Conceptual Association of Functional Size Measurement Methods

Onur Demirors, Middle East Technical University
Cigdem Gencel, Blekinge Institute of Technology

Functional size determines how much functionality software provides by measuring the aggregate amount of its cohesive execution sequences. Alan Albrecht first introduced the concept in 1979. Since he originally described the function point analysis (FPA) method, researchers and practitioners have developed variations of functional size metrics and methods (see the “Related Research in Functional Size Measurement” sidebar).

Although functional size measurement (FSM) methods give roughly similar results, they’ve been designed neither to measure the same attribute nor to use the same rating scale. Consequently, when measured using different methods, a software product has different functional sizes. Each method has its own definition of functionality and uses different counting schemas for different functional user requirements (FUR) entities. Moreover, FSM methods use their own underlying software engineering concepts rather than those of a particular software development methodology. They use different abstractions during their measurement processes or view the same concepts from different perspectives.

In practice, these differences cause difficulties. In many cases, software organizations must change the FSM method used or convert one FSM method’s size measures to that of another. Several things might influence the need for this, including the acquirers’ requirements, a specific method’s suitability for a new implementation domain, or the need to use benchmark sets established in specific implementation domains. Because measurement processes, rules, and counting schemas aren’t the same, changing existing methodology is difficult for organizations. An inability to use the established historical data in a new methodology makes the change even more challenging. In addition, the lack of a well-established conversion formula creates difficulties for developing and using benchmark sets.

To overcome these difficulties, we developed the Unified Model (UM) to measure functional size via three FSM methods simultaneously: International Function Point Users Group (IFPUG) function point analysis (FPA), Mark II (MkII) FPA, and Common Software Measurement International Consortium (Cosmic) FP. With some limitations, UM lets us convert one method’s functional sizes to those of another method, thus addressing both the need for conversion and the need to use different methods in a single project or organization. To develop UM, we first examined the similarities and differences between the three FSM methods and defined a unified terminology for the common measurement concepts they use.
Related Research in Functional Size Measurement

Several researchers and practitioners have made significant attempts to measure functional size. Since the introduction of the first method for measuring function points, researchers have developed new functional size measurement (FSM) methods to improve or extend such methods’ applicability.1 Since 1996, the International Organization for Standardization (ISO) has been working to clarify and establish common principles for FSM methods and has developed the 14143 standard family.2 Currently, ISO certifies Mark II Function Point Analysis (MkII FPA), the International Function Point Users Group (IFPUG) FPA, the Common Software Measurement International Consortium (Cosmic) FPA, the Netherland Software Metrics Association (NESMA) FSM,3 and the Finnish Software Metrics Association (FiSMA) FSM4 as international standards.

Researchers have conducted several studies evaluating and comparing FSM methods.1,5 Both Thomas Fetcke and his colleagues5 and Marjan Hericko and his colleagues6 proposed formal representation models for MkII FPA, IFPUG FPA, and Cosmic FPA.

In addition, several studies examine the conversion of functional sizes obtained using different methods. Charles Symons compared the conversion of MkII FPA and IFPUG FPA functional sizes to one another on the basis of statistical analyses of data sets.8 They couldn’t obtain a unique conversion formula because formulas varied among different organizations. Later, Jean-Marc Desharnais and his colleagues identified reasons for the outlier projects’ data in defining conversion formulas.9 These include nonhomogeneity in the distribution of FPA function types and documentation types varying across development methodologies, or variation in the quality of such documentation across organizations.

The results of the literature survey show that further research is required to explore the differences and similarities between FSM methods. Moreover, solid approaches should be developed for converting functional sizes measured via different methods to each other.

References

Conceptual Similarities and Differences

To investigate the similarities and differences between different FSM methods, we defined a generic FSM process.4 On the basis of this process, we depict IFPUG FPA, Cosmic FP, and MkII FPA’s common properties and the differences among their measurement subprocesses.

FSM methods define two phases for measuring software that help to identify items deemed relevant for functional size.2,8

The first phase of all FSM processes is to extract FURs from the available artifacts and express them in a form suitable for measuring functional size. FURs represent an application as a set of transactions and data types.9

A transaction takes data types as input, processes them, and gives them as outputs as a result of the processing. FSM methods use transactions and data types to determine base functional components (BFCs are elementary units of an FUR that an FSM method defines and uses for measurement purposes). Then, these methods categorize each BFC as a BFC type and identify the type attributes relevant for obtaining the base counts.9

The second phase calculates each BFC’s functional size by applying a measurement function to the BFC types and their related attributes. The measurer then aggregates the results to compute the software system’s overall size.

Figure 1 shows the conceptual map of the generic FSM processes’ core concepts (rectangles) and the relationships among these concepts (arrows). It also shows the different terminologies used and the different concepts the IFPUG FPA, Cosmic FP, and MkII FPA methods specifically define (ovals).

Data Type Concept

One core concept among the three FSM methods is the data type, which the measurer recognizes in the form of a data element type (DET) or a data group. A DET represents the smallest data item meaningful to the user and is an attribute of an object of interest that’s related to a transaction. DETs are structured into logically related groups—that is, data groups.

According to IFPUG FPA, a DET is “a unique user-recognizable, nonrepeated field.”5 MkII FPA describes DET as “a unique user-recognizable, nonrecursive item of information about entity types.”6 Cosmic FP labels DET as a data attribute and defines it as “the smallest parcel of information, within an identified data
group, carrying a meaning from the perspective of the FURs.”

All three FSM methods also agree on the data group concept, but each uses different terminology. IFPUG FPA labels a data group a data function and defines it as a “user-identifiable group of logically related data or control information referenced by the application.”

MkII FPA calls a data group a data entity type and defines it as “something (strictly, some type of thing) in the real world about which the business user wants to hold information.”

Cosmic FP calls a data group a data group and defines it as a “distinct, nonempty, nonordered, and nonredundant set of data attributes where each included data attribute describes a complementary aspect of the same object of interest.”

Each method has its own classification for data groups. IFPUG FPA has two types of data functions: internal logical file (ILF) and external interface file (EIF). ILFs are those that the users maintain within an application’s boundary, whereas EIFs are those that are maintained in another application’s boundary.

MkII FPA has two data entity types: primary and nonprimary, depending on whether the data entity type is primary for that application. The measurement manual provides a detailed discussion about primary data entity types.

Cosmic FP’s guidelines for sizing business applications classify a data group as a primary data group or a secondary data group on the basis of whether some “thing” is an object of interest to certain types of users in the functional processes they use. IFPUG FPA doesn’t recognize this characteristic.
Cosmic FP differentiates data groups with respect to their degrees of persistence and distinguishes only transient and persistent (that is, short or indefinite). IFPUG FPA and MkII FPA, on the other hand, don’t recognize the persistency characteristics explicitly. The degree of persistency is related to the location of the data group, which might be on an I/O device or in either volatile or permanent storage. When the base count is obtained, not only most of the measurement concepts but also the measurement scales are common.

**Transaction Concept**

All three FSM methods mostly agree on the transaction concept. IFPUG FPA defines a transaction as a *transactional function* and defines it as “the smallest unit of activity meaningful to the user.” It must be self-contained, and it leaves the application in a consistent state.

MkII FPA labels a transaction as a *logical transaction* (LT) and defines it as “the lowest level business process triggered by a unique event of interest in the external world, or a request for information. When wholly complete, it leaves the application in a self-consistent state in relation to the unique event.”

Cosmic FP calls a transaction a *functional process* and defines it as “an elementary component of a set of FURs comprising a unique, cohesive, and independently executable set of data movements. It is triggered by one or more triggering events and complete when it has executed all that is required to be done in response to the triggering event type.”

Even though the transaction concept is similar among all three methods, differences occur in how entities and their attributes are treated when measuring a transaction’s functionality.

IFPUG FPA defines three types of transactional function: external input (EI), external output (EO), and external inquiry (EQ). However, MkII FPA and Cosmic FP don’t differentiate between transaction types. On the other hand, MkII FPA considers three components of an LT: input, processing, and output. Cosmic FP defines a functional process as comprising four types of subprocesses, defined as data movement types occurring during functional process execution—that is, entry, exit, read, and write data movement types.

IFPUG FPA measures a transactional function’s size on the basis of its type as well as its number of DETs and file types referenced (FTR).

In MkII FPA, an LT’s functional size is the sum of the size of the input, processing, and output components. The input and output components’ sizes are proportional to the number of uniquely processed DETs crossing into and out of the application boundary, respectively. The processing component’s size is proportional to the number of primary data entity types referenced during LT execution.

In Cosmic FP, a functional process’s size is proportional to the number of entry, exit, read, and write data movements. Cosmic FP measures not at the granularity level of the DETs’ movements but rather at the level of data group movements.

Regarding the three FSM methods’ measurement scales, all use absolute scale to get the base counts. After IFPUG FPA obtains the base counts, it uses an ordinal scale to assign functional size values to each BFC type. MkII FPA assigns weights to the base counts, and Cosmic FP assigns one unit of measure to each BFC type’s base counts and uses a ratio scale.

**The Unified Model**

To define the UM, we first unify the FSM methods with respect to their least common denominator at the first phase of the measurement.

At this phase, when the base count is obtained, not only most of the measurement concepts but also the measurement scales are common. We use the set operations to define and relate the sets of base counts for each method. Then, the concepts and measurement rules specific to each method are reflected to the UM.

In Figure 2a, we show how the UM considers different types of data groups (DGs in the figure). With respect to the application boundary, the application might maintain a data group inside (the set shown in the tan oval) or the data group might lie outside (the set outside the tan oval). A data group might be primary (the red dots) or nonprimary (blue diamonds) for the application being measured and could have subgroups (as shown in the dashed circles).

In the UM, a data type (DET or data group) might exist on an I/O device at the front end and in persistent storage at the back end (as in Cosmic FP, we don’t differentiate volatile storage from persistent storage).

For the UM, we set general rules for the data...
types (see Figure 2b). Then, we assigned UM data type concepts to the specific methods. Figure 2c shows the mapping examples of some IFPUG and UM data type concepts.

In the UM, a transaction consists of an input part, a process part, and an output part. The input part provides functionality for carrying user inputs across an application boundary. It has two elements: input DETs across the application boundary (“read DETs from I/O device”) and output data groups that hold these input DETs (“read DG from I/O device”).

The process part provides functionality for processing data during transaction execution. It has two subtypes: read and maintain. The read subprocess has one element, “read DG from PS,” and the maintain subprocess has one element, “write DG to PS.” Because no method we examined in this study defines explicit rules for measuring “information creation type” processing’s functional size, we didn’t include it in the UM. These subtypes manipulate and compute new data, such as simple Boolean operations or complex mathematical algorithms.

The output part provides functionality for formatting and sending data across the application boundary. It also has two elements: output DETs across an application boundary (“write DETs to I/O device”) and output data groups that hold these output DETs (“write DG to I/O device”).

We assigned UM transaction concepts to each method. Figure 2d shows examples of how some IFPUG transaction concepts map to UM.

On the basis of these assignments, we define the measurement functions for each method in terms of UM concepts.

**UM Restrictions**

The model we developed has certain restrictions. For one thing, it doesn’t handle the detailed rules each method suggests (for example, the IFPUG method specifies, “don’t count code tables,” whereas the other methods don’t have this restriction).

In addition, the UM calculates sizes as unadjusted function points and doesn’t implement value adjustment factors. The UM’s other constraint is that although it addresses unifying methods to measure the sizes of “whole pieces” of software (such as the work product of a new software development project), it can’t measure the sizes of changes made to an existing piece of software (as in an enhancement project).

Finally, the UM isn’t a substitute for conventional FSM methods because its measurement process and detailed measurement guidelines aren’t separately described.

**Case Studies**

We conducted two case studies (projects) to evaluate the proposed UM. We first measured each project’s functional size using the conventional approach—that is, by applying FSM method guidelines. We then remeasured the functional sizes by implementing the UM.

For conventional measurement, we used Microsoft Excel to collect the data. To apply the UM, we developed a software tool that implements the rules defined in the UM. We entered a single set of data to the tool, as Figure 3 (on the next page) shows. The tool then provided detailed measurement reports for each FSM method.

**Case Study 1**

The first case study was a Web-based, military inventory-management project integrated with a document management system. We determined the case project’s functional domain to be an “information system,” using the CHAR method.

The project ran from October 2004 to December 2005. It used 7,493 person-hours. The...
The project staff consisted of six people, three of whom worked part time. The project used the Internal Development Framework and Java as programming languages, IBM WebSphere Application Developer as its development environment, Rational Rose as an analysis and design tool, Oracle 9i as its database management system, and Tomcat as its application server.

The company uses a Software Requirements Specification (SRS) standard it developed; we performed size measurements on the basis of the project’s SRS documents.

Two people performed the FSM: one from the development organization involved in the project and one of the authors of this study. Both measurers were experienced in using the methods. To avoid measurers’ different assumptions, the same people conducting the conventional measurement also performed the UM’s FSM.

Because we already identified the transactions, data groups, and DETs during conventional measurement, we used these figures during UM implementation.

We measured case study 1’s functional size using IFPUG FPA, MkII FPA, and Cosmic FP, successively (see Tables 1, 2, and 3).

Case Study 2

The second case study involved two subsystems (projects 2a and 2b in the tables) in an avionics management system for small- to medium-size commercial aircraft on a flight display system. We determined the project’s functional domain to be a “complex data-driven control system,” using the CHAR method.

The project ran from November 2003 to September 2005, and the project staff consisted of 15 people. It used Telelogic Doors for software requirements analysis, Rhapsody for object-oriented software design, and Visual Studio C++ for software coding.

In this case study, we implemented the UM to remeasure the functional size of the development project’s two subsystems, which had already been measured by MkII FPA and Cosmic FP methods. We also measured the IFPUG functional size by entering the additional data required. We performed this case study to observe how the UM would behave when measuring subsystems from another functional domain.

Table 1 shows the FSM details for both subsystems in case study 2.

Results and Discussion

Our case studies determined that projects’ functional sizes are the same whether we measure them using the conventional approach or the UM.

When we first performed the measurements, we had slightly different results for conventional versus UM measurements. That is, we had an error rate of roughly 0.4 percent for the first case project and 0.75 percent for the second.

We checked the measurement details and observed that we’d made some manual counting errors during the conventional measurement as well as some mistakes in reflecting the changes in the FURs to the measurement. For example, owing to a change in the FUR that a customer requested, a DET no longer existed. However, we hadn’t noticed this and so hadn’t removed the DET from some of the functional processes. We identified and corrected 27 errors in total.

Our case study results show that the UM worked properly to measure software products’ functional sizes when implementing all three FSM methods simultaneously.

Because the UM collected all the BFC types in a unified way, we had only one set of data for all measurements. When a project had a requirements
change, we could easily identify it and make the necessary changes without having to change it in every measurement method. So, the UM decreased the possibility of making mistakes during multiple method measurements.

An interesting result of the case studies is that, when we compare the projects’ sizes via different methods, project 2b is larger than project 1 when measured via IFPUG FPA and Cosmic FP and nearly equal when measured via MkII FPA. We can attribute this to numerous factors—for example, MkII FPA uses industrial weights to calculate size. Also, IFPUG FPA’s scale has upper and lower size limits for BFC types, whereas MkII FPA and Cosmic FP have no such limits. Such factors make defining solid statistically based conversion formulas difficult. Because the UM conducts unification at base-count levels, it reflects the methods’ differences when applying their measurement functions. So, conventional conversion techniques’ restrictions don’t affect the UM.

One constraint in our case studies was that measurements were performed by the same people relying on the same assumptions. Another constraint was related to the number of case studies performed. Further case studies might let us identify the model’s behavior in exceptional cases.

On the basis of results of our studies, we think practitioners and researchers might use the UM in several ways:

- **Simultaneous measurement.** The model enables simultaneous measurement by one, two, or all three methods. Organizations that must...
Cigdem Gencel is an assistant professor in the Department of Systems and Software Engineering at the Blekinge Institute of Technology. Her work focuses on software measurement and metrics, software size measurement, software effort and cost estimation, software project management, software process improvement, and software requirements elicitation. Gencel has a PhD in software functional size measurement from the Middle East Technical University. Contact her at cigdem.gencel@bth.se.

Onur Demirors is an associate professor and the chair of the Software Management Program in the Information Systems Department at Middle East Technical University, as well as the strategy director of Bilgi Grubu. His work focuses on software process improvement, software project management, software measurement, software engineering education, software engineering standards, and organizational change management. Demirors has a PhD in computer science from Southern Methodist University. Contact him at demirors@metu.edu.tr.

Gencel has a PhD in software functional size measurement from the Middle East Technical University. Her work focuses on software measurement processes and rules. We've already initiated further studies on the concept to improve the model and overcome its restrictions, extend the rules of the methods for specific situations, such as sizing changes, and implement the methods' approximation techniques.

Acknowledgments

We thank Pinar Efe for her contributions in developing the model and performing the case studies, and Nil Korkut for reviewing and proofreading the manuscript. We also thank the Scientific and Technological Research Council of Turkey (TUBITAK), project 107E010, for its support.

References


For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.