A Hierarchical Software Quality Model for Estimating Function Point Analysis

BASSEM MATALKAH, NADEEM EL-ADAILEH, MOAATH KHAMAYSA HAJAYA
Department of Data Science, Faculty of Information Technology,
Mutah University,
P.O. Box 7, Mutah 61710, Karak,
JORDAN

Abstract: - Research indicates that the method of scaling software according to the quantity of business functions an application needs to complete is called function point analysis (FPA). As a result, a model that guarantees the quality attributes of such systems is required. An information system's functional size can be determined using the Function Point Analysis (FPA) approach. While there are several quality methodologies now in use for estimating FPA in general, none of them are specifically focused on Hierarchical Task Analysis (HTA). This contribution includes a study of an analysis carried out on multiple approaches for different function points: ISO/IEC 20926:2003 International Function Point Users Group (IFPUG), ISO/IEC 20968:2002 Software engineering Mk II Function Point Analysis, ISO/IEC 19761:2011 Common Software Measurement International Consortium (COSMIC), and ISO/IEC 29881:2010 Functional Size Measurement method (FiSMA 1.1). The study also lists the drawbacks of the current methodologies, including their propensity to overlook specific Non-Functional Requirements (NFR) features like Accuracy, Security, Time Behavior, and Recoverability or their omission of a description of the methods used for quality measurement. This investigation leads to the development of a new software quality model that includes newly defined sets of sub-characteristics that are associated with a standard set of quality characteristics that are acceptable for estimating FPA components. The new software quality model matches the right kind of stakeholders with related quality attributes, avoiding some of the drawbacks of the previous approaches. The Software Quality Model (SQM) aims to help companies with the process of creating systems for Function Point Analysis (FPA) estimation, which is essential to the success of Software Project Management (SPM).

Key-Words: - Stakeholders, Function Point Analysis (FPA), Hierarchical Task Analysis (HTA), ISO/IEC 19761:2011, Non-Functional Requirements (NFR), International Function Point Users Group (IFPUG), Software Project Management (SPM).

Received: April 29, 2024. Revised: November 22, 2024. Accepted: December 23, 2024. Published: January 27, 2025.

1 Introduction

The accuracy of the planning and estimating of the activities to be completed, as well as the estimated time, cost, and quality, are crucial to the success of a software development project. Software developed with multiple user parameters can be measured to determine the project's users, [1]. The estimation of effort can have an impact on scope, cost, schedule, and quality, [2].

Numerous techniques, including parametric models, dynamics-based models, learning-oriented techniques, expert-based techniques, regression-based models, and composite Bayesian techniques for combining expertise and regression-based models, are presented to produce trustworthy estimates, [3]. The primary input for many efforts estimates models and tools, including COCOMO II [4], is the functional size of the product [5].

Functional size is a measure of the amount of functionality provided by the software, completely independent of any technical or quality. The capability to accurately quantify the size of software in an early stage of the development lifecycle is critical to software project managers for evaluating risks, [6].

Function Point Analysis was invented by Allan Albrecht of IBM as a means of sizing business application software independently of the technology used for its development, Albrecht introduced function point analysis methodology to evaluate software development from the point-of-view of functional points or business requirements of the application. By 1986 a core group of function point analysis users established the International Function Point User's Group, [7].

In the literature on approaches, the reader can observe that different authors have proposed different approaches, for example ISO/IEC 29881FiSMA 1.1 functional size measurement method proposed in 2010, ISO/IEC 19761 Common Software Measurement International Consortium (COSMIC) presented in 2011, ISO/IEC 20926 International Function Point Users Group (IFPUG) proposed in 2003, and SO/IEC 20968 Software engineering Mk II Function Point Analysis proposed in 2002. Such approaches are intended to estimate the Function Point Analysis of software project management in general; all of the approaches ignore Non-Functional Analysis.

Establishing a baseline for development team productivity, monitoring productivity over time, identifying process or system inefficiencies, assessing the efficacy of process improvements, assisting management in decision-making, and estimating maintenance costs and/or effort of implemented systems are just a few advantages of using function point analysis in software project management.

A Functional and Non-Fictional Point Analysis has become an important requirement for managing software products. A starting point for a Functional and Non-Fictional Point Analysis approach might be to consider some of the existing Functional Point Analysis approaches.

ISO/IEC 29881 FiSMA 1.1 Functional Size Measurement Method, ISO/IEC 19761Common Software Measurement International Consortium (COSMIC), ISO/IEC 20926 International Function Point Users Group (IFPUG), and SO/IEC 20968 Software engineering Mk II Function Point Analysis, are non-hierarchy models.

All of the above models have been studied, and analyzed to develop a new model that (i) overcomes some of the existing limitations, (ii) features standard quality characteristics and associated sub-characteristics to develop new hierarchies' model, (iii) associates all categories of stakeholders involved in the process with the appropriate set of quality characteristics. As a result, the new Hierarchical Software Quality Model will be utilized as a tool for estimating Functional Point Analysis to manage software products. The remainder of this paper is structured as follows: Analysis study of selected functional point analysis is examined in Section 2. Section 3 addresses the problem. Proposed multi-criteria decision methodology in Section 4. Section 5 highlights the new model features. Finally, the conclusion and recommendations are presented in Section 6.

2 Analysis study of Selected Functional Point Analysis

The choice of existing models to be evaluated here was based on the most standard and well-known quality models, the following models have been described: The International Standard Organization (ISO), International Electro-Technical Commission (IEC), and ISO 14143 Standard on FSM together comprise the specialized system for global standardization. Working groups were gathered by ISO in 1994 to create an international standard for measuring functional sizes. To provide a collection of standards and technical documentation for functional size measurement methods, they established the ISO/IEC 14143 series, [8], [9], [10], [11], [12], [13]. The ISO/IEC 14143 series consists of six parts: Part 1 of ISO/IEC 14143-1, which was released in 2007, is devoted to concept definition. Its objectives are to outline the general guidelines for using an FSM method and to clarify the basic ideas of functional size measurement (FSM), [8]. The second part of the 2011 publication, ISO/IEC 14143-2, addresses the conformance evaluation of software size measurement methods to ISO. Its goal is to set up a framework and outline the procedure for comparing a proposed FSM method's compliance with ISO/IEC 14143-1:2007. It also included a checklist to help with the conformity evaluation of the standard FSM method and recommendations for judging the competency of conformity evaluation teams, [9]. Functional size measurement method verification is covered in Part 3 of ISO/IEC 14143-3:2003, with the aim of "establishing a framework for verifying the statements of an FSM method and/or for conducting the tests requested by the verification sponsor, [10]". Part 4: ISO/IEC 14143-4:2002 outlines a Reference model, the purpose of which is to validate an FSM technique, [11]. The purpose of Part 5: ISO/IEC 14143-5:2004 is to "describe the characteristics of functional domains and procedures by which characteristics of Functional User Requirements (FUR) can be used to determine functional domains." Part 5 deals with determining functional domains for use with functional size measurement, [12]. Part 6: ISO/IEC 14143-6:2005 is a guide that explains how to use the ISO/IEC 14143 series and other international standards. It offers an overview of the standards that are connected to FSM and the connections between them [13], the core ideas of the FSM technique are as follows: (i) Functional User criteria (FUR): these are usage criteria that serve as the foundation for designing and assessing interactive systems to satisfy specific user needs,

[14]. (ii) Base Functional Component (BFC): the fundamental unit of Functional User Requirements that is measured and described by an FSM Method, [8]. (iii) Type of base functional component: a category of base functional component that has been defined, [15].

2.1 International Function Point Users Group (IFPUG)

In [16] say the Function Point Method FPM is a method to measure the functional size of software from the user's perspective. Functional size is defined according to ISO/IEC 14143-1:2007 define concepts as: "size of the software derived by quantifying the Functional User Requirements, where the Functional user requirements FUR are in turn defined as a subset of the User Requirements, [8]. The IFPUG Function Point Measurements FPM measures the functionality in software delivered to the user as required by the user, and quantified by the IFPUG Counting Practices Manual CPM, [9].
In [16] define a user as any entity that needs to communicate with a piece of software. This definition is similar to that of an actor in object-oriented or use-case technology, where the functional size measure is unaffected by the technology employed in implementation.

Figure 1 (Appendix) illustrates IFPUG Function Point counting process: (i) Collect Functional Requirements: is the process of identifying, documenting, and controlling changes to the project requirements, (ii) Perform Mapping: are rules according IFPUG Function Point Counting Practices Manual CMP, (iii) Functional Requirements according to IFPUG Model are functional component types : (a)Transactional Functions TF e.g. External Input EI an elementary process in which data crosses the boundary from outside to inside, External Output EO an elementary process in which derived data passes across the boundary from inside to outside , External Inquiry EQ an elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files, (b) Data Functions DF e.g. Internal Logical File ILF a user identifiable group of logically related data that resides entirely within the application boundary, External Interface File EIF a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application , (iv) Perform Evaluation Measure : are rules according IFPUG CMP.

After identifying five Base Functional component types, they are assigned functional complexity weights as 'simple', 'average' or 'complex' depending on some parameters described next paragraph Pre-defined FP values are assigned for each Base Functional Component Type category and associated complexity and these are summed to arrive at a corresponding number of FP, [17].

ences (FTRs) are counted to rate the 'Transaction Functions 'complexity whereas for 'Data Functions', the DETs and Record Element Types (RETs) are counted. The size of a piece of software is deemed to be the sum of the sizes of its five types of components as identified in the software FUR. The sum of the weighted counts of the BFCs gives the size of the software in units of 'Unadjusted Function Points' (UFP).

The Data Element Types (DETs) and File Type ReferThe weights of an elementary process can range from 3 to 7 UFP and from 5 to 15 UFP for a file. There is therefore an upper limit to the assigned complexity of a BFC, [18].

There are some limitations found in this approach, ignores Non-Functional Requirements (NFR) and ignores Hierarchical Task Analysis (HTA) and the measure is unaffected by the technology employed in implementation.

2.2 Finnish Software Measurement Association (FiSMA)

FiSMA Functional Size Measurement Method (FSM) Version 1.1 was proposed in 2010 [19], and measures the size of software, FiSMA method was developed in the working group of the "Finnish software measurement association" that is called FiSMA. This method was applied in Finland.

In [20] says FiSMA method measured more than 600 software development projects between 1997 to 2003. FiSMA functional size measurement method is based purely on Functional User Requirements (FUR), FiSMA function size measurement methods are service-oriented. Which can be the identification of all the different services given by software. Here the rule requirements are functional. It measures the size of the software from the perspective of the user.

Functional user requirements specify as a function i.e. Base Function Components (BFC), Base Function Type are sub-attributes which are divided into many small components then they are measured as a separate domain and further the functional size is added to it, the BFC is divided into the base function classes and the Base Function Classes BFC is further divided into Base Function Types BFT. There are seven classes of BFC: (i) Interactive end-user navigation and query services (ii) End-user input services (iii) End-user output

services (iv) Interface services to other applications. (v) Interface services from the other application. (vi) Data storage services. (vii) Algorithm and manipulation services. These BFC classes are further decomposed into the BFC types. There are 28 BFC types, [21]. Figure 2 (Appendix) illustrates the FiSMA 1.1 relationship chain between users and the developed piece of software involving user needs and services, [22].

There are some limitations found in the FiSMA method, based purely on Functional User Requirements, ignores Hierarchical Task Analysis, and the rules that calculate the function size are very complicated, because there are lots of components, and measurement of all the components are very typical, some components are not quantifiable. That's why we introduced the new model with few components which easily calculates the size of the mobile application.

2.3 Common Software Measurement International Consortium (COSMIC)

COSMIC (ISO/IEC 19761:2003) [23] was developed to measure the functional size of business application software, real-time software, and hybrid of these, [24], [25]. COSMIC Software Context Model contains the principles that pertain to identifying the nature and structure of the Software to be measured as required by the COSMIC method, leading to the identification of its FUR. The Generic Software Model: it contains the principles to be applied to the Functional User Requirements FUR to extract and measure the elements that contribute to the functional size using the COSMIC method, [18], [23], [26], [27].

COSMIC measurement takes place in three phases: (i) Measurement Strategy Phase: is about gathering Functional User Requirements FUR, and defining the purpose of the measurement. (ii) The mapping phase, is meant to map an artifact to the COSMIC Generic Software Model to identify Functional Processes and Data Movements. (iii) Measurement phase, is where the actual measurement in CFP (COSMIC Function Point) is done by summing up the number of data movements identified. Figure 3 (Appendix) illustrates the main COSMIC measurement process phases.

There are some limitations found in this approach, ignores Non-Functional Requirement NFR, ignore Hierarchical Task Analysis, and has not been designed for measuring the functional of a piece of software, or its parts, which is characterized by complex mathematical algorithms or other specialized and complex rules, such as can be found in expert systems, simulation software, self-learning software and weather forecasting systems, or processes continuous variables such as audio sounds or video images, such as can be found in computer game software, musical instruments and the like.

2.4 ISO/IEC 20968 Software Engineering Mk II Function Point Analysis in 2002

A new approach for determining the system size is provided by [28] and is referred to as the "Mk II Function Point Analysis Method" in [29]. The following presumptions are included in this metric: (i) Logical input/process/output (i): interfaces are handled as any other input or output at the logical level; (ii): queries are seen as any other input, process, or output; (iv): a logical file notion is regarded as an entity at the logical transaction level. The Mk II technique consists of the following steps: - (i) Define the Boundary of the Count, a system's boundary is the logical line dividing users from the system, when a user and a system interact, it is used to identify logical transactions like inputs and exits those cross boundaries. (ii) Identify the Logical Transactions: although a transaction may be completed from several points within the application, it is only counted once. (iii) Determine and Classify Data Entity Types: these are logical data structures that hold information that the user can understand, there is only one kind of data entity type used in the Mark II method: objects, to count objects, they must be accurately identified. (iv) Count the types of input, output, and data entity types that are referenced, this can be done with a basic spreadsheet; the values must be added up and a formula performed to determine the precise MKII FPA value. (v) Determine the Functional Size, the functional size of the system can be determined by counting the transactions and objects in it once they have been identified, weighted counts of objects inside the system border and input/exit transactions are used to describe the functional size of the system,

One way to describe the Mark II formula for product size expressed in unadjusted function points (UFP) can be written [28], [29]:

$$UFP = N_I W_I + N_E W_E + N_o W_o$$

where N_I denotes the number of types of input data elements, W_I denotes the input data element type's weight, N_E denotes the number of references to entity types, W_E denote the entity-type reference's weight, N_o denotes the total number of types of output data elements and Wo denote the output data element type's weight. [28], [29] defines the following weights for the Mark Il formula:

$$UFP = 0.44N_I + 1.67N_E + 0.38N_O$$

In [29], uses the Technical Complexity Adjustment (TCA) method to count the software's ultimate size:

$$TCA = 0.65 + C^* TDI$$

where the current industry average value of "C" is 0.005, and the TDI depicts the total degree of influence as calculated by the sum of the adjusting factors. The function point index (FPI) can be determined:

$$FPI = UFP * TCA$$

There are some limitations found in this approach, ignore Non-Functional Requirement NFR, and ignore Hierarchical Task Analysis. There are no reliable comparisons based on large enough statistical samples of system size metrics, and there are no tools available to compute them during the specification phase, the elementary process is derived from the input-processing output logic, which states that every elementary process has a logical file reference and an input processing output component, It is not affected by the project management method to be used (e.g., object-oriented, SSADM, Information Engineering, etc.) or the project management technique to be utilized (e.g., waterfall, spiral, incremental), and it is a measure of the logical, business requirements, independent of how they are implemented.

3 Addressing the Problem

There is no general consensus on the Functional Point Analysis Quality Model which can fit into all types of applications, taking into account the various classifications: IFPUG, FiSMA, COSMIC, and Mk II ignored Non Functionality, and Hierarchical Task Analysis, FiSMA the rules which calculate the function size is very complicated, some components are not quantifiable, COSMIC not been designed for measuring the functional of a piece of software, or its parts, which is characterized by complex mathematical algorithms or other specialized and complex rules, such as can be found in expert systems, simulation software, self-learning software and weather forecasting systems, or processes continuous variables such as audio sounds or video images, and Mk II no reliable comparisons based on large enough statistical samples of system size metrics, no tools available to compute the specification phase, not affected by the project management method to be used (e.g., object-oriented, Structured Systems Analysis and Design Method SSADM, Information Engineering, etc.) or the project management technique to be utilized (e.g., waterfall, spiral, incremental). That's why we introduce the new Hierarchical Functional Point Analysis Quality Model to easily calculate the Size of Function Point Analysis. Furthermore, none of the existing approaches make an effort to link a particular quality with the kinds of stakeholders that are most interested in that characteristic.

Software products are created when proper physical design and programming practices are applied; actions related to need identification, interface design, and general network design determine how effective the product is. Project management activities such as meeting deadlines, increasing productivity, and saving resources are linked to process efficiency. Process effectiveness is correlated with general management practices such as group and human relations, change management, and leadership as these foster good relationships between the members of the information systems development team.

Three areas are identified by IFPUG as a software product's external quality characteristics, gather functional requirements, perform mapping rules by the IFPUG Function Point Counting Practices Manual CMP, and identify functional component types by the IFPUG Model, therefore represents the effectiveness of the product. Internal properties defined by the FiSMA Functional Size Measurement Method designate seven Base Function Classes BFC: interactive end-user query and navigation services; end-user input services, end-user output services; interface services to and from other applications; data storage services; and algorithm and manipulation services. FiSMA thus represents product efficiency. IFPUG and FiSMA do not discuss process effectiveness or efficiency; however, a reference model known as ISO/IEC TR 15504-2 offers a solution. The International Electro-Technical Commission is what IEC stands for. It classifies the processes into four groups, each of which is linked to a collection of processes, similar to the ISO/IEC model that is discussed in [26]. Table 1 (Appendix) displays these.

4 Suggested Methodology

Several Function point analysis (FPA) techniques are being practiced under different operational environment conditions for scaling software based on the number of business activities for a software

development project to be successful, the accuracy of the planning, and estimates of the activities to be performed, together with the projected time, cost, and quality, are essential, therefore, a model that ensures the quality attributes of such systems is needed. While there are several quality methodologies now in use for estimating FPA in general, none of them are specifically focused on Hierarchical Task Analysis (HTA). This contribution comprises a study of an analysis performed on several methods for different function points: ISO/IEC 19761:2011 Common Software Measurement International Consortium (COSMIC), ISO/IEC 20926:2003 International Function Point Users Group (IFPUG), ISO/IEC 20968:2002 Software engineering Mk II Function Point Analysis, and ISO/IEC 29881:2010 Functional Size Measurement technique (FiSMA 1.1). A new software quality model that includes newly defined sets of sub-characteristics related to a standard set of quality characteristics that are appropriate for estimating FPA components is developed as a result of this work. To set guidelines for our work here, to build a new quality model for estimating FPA, a suggested methodology is presented, and concluded in the following steps based on IFPUG, FiSMA, and Simple Guide to Hierarchical Task Analysis [27]:

Step 1: Identify a limited number of high-level quality attributes that have been agreed upon, and then break down each attribute into a set of subordinate attributes using a top-down approach.

Step 2: Differentiate between measures that are internal and external. Specifically, internal metrics— also referred to as "white box" metrics measure a product's internal attribute (such as its specification or source code) throughout the design and coding phases, these distinctions are crucial for FPA components. External metrics, on the other hand, focus on the behavior of the system from an external perspective during testing and component operation. Actually, "black-box" external measures are better suited for FPA components.

Step 3: Determine which user types are stakeholders for each high-level quality attribute.

Step 4: Put the pieces together and create a new model that recognizes the right stakeholders for each set of qualities by implementing concepts from Simple Guide to Hierarchical Task Analysis [27], IFPUG, and FiSMA

Step 5: At the product and process level, Identify attributes for each sub-characteristic.

Step 6: Draw a framework (shows FPA components Quality-Attributes).

5 Highlighting the New Model Features

Our new model's goal is to construct a model that can be used with a range of Function Point Analysis (FPA) based systems. The IFPUG serves as the foundation around which our model is built since it incorporates the shared qualities of software quality that the other three techniques support, as seen in Table 2 (Appendix). The FiSMA approach was excluded because the rules which calculate the function size are very complicated, there are lots of components, and measurements of all the components are very typical, and some components are not quantifiable.

The Sven areas for function point analysis, as proposed by IFPUG, Mark II, and COSMIC as a standard, are shown in Table 3 (Appendix).

The assessment of the high-level collection of characteristics and the corresponding sub-characteristics is discussed below; this is the application of step 1 of our methodology:

Transactional functions refer to the features that enable the user to handle data. The ability of a software product to fulfill explicit and implicit needs when employed under predetermined circumstances is known as functionality. An elementary process that handles data or control information coming from outside the application boundary is known as an external input sub-characteristic, maintaining one or more internal logical files and/or changing the system's behavior are the major goals of an external input sub-characteristic. A basic process that transmits data or controls information outside the application boundary is known as an external output sub-characteristic, an external output's sub-characteristic main purpose is to give information to the user via logic processing, either instead of or in addition to data retrieval or control information, at least one mathematical formula, computation, or generated data must be present in the processing logic, an external output sub-characteristic may also change the way the system behaves or preserve one or more internal logical files.

The functionality offered to the user to satisfy internal and external data requirements is referred to as data functions. Sub-characteristic internal logical

file is a user-identifiable collection of logically connected data or control information that is kept within the boundaries of an application, its principal purpose is to store data that is kept up to date by one or more of the program's basic functions. Sub-characteristic external interface file holds data referenced through one or more elementary processes within the boundary of the application counted, this means that an external interface file counted for an application must be in an internal logical file in another application, an external interface file is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application.

The processing part, output message, and input message comprise logical transactions. The input message, the output message, and the processing part can make up an external inquiry. Because COSMIC is limited to certain forms of data transfer, its granularity is considerably smaller. Let's say a user wants to add a name to the database, but he wants to make sure the name doesn't already exist beforehand. The user is then presented with a confirmation message whether the name was added. In Mark II, the name serves as the input, accessing the entity containing the name serves as the processing part, and confirmation serves as the output, however, there is a name entry in COSMIC that may be read to. For these reasons, we do not use COSMIC and Mark II in the Suggested software model.

Stakeholders are any individual or group that will be affected by the system, whether directly or indirectly. The end user who uses the system and every other person in an organization who might be impacted by its installation are considered stakeholders. Business managers and engineers working on linked systems development or maintenance could be considered additional system stakeholders. In this work, the proper category of evaluators for each quality feature has been named by using the normal set of stakeholders as described in [30]. The analysts who create the business models, the end users who engage with the system, the quality assurance personnel who do product testing, and the project managers who design and manage the process. The following criteria for identifying a stakeholder:

(i) The analysts and quality assurance should be able to verify that the solution meets all requirements, including functional and non-functional.
(ii) The architects to verify the solution and assess its suitability as a solution by evaluating trade-offs, this means outlining the system's objectives precisely.
(iii) The solution can be built by the developers. approach calls for breaking up the solution into understandable parts, each having a distinct interface, definitions, and dependencies that are explicitly mapped out
(iv) Quality assurance can test the product, this depends on the mentioned partitioning to schedule unit tests and traceability to confirm deployed features and attributes.
(v) The Project manager is qualified to handle the process, because this depends on dependencies (to plan work) and partitioning (to establish work units for teams and people), the project manager must be able to identify "intermediate deliverables" that are testable, useable, and enable the demonstration of working progress.

The third phase of our process can be implemented by reorganizing the stakeholders' domain according to the aforementioned classification:

(i) According to end-user requirements, the solution must provide the Transactional Function, and observable properties (External Input, External Output, and External Inquiries), and be validated by analysts and Quality Assurance.
(ii) The solution must offer the Data Function, observable properties (External Logical File and External Interface File), and requirements set forth by the Project Manager and Business Owner as confirmed by analysts and Quality Assurance.

Table 4 (Appendix) lists the parts that make up our new model. As a result, we have incorporated into our model the shared attributes that most other models have found and agreed upon, and they align with the FPA component evaluation criteria. We did, however, leave out a few traits that don't align with the new model specifications. We have included new features that are essential to strengthen our new model. Therefore, the above justification applies to every alteration phase, whether removal or additions. Next, each high-level feature that the new model supports has a new set of sub-characteristics associated with it. The entire system development life-cycle, including the operational and maintenance phases, stakeholders—the team members in charge of creating, maintaining, interacting with, and/or using the information system—have been categorized and matched with the appropriate characteristics.

The new model's ultimate structure, complete with all related parts, is depicted in Figure 4 (Appendix), which carries out the fourth stage of our methodology, this figure has three layers, the first one is stakeholder, which is formed of end users, analysis, quality assurances, business owners, project manager, Which is connected with the second layer H-L-Characteristic which is formed of transaction function and data function which will lead to the third layers, sub characteristics that formed of internal logical file, external logical file, external input, external output, and external inquiries.

6 Conclusion and Recommendations

A software development project's success depends critically on how well the tasks to be accomplished are planned and estimated, as well as on how long they should take, how much they should cost, and how well they will work. As a result, a model that guarantees the quality attributes of these kinds of systems is required. Our contributions include a study of an analysis carried out on multiple approaches for different function points: ISO/IEC 20926:2003 International Function Point Users Group (IFPUG), ISO/IEC 20968:2002 Software engineering Mk II Function Point Analysis, ISO/IEC 19761:2011 Common Software Measurement International Consortium (COSMIC), and ISO/IEC 29881:2010 Functional Size Measurement method (FiSMA 1.1). The study also lists the drawbacks of the current methodologies, including their propensity to overlook specific Non-Functional Requirements (NFR) features. This investigation leads to the development of a new software quality model that includes newly defined sets of sub-characteristics that are associated with a standard set of quality characteristics that are acceptable for estimating FPA components.

These models' characteristics have been examined, and evaluated, and their drawbacks described. In particular, ISO/IEC 29881 FiSMA 1.1, ISO/IEC 19761 COSMIC, ISO/IEC 20926 IFPUG, and ISO/IEC 20968 Mk II did not directly address Non-Functional Requirements and Hierarchical Task Analysis of a software product. The highly complex function size calculation rules can be found in ISO/IEC 29881 FiSMA 1.1. ISO/IEC 19761 COSMIC is not to measure the function of a piece of software or its parts that are typified by complex mathematical algorithms or other specialized and complex rules, as those found in expert systems. ISO/IEC 20968 Mk II measures the logical, business requirements regardless of how they are implemented. It is unaffected by the project management technique (e.g., waterfall, spiral, incremental) or the project management method (e.g., object-oriented, SSADM, Information Engineering, etc.).

Despite many drawbacks, we discovered that the ISO/IEC 20926 IFPUG is the most attractive model out of all the ones that have been investigated. We based our new model on the ISO/IEC 20926 IFPUG because of this. To help with the construction of the new model, which specializes in estimating functional point analysis, we established a four-step methodology. The analysis stage helped us avoid repeating these restrictions while also utilizing general quality models that already exist.

A fresh set of sub-characteristics has been defined for each of the justified high-level characteristics that have since been projected. One of the main advantages of the new model is the inclusion of stakeholders, or the team members responsible for creating, developing, maintaining, and interacting, the categories of end users, analysts, quality assurance, project managers, and business owners are matched with the relevant characteristics that each of them finds important.

Finally, the components are assembled to create the new model. Our suggested model is more specialized and better than current models, but it is unable to quantify internal quality attributes. This can be done in subsequent studies by using an assessment method like the Analysis Hierarchy Process (AHP).

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work the authors used Grammarly for language editing. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References:

[1] Kristanto, Andri. 2018. Perancangan Sistem Informasi dan Aplikasinya (Edisi Revisi). Yogyakarta: Gava Media, [Online]. https://elibrary.bsi.ac.id/readbook/204997/perancangan-sistem-informasi-dan-aplikasinya (Accessed Date: August 20, 2024).

[2] Mohagheghi, P. Anda B., and R. Conradi, "Effort estimation of use cases for incremental large-scale software development", *27th International Conference on Software*

Engineering, 2005. ICSE 2005, pp. 303-311. St. Louis, MO, USA, 2005, https://doi.org/10.1109/ICSE.2005.1553573.

[3] Boehm, B., Abts, C. & Chulani, S. Software development cost estimation approaches — A survey. Annals of Software Engineering 10, 177–205 (2000), https://doi.org/10.1023/A:1018991717352.

[4] Boehm, Barry W. 2000. *Software Cost Estimation with Cocomo II*. Upper Saddle River, N.J, London: Prentice Hall: Prentice Hall International, [Online]. https://dl.acm.org/doi/abs/10.5555/1795822 (Accessed Date: August 20, 2024).

[5] Lavazza, Luigi. "Convertibility of functional size measurements: new insights and methodological issues." *In Proceedings of the 5th International conference on Predictor Models in Software Engineering (PROMISE '09)*. Association for Computing Machinery, New York, NY, USA, Article 9, 1–12, 2009, https://doi.org/10.1145/1540438.1540451.

[6] Abrahão, S., Poels, G. & Pastor, O. A functional size measurement method for object-oriented conceptual schemas: design and evaluation issues. *Software & System Modeling*, 5, 48–71 (2006). https://doi.org/10.1007/s10270-005-0098-x.

[7] Wijayasiriwardhane, Thareendhra, and Richard Lai. "Component Point: A system-level size measure for component-based software systems." *Journal of Systems and Software*, 83, no. 12 (2010): 2456-2470. https://doi.org/10.1016/j.jss.2010.07.008.

[8] ISO/IEC 14143-1:2007: Information Technology—Software Measurement—Functional Size Measurement, Part 1: Definition of Concepts, *Int'l Org. for Standardization / Int'l Electro technical Commission, 2007 and related International Standards, 1998*, [Online]. https://www.iso.org/standard/38931.html (Accessed Date: August 20, 2024).

[9] ISO/IEC 14143-2:2011–Information Technology–Software Measurement–Functional Size Measurement–Part 2: Conformity Evaluation of Software Size Measurement Methods to ISO. *IEC, 2011 and related International Standards*, 2002, [Online]. https://www.iso.org/standard/52325.html (Accessed Date: August 20, 2024).

[10] ISO ISO/IEC 14143 -3: 2003, Software Engineering- Functional Size Measurement Part 3: Verification of Functional Size Measurement Methods, [Online]. https://www.iso.org/standard/31918.html (Accessed Date: August 20, 2024).

[11] ISO/IEC TR 14143-4:2002 Information Technology -- Software Measurement -- Functional Size Measurement - Part 4: Reference Model, [Online]. https://www.iso.org/standard/31919.html (Accessed Date: August 20, 2024).

[12] ISO/IEC TR 14143-5:2004, Information Technology -- Software Measurement -- Functional Size Measurement -- Part 5: Determination of Functional Domains for Use with Functional Size Measurement, 2004, [Online]. https://www.iso.org/standard/39986.html (Accessed Date: August 20, 2024).

[13] ISO/IEC FCD 14143-6: 2012: Guide for the Use of ISO/IEC 14143 and related International Standards, 2005, [Online]. https://www.iso.org/obp/ui/#iso:std:iso-iec:14143:-6:en (Accessed Date: August 20, 2024).

[14] ISO/IEC 25066:2016(en) Systems and software engineering — Systems and software Quality Requirements and Evaluation. Accessed, [Online]. https://www.iso.org/standard/63831.html (Accessed Date: August 20, 2024).

[15] ISO/IEC 19761:2011(en) Software engineering COSMIC: a functional size measurement method, [Online]. https://www.iso.org/standard/54849.html (Accessed Date: August 20, 2024).

[16] Bundschuh, Manfred, and Carol Dekkers. "The IFPUG Function Point Counting Method." *The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement* (2008): 323-363. https://doi.org/10.1007/978-3-540-68188-5_12.

[17] Gencel, Cigdem, and Carl Bideau. "Exploring the convertibility between IFPUG and COSMIC function points: preliminary findings." In *2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement*, pp. 170-177. Assisi, Italy, IEEE, 2012. DOI: https://ieeexplore.ieee.org/document/6472583.

[18] Kralj, Tomaž, Ivan Rozman, Marjan Heričko, and Aleš Živkovič. "Improved standard FPA method—resolving problems with upper boundaries in the rating complexity process." *Journal of Systems and Software*, 77, no. 2

(2005): 81-90. https://doi.org/10.1016/j.jss.2004.12.012.

[19] ISO/IEC 29881:2008, Software Engineering, FiSMA Functional Size Measurement Method, Version 1.1. *International Organization for Standardization*, 2008, [Online]. https://www.iso.org/standard/45724.html (Accessed Date: August 20, 2024).

[20] Singh, Nidhi, and Devpriya Soni. "Proposing New Model for Effort Estimation of Mobile Application Development". *International Journal of Computer Applications*. 170. (2017) 14-18, DOI: http://dx.doi.org/10.5120/ijca2017914719.

[21] Cigdem Gencel, Onur Demirors. "Functional size measurement revisited". ACM Transactions on Software Engineering *and Methodology (TOSEM)*, Vol. 17, Issue 3 Article No.: 15, pp.1-36 (2008). DOI: https://doi.org/10.1145/1363102.1363106.

[22] ISO/IEC 29881:2010(en) Information technology Systems and software engineering FiSMA 1.1 functional size measurement method, [Online]. https://www.iso.org/standard/56418.html#:~:text=FiSMA%201.1%20is%20intended%20for,the%20perspective%20of%20the%20users (Accessed Date: August 20, 2024).

[23] Desharnais, Jean-Marc, The Common Software Measurement Int'l Consortium FFP, version 3.0, Measurement Manual, Common Software Measurement Int'l Consortium, 2007, [Online]. https://totalmetrics.com/function-points/COSMIC-Method-v3.0-Measurement-Manual.pdf (Accessed Date: August 20, 2024).

[24] Oligny, Serge, Alain Abran, and Denis St-Pierre. "Improving software functional size measurement." *In Proceedings of 14th International Forum on COCOMO and Software Cost Modeling*, Los Angeles, USA. 1999, [Online]. https://s3.amazonaws.com/publicationslist.org/data/a.abran/ref-1980/439.pdf (Accessed Date: August 20, 2024).

[25] A. Abran, FFP release 2.0: An implementation of COSMIC functional size measurement concepts, in *FESMA*, vol. 99, pp. 4–7, [Online]. https://espace2.etsmtl.ca/id/eprint/11975 (Accessed Date: August 20, 2024).

[26] ISO/IEC 33002:2015 Information technology — Process assessment — Requirements for performing process assessment. Published (Edition 1, 2015), [Online]. https://www.iso.org/standard/54176.html (Accessed Date: August 20, 2024).

[27] O'Donoghue, Jack "A Simple Guide to Hierarchical Task Analysis", [Online]. https://makeiterate.com/a-simple-guide-to-hierarchical-task-analysis/ (Accessed Date: August 20, 2024).

[28] Symons, Charles R. "Function point analysis: difficulties and improvements." *IEEE Transactions on Software Engineering*, 14, no. 1 (1988): 2-11. DOI: https://doi.org/10.1109/32.4618.

[29] Symons, Charles R.. "Software sizing and estimating - Mk II FPA, function point analysis." *Wiley series in software engineering practice* (1991), [Online]. https://api.semanticscholar.org/CorpusID:39923622 (Accessed Date: August 20, 2024).

[30] Thomas D. LaToza. "Using architecture to change code: studying information needs". *OOPSLA '06: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*. pp.764-765. Portland Oregon USA, 2006. https://doi.org/10.1145/1176617.1176714.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.

APPENDIX

Table 1. ISO/IEC TR 15504-2 1998(E) mode process types

Category	Processes
Customer-Supplier	System or Product Acquisition Process, Supply Process, Requirement Bidding Process, Operation
Engineering	Development, Software and System Maintenance
Support	Documentation, Configuration Management, Quality Assurance Process, Verification, Validation, Joint Review, Auditing, Problem Solving Process
Management	Management, Project Management, Quality Management, Risk Management
Organizational	Organizational Alignment, Change Management, Improvement, Infrastructure Process, Measurement Process, Re-use, Process

Table 2. Quality characteristics in IFPUG, Mark II, and COSMIC models

	IFPUG	Mark II	COSMIC
Elementary Process	X		
Logical Transaction		X	
Data Movement Type within a Functional Process			X
Transactional Functions	X		
Data Functions	X		
Logical Transactions		X	
Entry			X
Read			X
Write			X
Exit			X
External Input	X		
External Output	X		
External Inquiry	X		
Internal Logical File	X		
External Interface File	X		
Data Element Type	X	X	
File Type Referenced-	X		
Record Element Types	X		
logical Entity Types		X	

Table 3. IFPUG, Mark II, and COSMIC Basic Function component BFC

Basic Function Component	Characteristics	Sub-Characteristics
Elementary Process	Transactional Functions	External Inputs, External Outputs, External Inquiries
	Data Functions	External Logical File, External Interface File
Logical Transactions	Logical Transaction	No Sub-Characteristics
Data Movement within a Functional Process	Entry, Exit, Read, Write	No Sub-Characteristics

Table 4. Tabular representation of the components of the new model

Stakeholders	Characteristics	Sub- Characteristics
End user, analysts, quality assurance	Transactional Function	External Input External Output External Inquiries
Project Manager and Business Owner	Data Function	External Logical File External Interface File

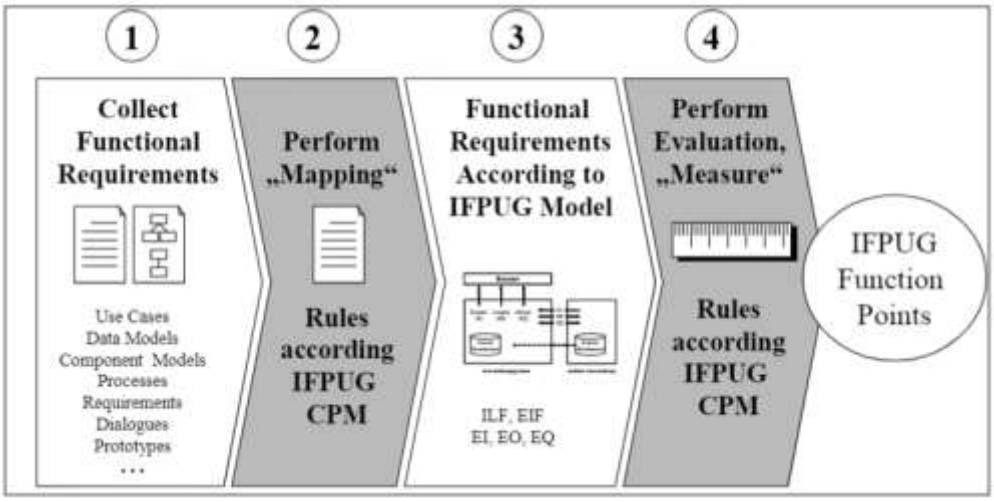

Fig. 1: IFPUG Function Point counting process

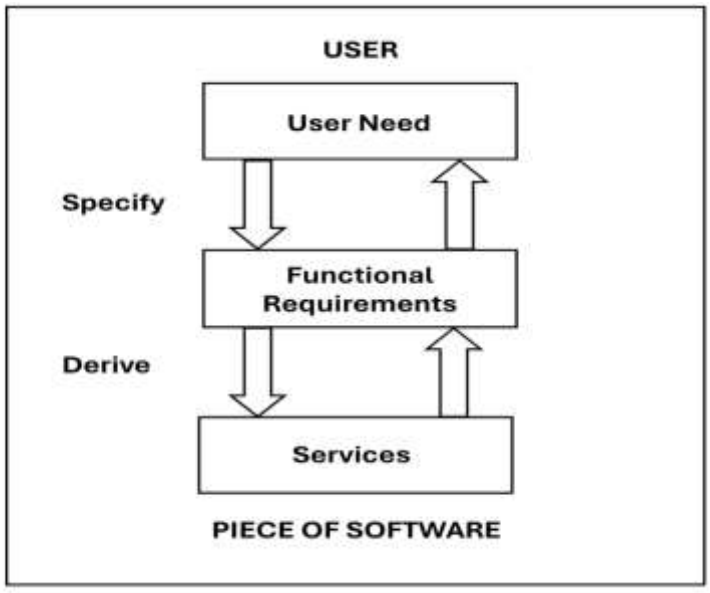

Fig. 2: Relationships between users and the developed piece of software

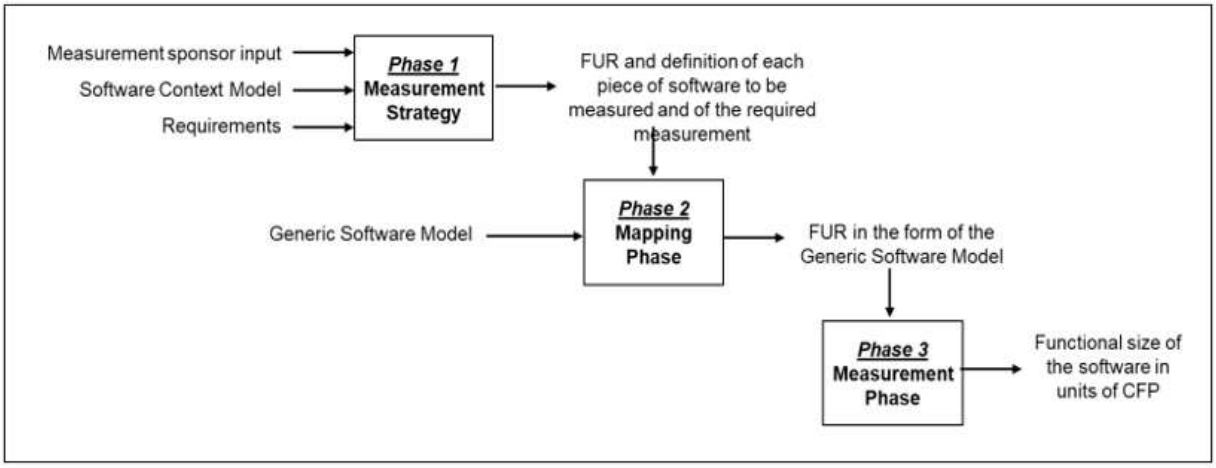

Fig. 3: COSMIC method measurement process

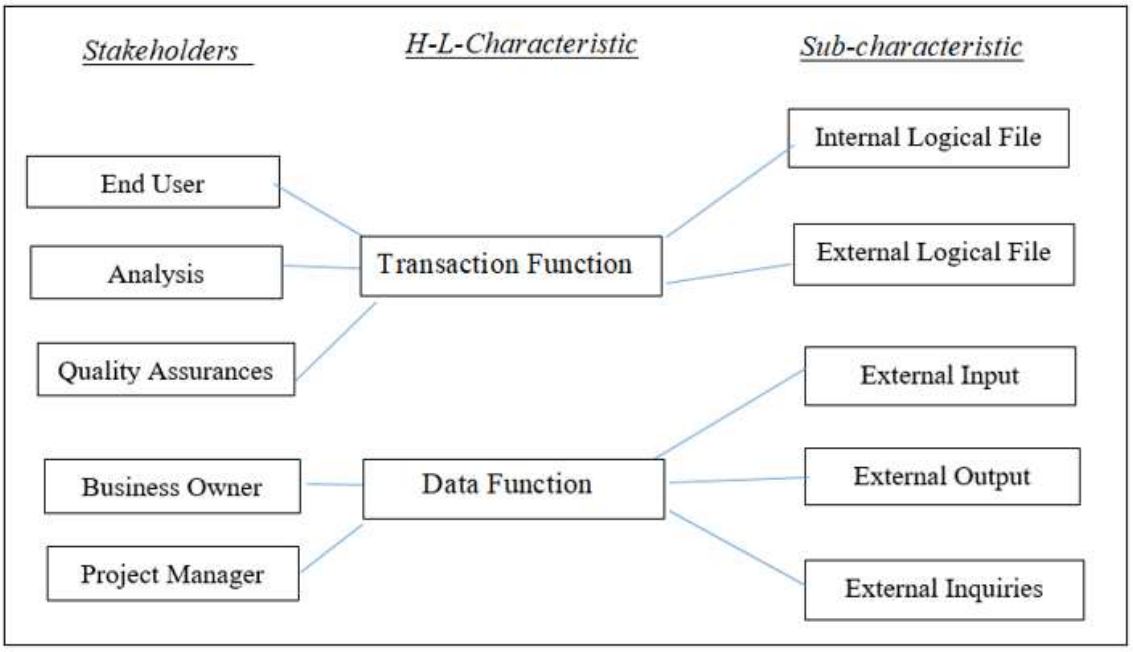

Fig. 4: The new quality model for Estimating Functional Point Analysis