element	Simulation
Function	From 90 to 100, icreasing 1.
Cost	Fixed at 100.
Quality	From 0 to 100, increasing 1.

Possible combinations
Quality = 80 and Function= 120
Quality = 86 and Function= 114
Quality = 90 and Function= 110

Figure 26: Value simulation, with fixed cost, varying function and quality

Figure 27 it is a simulation varying cost and quality, with fixed function, in this figure some combinations that achieve the desired value can be seen (as examples, but there are more combinations), for this case is necessary to increase the quality and reduce the cost, of course in this case the cost is no longer decreasing but it is an interesting fact to consider for a future release of *PQ*.

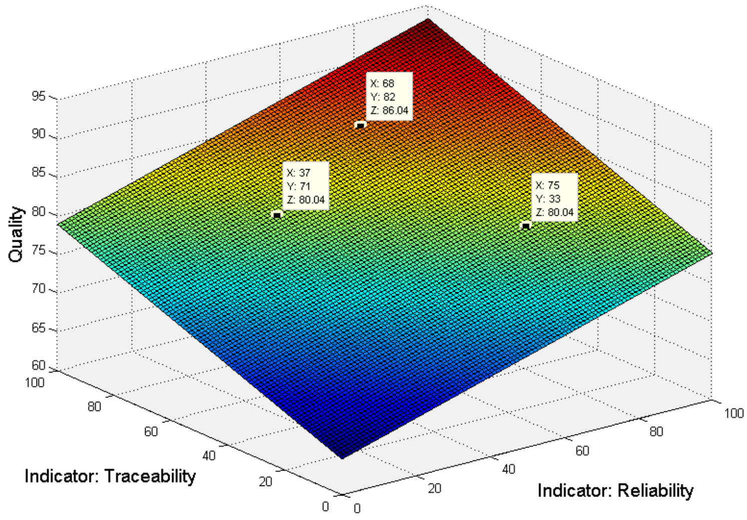


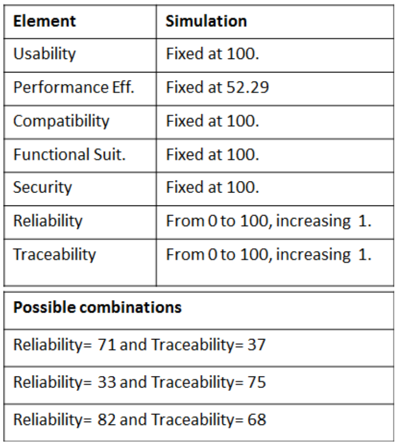
Element	Simulation
Function	Fixed at 90.16.
Cost	From 0 to 70, increasing 1.
Quality	From 0 to 100, increasing 1.

Possible combinations
Quality = 75 and Cost = 60
Quality = 86 and Cost = 82
Quality = 94 and Cost = 98

Figure 27: Value simulation, with fixed function, varying cost and quality

On Figure 28 a simulation is performed, varying the indicators of reliability and traceability, since these are the lowest on the results of the estimation, the rest was fixed with the results of the estimation. The resultant combinations can impact directly on value, this can be seen on Figure 26 and Figure 27, because when quality reaches 80% or 86% other indicators of value can be varied (function and cost) to reach the desired value, obviously all these decisions are totally responsibility of the product manager and the company.





Element	Simulation
Usability	Fixed at 100.
Performance Eff.	Fixed at 52.29
Compatibility	Fixed at 100.
Functional Suit.	Fixed at 100.
Security	Fixed at 100.
Reliability	From 0 to 100, increasing 1.
Traceability	From 0 to 100, increasing 1.

Possible combinations
Reliability= 71 and Traceability= 37
Reliability= 33 and Traceability= 75
Reliability= 82 and Traceability= 68

Figure 28: Simulation of quality, varying reliability and traceability, the lowest indicators of the estimation

4.2.11 PQ: Limitations

The certainty of the results of this value estimation study depends on the experience and perception of the evaluator, the information provided and the cooperation of the product manager and the development team.

For this specific case study the majority of the needed information is available. All functions and needs of PQ 1.0 were specified, but there was not any kind of measure, because of this lack of measurement of the function a task was performed to assign function points to each function. Another particular aspect of this case study is that because of the lack of information the maintainability and its sub-characteristics could not be measured: modularity, reusability, modifiability and testability, this was due to the lack of documentation and because the code could not be verified directly, diagrams of dependency packages were not available, this is an important limitation of the case study.

For the evaluation of some indicators such as the sub-characteristic of usability: user interface aesthetics it was evaluated only with the perception of a single evaluator. It would be of great importance for an assessment in the future to have more than one evaluator or persons able to evaluate this aspect, as this would broaden the vision and perception about this indicator.

The disposition and participation of the product manager was very good, because for every issue that it was required, there was a quick response, resolving doubts and problems that appeared during the development of the estimation.

A big limitation of this case study is that the expected measure was not assigned directly to each sub-characteristic of the value quality indicators, which means it was assigned directly to a value quality indicator. Additionally, to obtain the result of the estimated measure for each value quality indicator an average between the sub-characteristics of the indicator was done, for example: functional suitability it is defined by the measures of functional appropriateness and accuracy.

4.2.12 PQ: Conclusions

This case study of PQ 1.0 has been of great contribution to the thesis, because it allowed the application of the reference model to estimate the value of software products in a real industrial environment of software development, which nurtures various aspects and gives a very good overview of how this approach works in a development company. Based on the objectives outlined in the case study the following conclusions are made:

The software products value evaluation has interesting aspects for the company; it allows visualizing a lot of quantifiable data to know if the development of the product is going on the right track, this quantification gives visibility to determine the status of the software product development.

The element on which the study gives more information and where the project decisions can impact more it is quality, which has revealed several problems that can be viewed in Table 92, but also there are other indicators on where the initial expected measure is exceeded (see Table 91) and from which efforts can be diverted towards the indicators that did not reached the expected measure.

Table 92: Value quality indicators of PQ 1.0 that did not reach their expected measure

Value Quality indicators	**Percentage achieved**
Traceability to the requirements	0.00%
Reliability	50.00%
Performance Efficiency	52.29%

PQ for the release 1.0 did not achieved its expected value, this conclusion can be viewed in detail in Section 4.2.9. It must be taken into account that PQ 1.0 has not been released to the market yet, therefore with the data of the value estimation, the product manager has a wide range of options to make decisions about modifying PQ 1.0 to achieve the value he expects to be the correct one to launch the product to the market with a high value and to keep up with their competitors.

To perform the value estimation of PQ 1.0 it was necessary a total of 67.7 hours of effort (8 hours of staff and evaluator, and 59.7 of only evaluator, see Table 48). The developed functions of PQ 1.0 sum a total of 500 man-hours, this means that the execution of this value estimation model adds a 13.54% of extra effort, of course, keeping in mind that this is the first application of the reference model for software product value estimation, and through the experience this effort will be reduced considerably.

For this particular case the results help to obtain a perspective about the actual estate of the software product value, starting from the elements that define value, like function, quality and cost. Inside of this estimation the most detailed part is the quality, defined together with the product manager.

Lessons have been learned for future executions of the value estimation model, some already explained in the limitations that were taken in the case study. Experience as an evaluator is very important, and has also been a great contribution to refine the value estimation model, especially in the equations and the method of evaluating indicators. Also the experience of evaluate the products of a development team because it is interesting to share views and this helps to implement some of the measures that they deem appropriate for future developments.

Chapter V

CONCLUSIONS

5.1 Main Contributions

The main purpose of this thesis is to define a model to estimate the software product value during the development process which offers visibility and quantitative data. There are interesting contributions in this thesis:

1. A broad review of the state of the art is provided, concerning the software product value estimation area, developing step by step a Systematic Mapping Study in order to gain knowledge about the status of the current research in this topic. Also based on the results of the SMS, the related approaches to software product value estimation and value definitions for software product were analyzed. The importance of the value concept in software products for companies and researchers was an interesting finding; also important gaps in the software product value estimation research were detected. The importance of the quality factor to define value was a key element that later centered the RESVEP (proposed in Chapter 3). The SMS and the findings helped to support the theory and development of the RESVEP. Additionally, a paper was published based on the SMS and the findings of the state of the art in [5].

2. This thesis provides a value definition (Section 3.1.1), specifically for the software products, which is very important to state the value concept in the context of this thesis. Additionally the value indicators concept was proposed (Section 3.1.2), which is the basis of the RESVEP. The value indicators are crucial to this thesis, since value is defined by value indicators, which have the characteristic of being identifiable and quantitative. Those definitions were developed based

on the findings of the state of the art, and were published first in [4].

3. This thesis provides a Reference Model to Estimate the Software Product Value During the Development Process, the RESVEP, defining specifically steps to perform to execute the RESVEP (in Section 3.2) and estimate the value of software products. The RESVEP can be used in the different stages of the development process and it is centered on the value indicators.

4. Support guidelines are provided in order to apply the RESVEP, Section 3.3, offers a set of activities to perform the RESVEP. It also offers a generic set of *Value Quality Indicators* with its measurements and formulas for the development phases (based on ISO 12207 [11]) *Software Detailed Design Process* and the *Software Construction Process* to obtain quality, and the required data to estimate the value.

5. The RESVEP has been applied in case studies. In Chapter 4 two case studies are presented, using a specific instantiation of the RESVEP for each case, the case studies were carried out with real life data. Through these case studies the reference model has proven its feasibility, in this thesis all the elements needed to apply the reference model specific for the *Software Detailed Design Process* and the *Software Construction Process* are provided (requirements, activities, value indicators and its measurements, equations, description, etc.).

5.2 Accomplishment of the Objectives

The thesis objectives are stated in Section 1. The justifications about the accomplishment of each thesis objectives are presented as follows:

1. *Investigate the actual research on software product value estimation.*

   Chapter 2 provides the SMS performed on the software product value estimation topic, and the related approaches on the researched topic. The actual research on software product value estimation is growing, especially on the industrial areas. The SMS development is explained step by step in Section 2.1.

An overview of the software product value estimation actual research is provided in Section 2.2. The concept of value is highly related to perception, therefore the next step was to identify and quantify the elements that define the value concept specifically for software products. Through this identification and quantification the perception would be closer to the real value that the software product delivers to the customer.

2. *To establish the elements that define software product value, those elements should be quantifiable to allow visibility of the value product through the software development process.*

   The concept of value indicators (stated in Section 3.1.2), which is the basis of the RESVEP is very important, they have the characteristic of being quantifiable and the quantification of these indicators offers visibility of the product value through the software development process.

   The value indicators that define value according to the thesis proposal are: *Function*, *Quality* and *Cost*. But a key element that was found on the research is that quality has a major impact on the value of the software product, this is one of the reasons why the RESVEP is centered on quality.

3. *Propose a model to support the value estimation of the software product throughout the software production chain: from the definition of product value from the business areas, to the stages of software product development.*

   The RESVEP supports the estimation and verification of the software product value. This verification is made from the expected value and the estimated value. The expected value is obtained from the business criteria established on the project plan of the product, and the estimated value it is obtained through the proposed approach in the reference model.

The RESVEP offers fully quantitative results in terms of value indicators (which define value) as: *Function*, *Quality* and *Cost*.

These quantitative data help to identify deviations of the product project plan in terms of function, quality and cost, initially established. Identifying these deviations at a correct stage of the development, they could be corrected in order to reach the expected value that the software product needs to obtain success in the market.

4. *Establish the phases and activities of the software development on which the model would be centered.*

   In the scope of this thesis the RESVEP is centered on the *Software Detailed Design Process* and the *Software Construction Process* of [11], which are crucial phases of the software development. This is mainly due the case studies instantiation of the RESVEP (Sections 4.1.4 and 4.2.4). Additionally if there are deviations or problems in these processes, they could still be corrected before launching a product to the market. The model can be instantiated to any process, phase or discipline of the software development process, through the instantiation it is fully adaptable to the specific case in which is going to be applied.

5. *Set up the requirements and activities needed to apply the software product value estimation model.*

   Section 3.3 presents a full guide with the activities and required data to apply the RESVEP. The activities are aligned with the steps of the RESVEP presented in Section 3.2. Also an example of the Quality Calculation and Value Equation Application are provided in Section 3.3.2.3 and Section 3.3.3.

6. *Elaborate a case study to prove the utility and feasibility of the model.*

   Chapter 4 presents two performed case studies, which were done with real life data. Through the case studies the RESVEP has proved its feasibility.

   Based on the developed case studies the conclusions of the extra effort needed to apply the RESVEP is 13.54%, the extra effort needed from the product manager and/or development team is 1.6%, and the effort of the evaluator is 11.94%. The detail of the effort data is in Section 4.2.2.2. This data means that if the RESVEP is going to be applied to a 750 man-hours project, the extra effort would be approximate of: 851.55 man-hours (12 hours of the product manager and/or development team and 89.55 man-hours for the evaluator). These are approximate figures of effort, the better experience of the evaluator and the support of a tool to apply the RESVEP should reduce those figures substantially.

   Additionally based on the case studies, the utility of the RESVEP was identified, according to the results, deviations on the development process that affects the value indicators, and mainly quality details were identified which at the end will affect the value of the product when released to the market. The value discussion and conclusions of the case studies in Sections 4.1.10 and 4.2.10 offers an overview of the value indicators results of the case studies for the product managers of the estimated products.

5.3 Future Work

Firstly, facilitate the step 1 of the RESVEP, Project Context Definition for the Value Estimation, with a generic software development process which must comprise the entire life-cycle. This is a very extensive work because for each work product, different *Value Quality Indicators* with its measurements and equations should be developed. This will allow the evaluator the possibility to apply the RESVEP in the entire software development process with a systematic methodology. In this sense, the software development process' work products will then have pre-defined the value indicators at hand, just to be selected in a particular estimation.

Design and develop a visual tool to ease the execution of the RESVEP. The tool should allow to model the specific case study where is going to be applied, assigning the project context, selecting value indicators, create links between the work products and the value indicators, calculate the value indicators and the product value and finally create reports and graphics of the result data.

A future step is to integrate the visual tool to model and execute the RESVEP with different plug-ins or tools that verify the code and the project documents, in order to measure the indicators automatically; this would save a lot of time and errors to the evaluator.

The execution of more case studies, the case studies presented in this thesis were very helpful to validate the RESVEP. More case studies will have a great impact on the RESVEP and the feedback from the industrial areas would help to improve the model itself, the activities, the equations of the value indicators, the way that the estimation is applied, simulations, etc.

5.4 Limitations

As in any software product development project, the plan from the business side is very important, the project plan is fundamental it is the basis from where the RESVEP's technical specifications are stated. If those technical specifications are wrong the application of the RESVEP is pointless.

The RESVEP success depends on the quality of the project information provided to execute the estimation. Certain elements should be provided according to the project context on where the RESVEP is going to be applied (for example: the architecture and design for the *Design Process*).

The *Expected Measures*(the planned measures that are defined by the project personnel) of the *Value Quality Indicators* have been assigned to the major value quality indicators, but for more accuracy on next case studies it should be assigned to the quality sub-indicators. For example instead of assigning directly an expected measure to *Security*, the stakeholders should assign measures to the quality sub-indicators that define *Security*: *Confidentiality*, *Integrity*, *Non-repudiation*, *Accountability* and *Authenticity*. This will give more accuracy to the expected measures, and at the end to the value estimation.

The cooperation and communication of the stakeholders involved in the estimation process is very important, because the exchange of information and opinions it is crucial to obtain accurate results in the application of RESVEP.

The experience and personal view of the evaluator is a main factor of the estimation, especially on the value indicators on where the perception of the evaluator is highly used.

An important limitation is that the RESVEP utilizes the SQuaRe [12] and the ISO 9126 [10]. The access to the information of these standards is not free, and the use of RESVEP in a commercial way it is not possible until the right to use those documents is paid.

Appendix A

ORIGINAL PUBLISHED PAPERS

A.1 Papers Published with Results of this Thesis

A.1.1 Findings based on a Systematic Mapping Study on Software Product Value Estimation

Castro, O., Espinoza, A., Martínez-Martínez, A.: Findings based on a Systematic Mapping Study on Software Product Value Estimation. In: Tendencias en investigación e Innovación en ingeniería de Software: un enfoque práctico, pp. 5764. CONISOFT (2012).

URL: http://pcyti.izt.uam.mx/ARTICULOS/OscarCastro_CONIISOFT2012.pdf

This paper presents all the work related to Section 2, it describes step by step the development of a Systematic Mapping Study (SMS) related to the software product value estimation topic. It also summarizes the related approaches on software product value estimation of different authors, and provides useful findings in the software product value estimation area.

Abstract. Software product value has become a major concern in development companies; there is a fierce competition to deliver better products and offer higher value to the customer. There is a need for knowledge and better understanding on how to manage the value concept; this is especially important for the product offering companies. There is a lot of research in the product value area from the business side, but the specific area of software product value is barely addressed nowadays. The most related approach is Value-Based Software Engineering, but this approach addresses the added value to the Software Engineering tasks. The purpose of this paper is to present an analysis of the related literature on software product value following the

guidelines of Systematic Mapping Study methodology. The current state of the art will be presented as well as interesting findings which serve as a basis for next research in this topic.

A.1.2 Estimating the Software Product Value during the Development Process

Castro, O., Espinoza, A., Martínez-Martínez, A.: Estimating the software product value during the development process. In: O. Dieste, A. Jedlitschka, N. Juristo (eds.) Product-Focused Software Process Improvement, Lecture Notes in Computer Science, vol. 7343, pp. 7488. Springer Berlin / Heidelberg (2012).

URL: http://dx.doi.org/10.1007/978-3-642-31063-8_7

This paper introduces a proposal for the definition of value, specifically for software products and the concept of value indicators. It proposes a *process model to estimate the software value, based on value indicators for development work products*, which later would become the RESVEP as it is stated on this thesis, it comprises the information on Section 3. It also presents the results and findings of the case study presented in Section 4.1.

Abstract. Nowadays software companies are facing a fierce competition to deliver better products but offering a higher value to the customer. In this context, software product value has becoming a major concern in software industry, leading for improving the knowledge and better understanding about how to estimate the software value in early development phases. Other way, software companies encounter problems such as releasing products that were developed with high expectations, but they gradually fall into the category of a mediocre product when they are released to the market. These high expectations are tightly related to the expected and offered software value to the customer. This paper presents an approach for estimating the software product value, focusing on the development phases. We propose a value indicators approach

to quantify the real value of the development products. The aim is early identifying potential deviations in the real software value, by comparing the estimated versus the expected. We present an internal validation to show the feasibility of this approach to produce benefits in industry projects.

REFERENCES

[1] Anderson, J. and Narus, J., "Business marketing: Understand what customers value," *Harvard Business School*, 1998.

[2] Barney, S., Aurum, A., and Wohlin, C., "A product management challenge: Creating software product value through requirements selection," *J. Syst. Archit.*, vol. 54, pp. 576–593, June 2008.

[3] Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., and Grünbacher, P., eds., *Value-Based Software Engineering.* Berlin: Springer, 2006.

[4] Castro, O., Espinoza, A., and Martinez-Martinez, A., "Estimating the software product value during the development process," in *Product-Focused Software Process Improvement* (Dieste, O., Jedlitschka, A., and Juristo, N., eds.), vol. 7343 of *Lecture Notes in Computer Science*, pp. 74–88, Springer Berlin / Heidelberg, 2012.

[5] Castro, O., Espinoza, A., and Martinez-Martinez, A., "Findings based on a systematic mapping study on software product value estimation," in *Tendencias en investigacion e Inovacion en ingenieria de Software: un enfoque practico*, pp. 57–64, CONISOFT, 2012.

[6] Cedergren, S. and Larsson, S., "Improving traceability by focusing on value during development," in *Proceedings of the 1st International Workshop on Value-Based Software Traceability (VALSOT 2011) in XP 2011 Conference* (de Madrid, U. P., ed.), 2011.

[7] Day, E. and Crask, M. R., "Value assessment the antecedent of customer satisfaction," *Journal of Consumer Satisfaction, Dissatisfaction and Complaining Behavior*, vol. 13, 2000.

[8] IEEE, "Ieee recommended practice for software requirements specifications," 1998.

[9] international function point users group, T., *Function Point Counting Practices Manual.* The international function point users group, 191 Clarksville Road Princeton Junction, NJ 08550 U.S.A., release 1.4.1 ed., April 2000.

[10] International Organization for Standardization, "ISO/IEC 9126 Information Technology - Software Product Quality," November 1999.

[11] International Organization for Standardization, "ISO/IEC 12207:2008 Systems and Software Engineering - Software life cycle processes," 2008.

[12] International Organization for Standardization, "ISO/IEC FCD 25010: Systems and software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality models for software product quality and system quality in use," July 2009.

[13] Jacobson, I., Booch, G., and Rumbaugh, J., *The unified software development process.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[14] Keane, "Application rationalization," tech. rep., Keane, 2011.

[15] Kitchenham, B. and Charters, S., "Guidelines for performing systematic literature reviews in software engineering," Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[16] Kitchenham, B. A., Budgen, D., and Brereton, O. P., "Using mapping studies as the basis for further research - a participant-observer case study," *Information & Software Technology*, vol. 53, no. 6, pp. 638–651, 2011.

[17] Kruchten, P., "Architectural blueprints: The "4+1" view model of software architecture," *IEEE Software*, vol. 12, pp. 42–50, November 1995.

[18] Ojala, P., "Experiences of a value assessment for products," *Softw. Process*, vol. 14, pp. 31–37, January 2009.

[19] Ojala, P., "Value of project management: a case study," *WSEAS Trans. Info. Sci. and App.*, vol. 6, pp. 510–519, March 2009.

[20] Oxford Dictionaries, 2011.

[21] point Modeler Inc., F., "Function point modeler standard (http://www.functionpointmodeler.com)."

[22] Rönkkö, M., Frühwirth, C., and Biffl, S., "Integrating value and utility concepts into a value decomposition model for value-based software engineering.," in *PROFES'09*, pp. 362–374, 2009.

[23] SAVE International, "Value methodology standard and body of knowledge," June 2007.

[24] Schwaber, K. and Beedle, M., *Agile Software Development with Scrum.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st ed., 2001.

[25] Shefveland, A., "Application value assessment," tech. rep., Fujitsu, 2004.