FUNCTIONAL SIMILARITY IMPACT ON THE RELATION BETWEEN FUNCTIONAL SIZE AND SOFTWARE DEVELOPMENT EFFORT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZDEN ÖZCAN TOP

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2008

Approval of the Graduate School of Informatics

Prof. Dr. Nazife Baykal
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Yasemin Yardımcı
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Onur Demirörs
Supervisor

Examining Committee Members

Prof. Dr. Semih Bilgen	(METU, EE)	
Assoc. Prof. Dr. Onur Demirörs	(METU, IS)	
Assoc. Prof. Dr Ali Doğru	(METU, CENG)	
Dr. Altan Koçyiğit	(METU, IS)	
Prof. Dr.Hayri Sever	(HU, CENG)	

ABSTRACT

FUNCTIONAL SIMILARITY IMPACT ON THE RELATION BETWEEN FUNCTIONAL SIZE AND SOFTWARE DEVELOPMENT EFFORT

ÖZCAN TOP, Özden
M.S., Department of Information Systems
Supervisor: Assoc. Prof. Dr. Onur DEMİRÖRS

September 2008, 78 pages

In this study, we identified one of the reasons of the low correlation between functional size and development effort which is overlooking the similarity of the functions during the mapping of the functional size and development effort. We developed a methodology (SiRFuS) that is based on the idea of the reuse of the similar functions internally to provide high correlation between functional size and development effort.

The method is developed for the identification of the similar functions based on the method of Santillo and Abran. Similarity percentages among the functional processes and Similarity Reflective Functional Sizes are computed to attain adjusted functional sizes. The similarity reflective functional sizes were named as Discrete Similarity Reflective Functional Size and Continuous Similarity Reflective Functional Size based on the characteristics of the adjusted functional sizes. The SiRFuS method consists of three stages: measurement of the software product with COSMIC Functional Size Measurement (FSM) method; identification of the functional similarities bases on the measurement results and calculation of the similarity reflective functional sizes.

In order to facilitate the detection of similar functions, calculation of the percentage of the similarities and similarity reflective functional sizes; a software tool is developed based on the SiRFuS method.

Two case studies were performed in order to identify the improvement opportunities and evaluate the applicability of the method and the tool.

Keywords: Functional Size Measurement, Software Development Effort and Functional Size Relation, Functional Similarity

ÖZ

FONKSİYONEL BENZERLİKLERİN FONKSİYONEL BÜYÜKLÜK VE YAZILIM GELİŞTİRME İŞGÜCÜ ARASINDAKİ İLİŞKİYE OLAN ETKİSİ

ÖZCAN TOP, Özden
Yüksek Lisans, Bilişim Sistemleri
Tez Yöneticisi: Doç. Dr. Onur DEMİRÖRS

Eylül 2008, 78 sayfa

Bu çalışmada, fonksiyonel büyüklük ile yazılım geliştirme işgücü arasındaki ilişkinin düşük olmasının nedenlerinden birinin bu ilişki oluşturulurken benzer fonksiyonların göz ardı edilmesi olduğunu belirledik. Benzer fonksiyonların aynı ürün içerisinde tekrar kullanılma fikrinden yola çıkarak fonksiyonel büyüklük ile işgücü arasındaki ilişkinin yüksek olmasını sağlayacak bir yöntem geliştirdik (SiRFuS).

Yöntem, Santillo ve Abran'ın yaklaşımından yola çıkarak ürün içerisindeki fonksiyonel benzerliklerin ve benzerlik yüzdelerinin belirlenmesi ve Benzerlik Etkisindeki Fonksiyonel Büyüklüklerin hesaplanarak yazılım işgücü ve fonksiyonel büyüklük arasındaki ilişkiyi güçlendirecek uyarlanmış bir büyüklük elde etmek amacıyla geliştirilmiştir. Benzerlik etkisindeki fonksiyonel büyüklükler, uyarlama yaklaşımının özelliklerine göre “Ayrık Benzerlik Etkili Fonksiyonel Büyüklük” ve “Devamlı Etkili Fonksiyonel Büyüklük” olarak adlandırılmıştır. SiRFuS yöntemi üç aşamadan oluşmaktadır: Yazılım ürününün COSMIC Fonksiyonel Büyüklük Ölçüm (FBÖ) yöntemi ile belirlenmesi, ölçüm sonuçlarından fonksiyonel benzerliklerin elde edilmesi ve benzerlik etkisindeki fonksiyonel büyüklüklerin hesaplanması.

Benzer fonksiyonların bulunarak fonksiyonel süreçler arasındaki benzerliklerin yüzdelerinin belirlenmesi ve benzerlik etkisindeki uyarlanmış fonksiyonel büyüklüklerin bulunmasını kolaylaştırmak için bir araç geliştirilmiştir.

Gelişim fırsatlarını belirleyebilmek ve yöntemin ve aracın uygulanabilirliğini değerlendirebilmek için iki durum çalışması yapılmıştır.

Anahtar Kelimeler: Fonksiyonel Büyüklük Ölçümü, Yazılım Geliştirme İşgücü ve Fonksiyonel Büyüklük İlişkisi, Fonksiyonel Benzerlikler

To my beloved Can Barış Top

&

to the Great Memory

of

my Grandmother

Ayşe Güroğlu

ACKNOWLEDGMENTS

I would like to express my special thanks to Assoc. Prof. Dr. Onur Demirörs, for his great support, enlightening ideas, criticism and insight. He was always patient and kind to me throughout the research. I am literally saying that, he was one of the most important powers of my finalization this study by encouraging me and providing me to believe in myself.

I am grateful to Dr. Oktay Türetken for his valuable ideas on functional size measurement. I am doubtful if I could understand this concept so well without his enlightening thoughts. He always answered my questions and didn't hesitate to explain the concepts when I was confused.

I would like to thank my friend Barış Özkan for his ideas and suggestions on functional size measurement methods. Special thanks to Dr. Çiğdem Gencel for the literature repository on functional size measurement. And thanks to Seçkin Tunalılar for the ideas on functional similarity concept.

I am grateful to Elif Urgun, Emre Ergüden and Onur Şentürk, since they provided me the case products and measurement results that were used in this study. Elif and Onur answered my question through the emails despite the density of their work. Thanks to Farid Khalikov for his support on counting LOCs of case products.

Hearthfelt thanks to my colleague Selim Nar who has allocated his limited time to fix the structure of my thesis.

I would like to thank my friends Hüseyin Küçükcığa and Devran Dönmez for their valuable friendships, and considerations.

Finally, very special gratitudes go to my mother Münevver Özcan, my father Mustafa Özcan, and my sister Zeynep Özcan, for their love and support throughout all my life.

AND my spouse Can Barış, who had provided me very valuable suggestions during the thesis study. It was not possible to accomplish the analysis of the huge amounts of data without his technical support. Besides, I always felt his love and support with me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AN	: Name of the Case Product
ASR	: Average Similarity Affected Functional Size
BFC	: Base Functional Component
BN	: Name of the Case Product
CFP	: COSMIC Function Point
Cfsu	: COSMIC functional size unit
COSMIC	: Common Software International Consortium
CN	: Name of the Case Product
CS	: Continuous Functional Size
DG	: Data Group
DM	: Data Movement
DS	: Discrete Functional Size
FSM	: Functional Size Measurement
FP	: Functional Process
FUR	: Functional User Requirements
KM	: Name of the Case Product
OOI	: Object of Interest
PS	: Plain Functional Size
SiRFuS	: Similarity Reflective Functional Size Method
SM	: Software Management
SN	: Name of the Case Product
TN	: Name of the Case Product
WBS	: Work Breakdown Structure

CHAPTER 1

INTRODUCTION

Unrealistic estimations are one of the major reasons for software failures (Tucker & Boehm, 2002) and estimation of a software project frequently depends on the software size. In most estimation techniques which use either functional size or length of code as input, it is possible to determine the required effort, cost, and duration to complete a software product with estimation models. That is, by knowing the accurate size of a software product and the relation between size and development effort it is possible to plan, execute and monitor projects successfully. However, there are problems with the identification of these sizes. Although, source of lines of code (SLOC) is commonly used and easy to measure, it is not probable to attain a reliable SLOC value in the early phases of a project (Valerdi, Chen, & Yang, 2004).

On the other hand, since Functional Size Measurement (FSM) methods measure the size by identifying the functionalities provided to the user; they are appropriate to be used in the beginning of the projects. Although, the functional size of a software product greatly relies on the assumptions and interpretations of the measurer; contributions of the functional size on effective project management are deeply investigated by (Ozkan, Turetken, & Demirors, 2008) in their research. As Ozkan (2008) emphasized, functional size is a base for project integration management, since it can be identified in the early stages of the software life cycle. It is also a base for scope, time and cost management since the functional size of each activity is known with the decomposition of the Functional User Requirements (FURs) into

Base Functional Components (BFCs) which refer to the tasks within the work packages of the work breakdown structure (WBS) of the projects.

The opportunities of using functional size as input for project management given above are an indicator of the importance of functional size as software measure. Therefore, the functional size should be determined precisely. Although the functional size of a software product can be measured with current methods, there are still difficulties in measuring the functional size and the relation between the functional size and the required effort (Gencel & Demirors, 2008). Besides the structural weaknesses of FSM methods, one of these difficulties is related with the functional similarity of functional processes which enlarges the functional size of the product, leading to unrealistic effort and cost estimation.

The functional similarity concept can be best explained by an example. Assuming that after the decomposition of a functional user requirement, two functional processes were identified as “Constitution of an Entity Model Element” and “Constitution of an Actor Model Element” for the measurement of the size of the product with COSMIC. Although these two functional processes (FP) have two different objects of interest (OOI) which are “Actor” and “Entity”; the OOIs are composed of exactly the same data groups (DG) such as “general information, source of information, properties, and relationships”. Since FPs have same data groups, they consist of same attributes as expected. Therefore, the data movements which are the determiner of the functional size of the product, input to (Entry), read from (Read), write to (Write) and represents (eXit) same data groups . In addition to these, they are constituted by using the same screens. Because of these reasons, they are 100% similar functional processes and good candidates for internal reuse.

The similar functions can be reused within the same or a new software product, since very similar software issues are required for the development of these functions. Our research focuses on the reuse of similar functions; however, whether the reusability of these similar functions is probable and effective is not within the scope of this study.

The software reuse concept which can be described as the creation of software systems based on the existing software, improves the productivity and software quality, and it has been subject to many researches since 1968 (Krueger, 1992), (Frakes & Terry, 1996). In addition to source code; specifications, design structures, test data and documentation can also be reused (Krueger, 1992).

Reuse is named “internal”, when an object, module or procedure created for a system is used multiple times within the same system and “external” when a module from a different system is used one or more times within a new system (Banker, Kauffman, & Zweig, 1993). Organizations usually take into account the functional similarity effect on effort in terms of external reuse. However, as external reuse, internal reuse has a significant impact on the total required effort, time, and cost.

Although software products have functionally similar modules or similar functional processes, it is not always easy to determine functionally similar software entities/processes especially at the beginning of the projects. Moreover it is not clear whether the functionally similar entities require exactly the same effort for the development or not, and what the impact of the similarity on the development effort should be (Ozcan Top, Tunalilar, & Demirors, 2008).

In the literature, there are a few approaches to determine functional similarities. One of the approaches which is also the subject of this study is functional reusability approach which determines the similarities among functional processes by assessing data groups and data movement action types on the products measured by COSMIC (Santillo & Abran, 2006). The other approach consists of the entity generalization/specialization practices which are widely used in object oriented methodologies and can be used in the grouping of the similar functional processes into one and as a result can be used to eliminate the replication of the same/similar functions (Turetken, Demirors, Ozcan Top, & Ozkan, 2008).

In this study, identified the functional similarities among functional processes which later can be internally reused and developed a methodology which provides us to reach more reliable equivalent functional size values correlated with software development effort.

The methodology developed, SiRFuS, has three stages: the first one is the measurement of the software by using COSMIC method which is one of the commonly used functional size measurement methods; the second stage is the identification of the functional similarities among the functional processes from the measurement data set; and the last stage is the calculation of the similarity reflective functional sizes: Discrete Similarity Reflective Functional Size (DS) and Continuous Similarity Reflective Functional Size (CS).

The discrete similarity reflective functional sizes (DS) are calculated by using constant functional similarity percentage values which change depending on five conditions. Besides the conditions, constant similarity values correspond to the conditions are determined by analyzing NESMA's reuse approach for enhancement projects (NESMA, 2001). Based on the constant similarity intervals that the highest functional similarity of a functional process corresponded, the DS is calculated.

On the other hand, the continuous similarity reflective functional sizes (CS) are calculated by using continuous functional similarity percentage values which are derived from the functional similarity matrix of the products. Based on the identified similarity values and the formulas the CS is calculated.

We performed two case studies as a part of this thesis study with the research objectives given below:

- to assess the functional similarity identification process of the current COSMIC based methods.
- to evaluate how the current COSMIC based functional similarity determination methods impact on the relation between functional size and the total effort.

- to determine the difficulties and the improvement opportunities of COSMIC based functional similarity identification methods with the case studies.

- to develop a method which provides high correlation between functional size and effort based on the research results.

- to evaluate the applicability of the methodology developed.

The first case study is a single-case study which was conducted to identify the problems of the current functional similarity identification approaches and bring into light the improvement opportunities related to the functional similarity identification approaches. The case product, KN (Karagöz, 2008), was chosen as the single case study, since it included so many similar functional processes, data entities and attributes that it can be described as a challenging application from the planning perspective.

The second case study is a multiple case study which involves eight cases. In this multiple case study, our purpose was to explore the applicability of the SiRFuS method. The case products were selected since they had well documented SRSs and COSMIC functional size measurement results. Another reason for the selection of these cases was the consistency and the accuracy of the collected effort and lines of code values. All of the case products are Information Systems Projects except from AN which is a Complex Data Driven System Project.

The case studies involved the measurement of the case products, (some of which were measured previously as a part of another study as explained in Section 4.2), by using the COSMIC Method, verification of the measurement results, and identification of the functional similarities and the adjusted functional sizes, by using Similarity Reflective Functional Size Method (SiRFuS).

The remainder of this thesis is organized as follows: In Chapter 2, the literature review related with the FSM methods, effort estimation methods and functional similarity identification methods is presented.

In Chapter 3, the method that we developed with the enlightenment of the results of the case study is described. The method is based on the idea that "the identification of the functional similarities within the product and evaluation of them to be reused and finding an equivalent size which provides high relation between effort and size". A detailed example is given to better explain the method at the end of this chapter.

In Chapter 4, the two case studies are explained in detail.

In Chapter 5, the contribution of our research to software project management and the lessons we learned from this study are given. Finally future research suggestions are explained in this section.

CHAPTER 2

LITERATURE SURVEY

This chapter presents a review of literature survey related to size measurement methods, the methods used for identification of functional similarities, and the effort estimation approaches. Although size of a software product can be measured by using various measures such as functionality measure, length measure, and object measure (Gencel, 2005); the scope of the related research consists of the functional size measurement methods since this thesis is related with the identification of functional similarities. Therefore expert judgment method, length of code, objects based estimation methods are out of the scope of this research.

2.1 Related Research on Functional Size Measurement Methods

The idea of measuring size of a software product in terms of its functionality was first introduced by Alan Albrecht in 1979 (Albrecht, 1979). The method is called Function Point Analysis (FPA) and has gained a considerable interest because it focuses on measuring the size from user perspective independent of the application itself.

Based on Albrecht's method several measurement techniques have been developed, each of which aims at the extension of the applicability of the techniques in different functional domains. Due to the proliferation of the techniques, the ISO/IEC workgroup has been initiated to identify fundamental concepts and to establish an international standard for functional size measurement (ISO/IEC, 1998), (ISO/IEC, 2002a), (ISO/IEC, 2003a), (ISO/IEC, 2002b), (ISO/IEC, 2004), (ISO/IEC, 2005a).

Today, IFPUG FPA (ISO/IEC, 2003c), Mark II (ISO/IEC, 2002c), COSMIC FSM (ISO/IEC, 2003b), NESMA FSM (ISO/IEC, 2005b) and FISMA (ISO/IEC, 2008) are accepted as international standards for functional size measurement by ISO/IEC. All these methods measure the functionality from the perspective of the functionality provided to the user; however, they use different units and rules during measurement.

This thesis study is based on the measurement of the products by COSMIC Functional Size Measurement (FSM) Method (ISO/IEC, 2003b). Therefore, the details of COSMIC are explained in the following paragraphs.

Since the COSMIC Functional Size Measurement Method was first introduced in 1999, it has been improved in time and new versions have been released.

The COSMIC Method is applicable in Business Application Software such as human resources management system or banking system; it is applicable in Real-Time Software such as the software embedded in devices like computers or telephones and the hybrids of these two domains such as airline and hotel reservation systems. However, it is not applicable for the measurement of the algorithmic complex systems, self-learning systems, simulation systems (ISO/IEC, 2003b).

In COSMIC v3.0, the functional size is measured from the "Functional User" viewpoint instead of the "End User" or "Developer" viewpoints introduced in COSMIC v2.2 since all size measurements are functionalities provided to the users. Another improvement has been made on the unit of the functional size. It has been changed from Cfsu (COSMIC functional size unit) to CFP (COSMIC Function Point) with the latest version.

The measurement process begins with the identification of the purpose and the scope of the measurement and the extraction of the Functional User Requirements (FURs) from the artifacts of software to be measured. In addition to these "functional users" and the "levels of granularity" should be determined in the beginning of the measurement.

In the mapping phase, with the purpose and scope of the measurement, “functional processes” are determined by decomposing the FURs.

A functional process (FP) is described as “an elementary component of a set of Functional User Requirements comprising a unique, cohesive and independently executable set of data movements. It should be triggered by an Entry a functional user that informs the piece of software that the functional user has identified a triggering event. It is totally complete when it has executed” (ISO/IEC, 2003b).

The next phase is the identification of the Object of Interest (OOI) and Data Groups (DG). OOIs can be any “entity” which is related with FURs, on the other hand Data Group is a distinct, non empty, non ordered and non redundant group of attributes related with one OOI.

Lastly in the measurement phase, functional processes are decomposed into the sub-processes known as data movements and data manipulations. Data Movements are the Entries, eXits, Reads and Writes crossing the boundary between the functional user and the application measured by moving the data groups (ISO/IEC, 2003b). On the other hand, Data Manipulations are not separately measured in the scope of the measurement since they have already been associated with one of the data movements and counted within them.

The data movements are described as follows:

- An Entry moves a data group from a functional user across the boundary into the functional process where it is required. It may have one to several data attributes.
- An eXit moves a data group from a functional process across the boundary to the functional user that requires it.
- A Read moves a data group from persistent storage within reach of the functional process which requires it.
- A Write moves a data group lying inside a functional process to persistent storage.

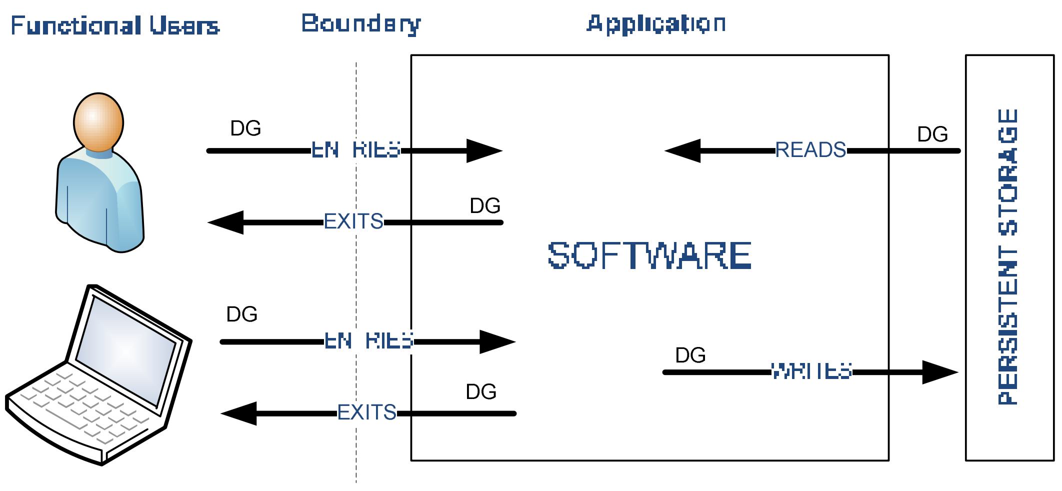

Figure 1 Symbolic Demonstration of Data Movements

The result of the measurement of a software product is calculated by aggregating the number of data manipulations.

$$SizeCFP(functional\ process_i) = \sum size(Entries_i) + \sum size(Exits_i) + \sum size(Reads_i) + \sum size(Writes_i)$$

2.2. Related Research on Functional Similarity Methods

In terms of functionality, similarities on software applications have been subject to research projects and defined by using different terminologies. Fenton defined a concept called “private reuse” as the extent to which modules within a product are reused within the same product (Fenton, 1991). Cruickshank and Gaffney also make first distinction for “internal” and “external” reuse in the literature from economical perspective (Cruickshank & Gaffney, 1992).

Whatever the terminology is, the functional similarity concept has a significant effect on all phases of the life-cycle of the projects. For example, the effect of functional similarities and reusability in maintenance was evaluated by Abran and Desharnais in (A. Abran & Desharnais, 1995). They have developed an approach for the identification and measurement of reuse in the enhancement projects by considering Function Point Analysis Method. Their approach depends on two key concepts: reuse indicator and predictor ratio. The study depicts how an alternative size measure can be obtained by combining predictive ratio and reuse indicator.

Functional similarity has been subject to one of the common functional size measurement methods, COSMIC. It defines the functional similarity concept in its Guideline for Sizing Business Applications document (Consortium, 2005). It is stated that developers might avoid duplications by realizing the functional reuse opportunities among functional processes; however, the user point of view ignores the functional similarities since the FURs are measured independently instead of grouping similar functional processes.

(A Abran & Maya, 1997) have evaluated similarities within a software product from a functional similarity perspective. They refined and extended the functional similarity measures to create a more precise measurement basis for the cost estimation and productivity models.

In addition to above, in the literature there are considerable numbers of research studies evaluating the software reuse performed at the source code level. However, few of these studies focus on developing methods to identify the functional similarities in the early phases of the software life cycle (Albrecht & Gaffney Jr, 1983), (Leach, 1996). (Ho, Abran, & Oligny, 2000) emphasized the importance of measuring the functional reuse impact in the early phases of the software life cycle rather than coding phase to improve the performance of the software engineering processes. Their work is based on extending the method of (A. Abran & Desharnais, 1995) by using the COSMIC FFP method. The approach proposed in the paper considers only the reuses without modification and called black box approach. The approach utilizes the functional relationships among functional layers.

The development effort and the functional size correlation have been subject to research studies as well. (Meli, 2000) discussed the problems faced during the development effort and functional size correlation. He stated that in some situations, it is possible to aggregate much different logical functionality which leads to rapid and economic implementations with a small amount of working effort. As a result of this, the effort needed to realize the overall system will decrease, and will not be proportional at all to the logical functionalities required.

Only a few research studies focus on methods about the association of the size and the effort considering the functional similarity. Santillo and Della Noce proposed a model named as "Worked Function Model" to achieve a more significant "work size" to be correlated with effort. Model includes "reuse", "replication" and "similarity" adjustments (Santillo & Della Noce, 2005).

In their study Santillo and Abran proposed the approach called "functional similarity" to identify the software reuse from a functional perspective (Santillo & Abran, 2006). The technique is based on uncovering the functional similarities from a data set that comprises functional processes, data movements and data manipulations which are evaluated by using the COSMIC method. Although their study comprises a method sorting out functional similarities, it does not provide an approach for the relation of functional size and development effort.

The functional similarity is described as "if two functions can be identified with the same set of data movements and/or data manipulations, they can be considered as similar functions" by Santillo and Abran (2006).

The method of Santillo and Abran consists of two stages. The first one which is called as "the first order evaluation" compares the functional processes only from data movements' point of view. Similarity among functional processes are determined by comparing the data group and data movement relationships; in addition to this, in some cases where the comparison technique does not suffice, it is suggested that the analyst make her best judgments in order to identify the functional similarities. The second stage, "second order evaluation" determines the functional similarities by considering both data group - data movement and data group - data manipulation action types. Santillo and Abran preferred to evaluate the data manipulations in the "second order evaluation" even if the measurement method does not include the data manipulations into the measurement process.

After the functional similarities are identified, the average, minimum, and maximum similarities are calculated. These calculated data are used for the assessing the potential reuse and decide to apply internal reuse or not.

Lastly, entity abstraction methods are also valid approaches for eliminating the measurement variances based on different point of views, providing abstract data sets by grouping similar functional processes and as a result eliminating the replication of the same functions. Although, they can be evaluated as methods for determining the impact of functional similarities on functional size, they can not be used as a method for identification of the similar functions. A research study considering this approach has been conducted by (Turetken, et al., 2008). They depicted the utilization of entity generalization concept in COSMIC and IFPUG FPA Methods and evaluated the effect of different interpretations on the measurement results.

2.3. Effort Estimation Models

Since accurate effort estimation is one of the most important tasks in software management; various effort estimation models have been developed considering the condition of the project in the software life cycle and management needs. Effort estimation models can be grouped considering various aspects. For instance, top down effort estimation approaches are suitable in the early phases of the software life cycle; whereas bottom up estimation approaches are suitable when each software component is known in detail.

Researchers used different assumptions to classify the effort estimation techniques. Boehm considered the effort estimation techniques in the scope of cost estimation models (BW Boehm, 1981). Cost estimation is determined as the process of estimating the required effort (Leung & Fan, 2002), and these models are used for;

effort estimation, duration estimation, and cost estimation. However, when the main idea is to predict the required effort for software development; we believe that there is a conflict between the name and the function of the methods and they should be named as effort estimation models instead of cost estimation models.

First, Boehm classified the effort estimation methods into seven titles which are Algorithmic Models, Expert Judgment, Analogy, Parkinson, Price-to-Win, Top-Down, and Bottom-Up (BW Boehm, 1981). In his classification, "expert estimation and bottom-up approach" is taken into account as a different approach. However, since analogy techniques work by comparing the current projects with previous ones; expert estimation and bottom-up approach can be considered in the scope of analogy based effort estimation techniques (Jørgensen, Indahl, & Sjøberg, 2003).

Later on, some of the researchers grouped these models under two types: non-algorithmic models and algorithmic models (Leung & Fan, 2002). Algorithmic models are based on mathematical formulas and/or statistical analysis (Leung & Fan, 2002). Boehm emphasized that none of the methods has superiority to another. Besides this, the methods can be used as complementary to each other such as expert judgment and mathematical models, and top-down approach and bottom-up approach (BW Boehm, 1981). Frequently used effort estimation approaches, can be explained briefly as follows:

3.2.1 Expert Judgment

Effort is identified based on the judgments of one or more expert(s) (Anderson, et al., 1999). This approach is suitable when the consultants are familiar with the projects to be developed. New technologies, applications and languages increase the judgment errors. However, Delphi and Wide Delphi Methods are structured approaches to minimize the judgment errors (Demirors, 2008).

3.2.2 Top-Down Effort Estimation

These methods are suitable for the early phases of the software life cycle (Anderson, et al., 1999). Based on the historical information in the organization, and comparing the project with previous similar ones, overall effort for the project is estimated at a high level (Jørgensen, 2004). Later, the effort is distributed over the lower level components considering life-cycle phases. Although top-down approach is easy and fast to implement, it is less accurate when compared to bottom-up approach, since the top down approach requires minimum project data (Anderson, et al., 1999).

Curve Fitting Estimation Models such as COCOMO, SLIM and PRICE-S; which are based on mathematical formulas and statistics; can be considered in the scope of the top-down approach. The details of COCOMO can be found in the following paragraphs.

COCOMO (Constructive Cost Model)

COCOMO 81 is a regression based model. Since it has been published in 1981 by Boehm, it is the most widely used effort estimation model. It is a methodology that allows the user to estimate effort, schedule and cost of the software projects (http://sunset.usc.edu/csse/research/COCOMOII/cocomo81.htm, 2008). The latest version of COCOMO has been published in 2000 with the name of COCOMO II.

There are three different levels of COCOMO 81: Basic, Intermediate, and Detailed. The effort is calculated based on three different difficulty modes of the projects, with Basic COCOMO. This level of COCOMO provides a rough estimation. The difficulty modes are as follows (Horowitz, 1994):

Organic mode is used to calculate effort for small size projects. The development team is familiar with application and language and constraints are not rigid.

Semi-Detached mode is used to calculate effort for the projects in which the constraints are greater than the organic mode. The team is not very familiar with the application to be developed.

Embedded mode is used to calculate effort for relatively large scale projects in which the constraints are rigid.

Based on these difficulty modes above, the formula given below is used with three different variables as in Table 1.

$$Effort = a * \mathrm{Size}^{b}$$

Table 1 Variables of Basic and Intermediate COCOMO formulas

	Basic		**Intermediate**	
Mode	**a**	**b**	**a**	**b**
Organic	2.4	1.05	3.2	1.05
Semi-Detached	3.0	1.12	3.0	1.12
Embedded	3.6	1.20	2.8	1.20

Intermediate COCOMO 81 uses the formula given above and takes into account 15 cost factors which are classified into four categories (Horowitz, 1994):

Product Attributes are the characteristics of the product to be developed which are reliability, database size, and product complexity. *Computer Attributes* are constrains on software implied by the hardware platform: The four attributes in this category are execution time constraints, main storage constraints, virtual machine volatility, and computer turnaround time. *Personnel Attributes* describe the qualification and experience of the development team. The five attributes in this category include: Analyst capability, applications experience, programmer capability, programming language experience, and virtual machine experience. *Project Attributes* include use of modern programming practices, use of software tools, and required development schedule.

These attributes which affect the effort are rated from "very low" to "extremely high" and the scales are aggregated. The result of this scale which is "Effort Adjustment Factor (EAF)" is multiplied with the effort.

Detailed COCOMO 81 includes some additional steps. First of all, the software product is decomposed into sub-components and cost drivers are evaluated for each component separately (Demirors, 2008). Detailed COCOMO uses different effort multipliers for each phase of a project (Masse, 1997). Masse emphases that although detailed model increases the predictability of efforts by considering each phase of the development life cycle, it is not robust enough to predict efforts at all phases of the development accurately. Because, the inputs of the later phases, such as design and coding, can not be estimated reliably in the early phases.

Since COCOMO is one of the most used effort estimation method, its ineffectiveness has been subject to many researches. Kemerer has investigated the accuracy of the effort estimation techniques including Basic COCOMO and Intermediate COCOMO (Kemerer, 1987). He performed his study in 15 case products and published that when the actual and estimated effort values compared, average percentage error is 610.1% and 583.8% for Basic COCOMO and Intermediate COCOMO respectively.

COCOMO II

COCOMO II was developed, to resolve the accuracy problem defined above, with the consortium of organizations and graduate students during 1990s and first released in 1996 (B. Boehm, Abts, Horowitz, & Madachy, 2000). The new method depends on three major steps and it supports software development models other than Waterfall model (Anderson, et al., 1999).

In stage 1, which is called as Application Composition, object point method is used for estimation of software size. This stage supports prototyping to identify risky issues and includes productivity rating. Developer's capability and experience are taken into account as an impact of effort required for software development.

The second stage, which is called Early Design, supports the measurement of the software in the early phase by using function point or source of lines of code measures. Since the SLOC is input to the model, function point results are converted to SLOC by a conversion table. In this stage, seven cost factors evaluated are: Product Reliability, Complexity, Required Reuse, Platform Difficulty, Personnel Capability, Personnel Experience, and Facilities and Required Development Schedule (Anderson, et al., 1999).

In stage 3, additional seventeen effort multipliers are evaluated some of which are the extension of the attributes in Intermediate COCOMO 81 (Dillibabu &

Krishnaiah, 2005), (Anderson, et al., 1999). These cost attributes are as follows: Required Reliability, Database Size, Product Complexity, Required Reusability, Documentation Required, Execution Time Constraints, Main Storage Constraint, Platform Volatility, Analyst Capability, Applications Experience, Programmer Capability, Personnel Continuity, Platform Experience, Language and Tools Experience, Use of Software Tools, Multiple Site Development, Required Development Schedule.

In the calculation of the required effort for software development, the cost attributes above, and the scale factors, which are Precedentedness, Development Flexibility, Architecture / Risk Resolution, Team Cohesion, Process Maturity, are taken into account as variables in the formula given below (Anderson, et al., 1999).

$$\boldsymbol{PM = [A * (Size^{\sim})^{B} * EM] + PM_{AT}}$$

where, PM is estimated effort in person months. Coefficient A can be set conditionally based on the organization’s culture.

$Size^{\sim}$ = Size (1 + BRAK/100) where BRAK is the percentage of code thrown away due to requirements volatility. Size is the sum of new and adapted KSLOC.

B = 0.91 + 0.01 (SF), where SF is the sum of five scale factors that vary between 0 and 5. EM is the impact of product of 17 effort multipliers. PM_{AT} is the effort for components automatically translated (Anderson, et al., 1999).

3.2.3 Bottom-Up Effort Estimation

To be able to use bottom-up estimation, each task in the work break down structure of the project should be well known, and historical data that involves productivity should be reliable. Since detailed information about the requirements and tasks are required to use this method, it is not suitable in the early phases. When the detail level of the requirements is suitable to use the method, the size of each task or component is estimated, and the required effort is calculated using historical productivity of the organization or the team (Demirors, 2008). The method is sufficiently reliable when the productivity of the team is consistent; however, it requires too much time to calculate (Anderson, et al., 1999) Therefore, it can be perceived as a time consuming process.

CHAPTER 3

SIMILARITY REFLECTIVE FUNCTIONAL SIZE: A SYHTHESIS METHOD TO RELATE EFFORT AND FUNCTIONAL SIZE

In this chapter, the structure and the process of the similarity reflective functional size calculation method and the tool developed to automate the method - Similarity Reflective Functional Size Measurement Tool, abbreviated as “SR Tool” - are explained in detail. A full example which explains how the method is applied is given in the last section of this chapter.

3.1 Similarity Reflective Functional Size (SiRFuS)

As discussed in the literature review section, (Santillo & Abran, 2006) defined a method which is used to identify the functional similarities within the products measured by COSMIC. SiRFuS is based on the approach of Santillo and Abran and extends its applicability.

SiRFuS consists of three stages. The first one is the measurement of the functional size of the product with COSMIC. The second one is the identification of the functional similarities within the product by comparing the data group data movement couples. The third one is the determination of the similarity reflective sizes which can be either discrete or continuous. The flow of the process is given on Figure 2.

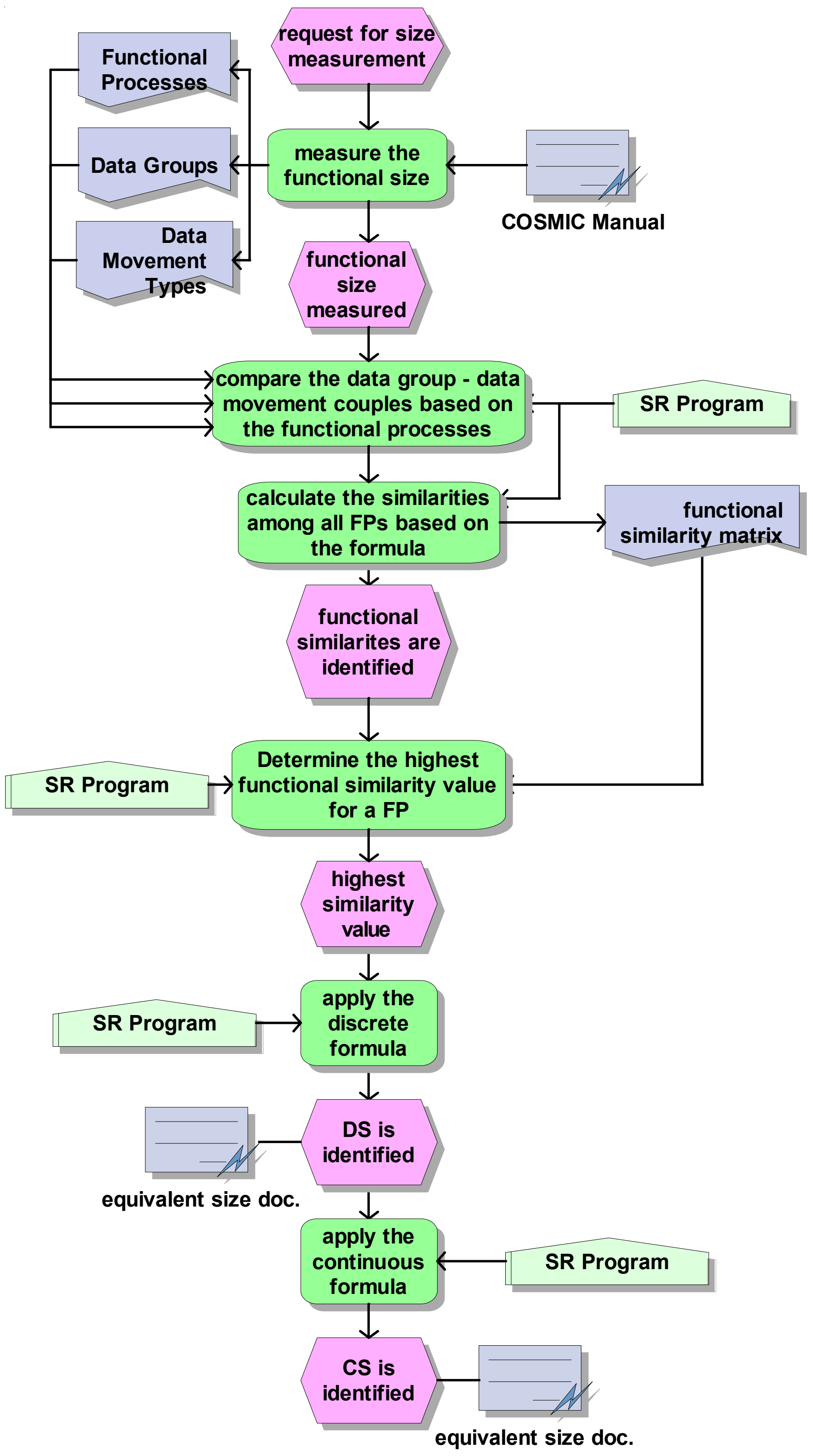
request for size measurement
Functional Processes
measure the functional size
Data Groups
COSMIC Manual
Data Movement Types
functional size measured
compare the data group - data movement couples based on the functional processes
SR Program
calculate the similarities among all FPs based on the formula
functional similarity matrix
functional similarites are identified
SR Program
Determine the highest functional similarity value for a FP
highest similarity value
SR Program
apply the discrete formula
DS is identified
equivalent size doc.
apply the continuous formula
SR Program
CS is identified
equivalent size doc.

Figure 2 SiRFuS Process Flow

3.1.1. Measurement of the Functional Size of the Product

According to the structure of the COSMIC method, functional user requirements (FURs) can be decomposed into the functional processes which are consisted of sub-processes known as data movements and data manipulations. Data Movements are the Entries, eXits, Reads and Writes crossing the boundary of software by moving data groups. Data groups are consisted of specific set data attributes which may belong to a single object of interest (ISO/IEC, 2003b). On the other hand, Data Manipulations are not separately measured in the scope of the measurement since they have already been associated with one of the data movements and counted within them.

During the measurement of the product, the measurer should use a specific measurement format which is an Excel table consisting of functional process name, data movement type, the number of the data movement, and data group name cells respectively. The names within the cells should be written without any space for the data to be used by the tool developed as part of this study. The format of the measurement table can be seen on Table 2. Another constraint related with the measurement process and the tool is the naming of the data groups: the measurer should use the same name for the data groups which are consisted of the same attributes. This kind of notation will provide the integrity of the measurement and the tool to distinguish similar functions.

3.1.2. Functional Similarity Identification Process

We described the similarity of the two functional processes as “If two functions contain common data movement (DM) and data group (DG) tuples, they can be considered as similar functions”. Therefore in the second stage of the method, the similarity of two functional processes is determined by comparing the data group data movement couples within the functional processes. An example set of the comparison data are given on Table 2. Same data movement and data group couples are marked with the same color.

Table 2 Example of Comparison Data

Functional Process	Data Movement	Number	Data Group
FunctionalProcessA	Entry	1	DataGroupA
	Read	2	DataGroupB
	Exit	3	DataGroupD
	Read	4	DataGroupD
	Exit	5	DataGroupC
FunctionalProcessB	Entry	6	DataGroupA
	Read	7	DataGroupB
	Exit	8	DataGroupD
	Exit	9	DataGroupE

Functional size of a process is a set of Data Movement and Data Group Tuples. The number of tuples in the Functional Process set is the functional size of the process in COSMIC. The similarity of the two functions can be defined formally as follows: The examples of these conditions can be found in section 3.1.4.

(A) SimilarFunctionalProcesses ==

$\{fpp: functional\ process\ pair \mid \exists\ A, B: functional\ processes\ \cdot A \in fpp\ \wedge B \in fpp\ \wedge \exists\ dmdg: (data\ movement, datagroup) \cdot dmdg\ in\ B \in A\}$

(B) 100%SimilarFunctionalProcesses ==

$\{fpp: functional\ process\ pair \mid \exists\ A, B: functional\ processes\ \cdot A \in fpp\ \wedge B \in fpp\ \wedge\ \forall\ dmdg: (data\ movement, datagroup) \cdot dmdg\ in\ B \in A\ \wedge \forall\ dmdg: (data\ movement, datagroup) \cdot dmdg\ in\ A \in B\ \}$

(C) Non-SimilarFunctionalProcesses ==

$\{fpp: functional\ process\ pair \mid \exists\ A, B: functional\ processes\ \cdot A \in fpp\ \wedge B \in fpp\ \wedge \forall\ dmdg: (data\ movement, datagroup) \cdot dmdg\ in\ B \notin A\}$

When the functional process A and the functional process B in Table 2 are compared, we reach the following results:

Functional size of A is 5 CFP where as functional size of B is 4 CFP. The three data movements within the functional process B are exactly the same as those within the functional process A. Therefore, B is % 60 functionally similar to A, which means B can use 60 % of the data movements of A and A is 75 % functionally similar to B which means A can use 75 % of the data movements of B.

Based on the approach described above, we identify the functional similarities manually which requires comparison of all the functional processes with each other in terms of data movement and data group couples. We observed the magnitude and complexity of this task when the comparison process was started and realized that a tool will provide significant improvement for the accuracy and required effort.

We developed the Similarity Reflective Functional Size Measurement Tool, abbreviated as "SR Tool" by using the MATLAB (MathWorks, 2007), which automatically compares the functional processes. The tool takes the COSMIC measurement results as input and presents a functional similarity matrix, and results of the similarity reflective sizes at the end of the process. The format of the measurement results which are input to the tool should have been formatted manually as given on Table 2.

Functional similarity matrix, which is a base for the calculation of similarity reflective sizes, consists of similarity percentages of the similar functional processes. The first horizontal and vertical cells of the matrix identify the functional processes and the numbers on the table are the functional similarity percentages among functional processes. An example of a functional similarity matrix can be seen on Table 7.

Other outputs of the tool, which are calculated based on the similarities among functional processes and the formulas, are "Discrete Similarity Reflective Functional Size" and "Continuous Similarity Reflective Functional Size". These adjusted sizes are presented within an Excel table, an example of which can be seen on Table 8.

Although, with the automation of the process, comparison time was decreased to seconds and a significant improvement was provided on the prevention of making mistakes during the comparison, it is not possible to make heuristic interpretations in the identification of functional similarities. Therefore, it is insufficient in the situations where the best judgment of the analyst is required.

3.1.3. Determination of the Similarity Reflective Functional Sizes

The third phase of the methodology is the identification of the similarity reflective functional sizes: Discrete Similarity Reflective Functional Size (DS) and Continuous Similarity Reflective Functional Size (CS). The calculation of these two sizes bases on the principles explained below. At the end of the calculation, the SR Tool represents two different similarity reflective sizes in addition to the functional similarity matrix.

Discrete Similarity Reflective Size (DS)

Discrete Similarity Reflective Functional Size (DS) is calculated by using constant functional similarity percentage values which change depending on five conditions. Constant similarity values are determined based on the functional similarities of the related functional process with remaining processes in the functional similarity matrix. The values within the rows on the functional similarity matrix show the similarity values of a functional process with other functional processes. The SR Tool determines the highest value from diagonal of the matrix to the left side of the row; checks one of the five conditions explained below and calculates the reflective functional size according to the formulas given below.

The constant functional similarity values, used within the DS formulas are derived from the software enhancement approach of NESMA (NESMA, 2001). In NESMA, sizes of data functions are multiplied by an impact factor, based on amount of the changes. The amount of change in NESMA is considered as the amount of functional similarity and the impact factors are taken as constants in our application. Discrete Functional Similarity Percentage Values can be seen on Table 3.

Table 3 Discrete Functional Similarity Percentage Value

Left Most Highest Similarity Percentage Value	**Functional Similarity Percentage Constants**
max<=34 %	0.25
0.34<max<=0.67	0.50
0.67<max<1.0	0.75
Max=1.0	0.1

Table 3 (Cont.)

Max=0	1.0

<u>*Continuous Similarity Reflective Size (CS)*</u>

Continuous Similarity Reflective Functional Size (CS) is calculated by using continuous functional similarity percentage values which are derived from the functional similarity matrix.

Continuous functional similarity percentage values, which are used within the formulas, are the highest values from diagonal of the matrix to the left side of the row in the Functional Similarity matrix. We assume that the highest similarity values are the closest candidates to be used in the reflective functional size calculations.

The other assumption is that the position of the functional process in the comparison data set depicts the functional process' production order. When functional process (FP) A is followed by functional process B; this means: FP A is coded before the FP B. Therefore the leftmost highest functional similarity percentage is chosen as the continuous similarity value.

To explain the structure of the method, let's consider six functional processes A, B, C, D, E and F. A<u>X,</u> B<u>X,</u> C<u>X</u>, D<u>X</u>, E<u>X</u> and F<u>X</u> shows the data movement and data group couples where X is the numbers.

<u>1st Condition</u>

A	: A1, A2, A3, A4, A5, A6
B	: B1, B2, B3

As can be seen above, A is 6 CFP while B is 3 CFP. Assuming that A1=B1, A2=B2 and A3=B3; we can say that B is % 50 similar to A and A is % 100 similar to B. The functional similarity matrix of this analysis is given on Table 4.

Table 4 Functional Similarity Matrix of Functional Process A and B

FP Name	**A**	**B**
A	1	0.5
B	1	1

When functional process A is coded before B and all of the data movements belonging to B occurs within the FP A; then to calculate the CSs; FP A's count is multiplied by 1, since it is the first functional process in the data set (1) and FP B's count is only multiplied by the reuse overhead since all its actions have been coded in A (2). To calculate the DS; FP A's count is multiplied by 1 (3), and FP B's count is only multiplied by the reuse overhead for the same reason above (4).

CSA = (count A) * 1 (1)

CSB = (count B) * reuse overhead (2)

DSA = (count A) * 1 (3)

DSB = (count B) * reuse overhead (4)

Since an investigation of the code is required for the previously produced DM-DG couples, the functional size of the process is multiplied by a reuse overhead. We have taken the reuse overhead as %10 for the all DS and CS formulas.

2nd Condition

C	: C1, C2, C3, C4, C5
D	: D1, D2, D3, D4

FP C's functional size is 5 CFP while D is 4 CFP. Assuming that C1=D1, C2=D2 and C3=D3; we can say that D is % 60 similar to C and C is % 75 similar to D. The functional similarity matrix of this analysis is given on Table 5.

Table 5 Functional Similarity Matrix of Functional Process C and D

FP Name	**C**	**D**
C	1	0.6
D	0.75	1

When functional process C is coded before D and some of the data movements belonging to C occurs within the FP D; then to calculate the CSs; FP C's count is multiplied by 1, since it is the first functional process in the data set (1) and FP D's size is calculated by the formula given in (5). To calculate the DSs; FP C's count is multiplied by 1 (3) because of the same reason above and FP D's size is calculated by the formula given in (6).

CS_C = (count C) * 1 (1)

CS_D = ((count D) * leftmost highest similarity value * reuse overhead) + ((count D) - (count D)* counter leftmost highest similarity value) (5)

CS_D = (4*0.75*0.1) + (4 - 4*0.75) = 1.3

DS_C = (count C) * 1 (3)

DS_D = (count D)* functional similarity percentage constant* reuse overhead) + ((count D) - (count D)* functional similarity percentage constant) (6)

DS_D = (4*0.75*0.1) + (4 - 4*0.75) = 1.3

3th Condition

E : E1, E2, E3, E4

F : F1, F2, F3, F4, F5

FP E's functional size is 4 CFP while FP D's is 5 CFP. Assuming that none of the data movement and data group couples is the same, we conclude that these two functional processes are not similar to each other, therefore each of them should be coded alone and their size shouldn't be modified.

CS_E = (count E) * 1

DS_F = (count F) * 1

4th Condition

If two functional processes are 100 % similar to each other, then the second functional process' size should multiplied by the reuse overhead value while the other's size remain the same as described in condition # 1.

3.1.4. Example for the application of the SiRFuS Method

In this subsection, we give an example to better explain how the method is applied. The example handled here is developed as a case product for the Software Management (SM) students who learn the measurement of the functional size with COSMIC. The measurement of the case was performed by the supervisor of the SM students' thesis. The requirements and the COSMIC measurement results of the example case can be found in Appendix A. Partial measurement data and the functional similarity matrix for this example are represented in Table 6 and Table 7 respectively.

The functional size of the example case is 68 CFP. It is constituted of 11 Functional Processes; 15 Entries, 22 Exits, 19 Reads and 12 Writes.

The functional similarity matrix of the case product which is calculated by SR Tool is given on Table 7. On this table, the first vertical and horizontal cells identify the functional processes, and the numbers identify the functional similarity percentages. The values in the diagonal of the matrix which are the same and 1 for all functional processes are colored with grey.

This similarity matrix was constituted by comparing the DM and DG tuples. The order of functional processes in the functional similarity matrix corresponds to the implementation order of the functions. For instance, we assumed that the first functional process will be developed before the second functional process. To identify the similarity of a functional process with other processes, the matrix should be read horizontally. For example 4th functional process (FP) is 100% similar to 1st FP; 5th FP is 33.3% similar to 1st FP whereas 1st FP is 7.6% similar to 5th FP (they are colored as blue in Table 7) and finally there is no similarity between 6th FP and 1st

FP. If we analyze three examples given above, we are able to explain all the conditions that can be seen during the similarity evaluation.

The similarity between FP 5 and FP 1 refers to condition A given in Section 3.1.2. The only similarity between these two functional processes is the “error conformation messages” which are colored with grey in Table 6. Therefore the FP 5 is 33% (1/3) similar to FP 1 which means FP 5 can makes use of the 33% of the tuples of FP1. Besides FP 1 is 7.6% (1/13) similar to FP 5 which means FP 1 can makes use of the 7.6% of the tuples of FP5.

On the other hand, the similarity between FP 4 and FP 1 refers to condition B given in Section 3.1.2. Since the all (DM, DG) tuples within these functional processes are the same; they are 100% similar to each other. However, if one of the functional processes were the subset of the other functional process, there wouldn’t be two-way 100% similarity.

The similarity between FP 6 and FP 1 refers to condition C given in Section 3.1.2. Since any of the DM-DG tuples of these processes are the same; there is no similarity between them.

Table 6 An Example set from the Measurement Data of Movie Manager

FP ID	FP Name	No	DM	DG
1	AddPerson	1	Entry	Personinfo
		2	Write	Personinfo
		3	Exit	Error/Confirmation
2	ListPersons	4	Entry	Listpersonsrequest
		5	Read	Personinfo
		6	Exit	Personinfo
3	RetrievePerson	7	Entry	Retrivepersondetailsrequest
		8	Read	Persondetailsinfo
		9	Exit	Persondetailsinfo
4	UpdatePerson	10	Entry	Personinfo
		11	Write	Personinfo
		12	Exit	Error/Confirmation
5	AddMovie	13	Entry	MovieInfo
		14	Read	Personinfo
		15	Exit	Personinfo
		16	Entry	Writerinfo

Table 6 (Cont.)

		17	Entry	Producerinfo
		18	Entry	Castinfo
		19	Entry	Directorinfo
		20	Write	MovieInfo
		21	Write	Writerinfo
		22	Write	Producerinfo
		23	Write	Castinfo
		24	Write	Directorinfo
		25	Exit	Error/Confirmation
6	QueryMovie	26	Entry	QueryParameters1
		27	Read	MovieInfo
		28	Exit	MovieInfotitleyear

Table 7 Functional Similarity Matrix of Movie Manager

FP	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	1	0.333	0	0	0	0	0	0
2	0	1	0	0	0.666	0	0	0.666	0.666	0	0
3	0	0	1	0	0	0	0	0	0	0	0
4	1	0	0	1	0.333	0	0	0	0	0	0
5	0.076	0.153	0	0.076	1	0	0	0.153	0.153	0	0.384
6	0	0	0	0	0	1	0.333	0	0.333	0.333	0.333
7	0	0	0	0	0	0.076	1	0	0.769	0.153	0.230
8	0	0.666	0	0	0.666	0	0	1	0.666	0	0
9	0	0.153	0	0	0.153	0.076	0.769	0.153	1	0.153	0.153
10	0	0	0	0	0	0.333	0.666	0	0.666	1	0.666
11	0	0	0	0	0.625	0.125	0.375	0	0.25	0.25	1

The Plain (PS), Discrete (DS) and Continuous Functional Sizes (CS) of case product Movie Manager is given on Table 8. The second row in the table gives the plain size of each functional process separately. The values in the third and the fourth rows are calculated by applying the formulas of DS and CS given in section 3.1.3.

Table 8 Plain, Discrete and Continuous Functional Sizes of Movie Manager

Size/FP	1	2	3	4	5	6	7	8	9	10	11	TOTAL
PS	3	3	3	3	13	3	13	3	13	3	8	**68**
DS	3	3	3	0.3	10.07	3	10.07	1.65	4.2	1.65	4.4	**44.37**
CS	3	3	3	0.3	11.2	3	12.1	1.2	4	1.2	3.5	**45.5**

As can be seen from the table above, the functional size of the Movie Manager was decreased from 68 to 44.375 and 45.5 when the functional similarities were considered for reuse.

The first functional process' size is preserved as is; since there is no functional process to be candidate for reuse produced before it. The 3rd functional process' DS and CS sizes are determined by multiplying its plain size (PS) by one; since FP3 is similar to neither FP1 nor FP2. The 4th functional process' PS size is multiplied only by the reuse overhead factor to calculate DS and CSs, since the FP4 is 100% similar to FP1. The 5th functional process' situation is a good example for the 2nd condition given in Section 3.1.3. The DS and CS sizes are calculated by the formulas given below.

DS_5 = (count FP5)* functional similarity percentage constant* reuse overhead) + ((count FP5) - (count FP5)* functional similarity percentage constant)

$DS_5 = (13*0.25*0.1) + (13 - 13*0.25) = 10.075$

CS_5 = ((count FP5) * leftmost highest similarity value * reuse overhead) + ((count FP5) - (count FP5)* counter leftmost highest similarity value)
$CS_5 = (13*0.153*0.1) + (13 - 13*0.153) = 11.21$

CHAPTER 4

CASE STUDIES TO DETERMINE FUNCTIONAL SIZE CONSIDERING FUNCTIONAL SIMILARITIES

4.1 Overview

We have conducted a single case study to analyze the reasons of the low correlation between functional size and development effort and to identify the improvement opportunities for this problem. After we had observed the reason of problem was the similarity of the functions, we applied the functional similarity identification method of Santillo and Abran. After identifying the reasons of the correlation problem between the functional size and development effort and the deficiencies of the method of Santillo and Abran, we developed a method to solve the problem. In addition to the single case study, we conducted a multiple case study involving eight cases in order to better evaluate the applicability of the functional similarity determination methods and to evaluate the impact of SiRFuS on the correlation of functional size and effort. Main goals were, to observe if the functional similarities would improve the relation between functional size and total effort and to total effort and to determine the best functional size (plain or adjusted) for the highest correlation.

For the single case study, KN (Karagöz, 2008) is chosen to evaluate the reasons of the correlation failures and improvement opportunities. KN included so many similar functional processes, data entities and attributes that it can be described as a challenging application from the planning perspective. The boundaries of KN's processes and of data entities can be changed based on the measurer's estimations and assumptions (Turetken, et al., 2008). This case is later included in the scope of the multiple case studies because of the reasons explained below.

For the multiple case study, we aimed to evaluate a number of products in different domains of application, to be able to generalize the applicability and the accuracy of the method we suggested. We have started case study research with 17 cases. We have selected only the 8 case products among these 17 cases. This is because we were not able to get the Software Requirements Specification (SRS) Documents or the measurement results for the six of the case products and the three case products were not proper for the functional similarity analysis, since their functional size measurement data were not collected and written in a systematic order.

As a result we have selected the case products; CN, SN, TN, AN, BN, DN, MN and KN since they had well documented SRS and functional size measurement results. Another reason for the selection of these cases was the consistency and the accuracy of the collected effort values for the cases. However, we have identified that there were major inconsistencies for the effort values of DN and MN. In DN we obtained only the programming effort value which was only 6 man-days. When compared to the other case products' efforts, it does not seem so accurate. In MN, the total effort for the development of the project was 280 man-hours which is not possible for a project whose size is 208 CFP; because of the reasons above, we used the Lines of Codes values for the observation of the effect of functional similarities for these two cases.

All of the selected case products are Information Systems (IS) Projects, except for the AN, which is a Complex Data Driven Control System Project with respect to the CHAR Method defined in (ISO/IEC, 2004). As almost all of them were IS projects, we grouped the case products based on their organizations. The productivity ratios of the teams could diverge for different organizations for different teams, and for different applications whereas they are expected to be similar for the teams that work in the projects for the same application domain in the same organization (Jones, 1998).

The case studies involved the measurement of the case products, (some of which were measured previously as a part of another study as explained in Section4.2), by using the COSMIC Method; and identification of the functional similarities and the equivalent functional sizes, by using Similarity Reflective Functional Size Method (SiRFuS). The aim of SiRFus is to determine an equivalent size which provides a relation with the total effort utilized to build a software product. Therefore, the equivalent sizes were determined using the method that gave the best analysis results among the four candidates. The first approach applied is called as “Plain Functional Size (PS)”, in which the functional sizes of the products are measured according to the rules given in COSMIC Guideline (ISO/IEC, 2003b), and functional similarities are not taken into account. The second one is called as “Average Similarity Reflective Functional Size (ASR)”, which is determined by applying the average functional similarity percentage values to the PSs. The third one “Discrete Functional Size (DS)” and the fourth one “Continuous Functional Size (CS)” are determined according to the SiRFuS method explained in Chapter 3. Best approaches, which provide a better relation between functional size and total effort, are determined by analyzing the case study results.

This chapter presents the details of the case studies.

4.2 Conduct of Case Studies

The size of a software product is the main input for most estimation models to determine the effort and the cost of software projects. Functional size is one of the size measures that can be calculated at the early phases of the software life cycle. Although the functional size measurement methods are improving and the functional size results reflect the real situation better, the mapping of the functional size with the total effort can not be achieved precisely by using the conventional approaches. However, one of the indicators of the size and effort relation, Productivity, should be similar for the teams in the same organization and for the projects that belong to the same application domain. Therefore, we hypothesized that the quantitative relation of the functional size and the total effort can be improved by determining the functional similarities in a software product and reflecting them to the functional size measurement results.

Considering the problem and the possible solution above; we determined the following research questions, and conducted a multiple case study to evaluate the hypothesis above.

- Are estimated efforts using COCOMO II for the case products consistent with actual software development efforts?
- What is the use of the identification of the functional similarities?
- How do the current COSMIC based functional similarity quantification methods including SiRFuS improve the relation between functional size and the total effort?
- What are the problems and the difficulties of the current COSMIC based functional similarity quantification methods and the improvement opportunities?
- Do the COSMIC based functional similarity identification methods efficient or is the effort required to evaluate the functional similarities acceptable?

The case study process progressed as follows:

Firstly, all of the case products were measured by using COSMIC v3.0 by the four measurers, one of whom is the author of this thesis, and the others were the former MSc students of the Software Management (SM) Program in Informatics Institute at METU. SM Program's students measured the case products in the scope of their term project studies based on the software requirement specification documents of the products'. The studies were coordinated by Dr. Onur Demirörs and Dr. Oktay Türetken.

In the scope of the term projects studies, each measurer first read and worked on the measurement manual of COSMIC and then measured a fictitious project (explained in Chapter 3) to be applicable for the size measurement and to provide the accuracy of the measurement results. The results of the fictitious project were verified by the supervisors of the term projects.

To provide the integrity among the size measurement results, students measured the functional sizes of the case products' by filling a preformatted Excel table which consisted of "functional process name", "data movement type", and "data group description". In addition to this, students were asked to fill the data collection questionnaire of the International Software Benchmarking Standards Group (ISBSG, 2007) that includes Project Progress, Technology, People and Work Effort information.

After all of the case products had been measured, they were controlled and verified by the supervisors of the term projects and the author of this thesis.

The second phase of the research consisted of the identification of the functional similarities within the software products. Therefore, all of the measurement data sets were arranged as to be inputs to the SR Tool. Measurement results, which were kept in Excel tables, were rearranged so as to include the "functional process name", "data movement number", "data group description", and "data movement type" information, and file formats were changed from "xls" to "txt".

The functional similarities were identified by comparing the functional processes based on the data movement - data group couples with the help of SR Tool (see section 3.2.2). The SR Tool takes the txt files as input and generates a functional similarity matrix table, a Discrete Functional Size (DS) result table and a Continuous Functional Size (CS) result table for each case product as output. The functional similarities, DSs and CSs are calculated according to the rules given in Chapter 3. Based on the functional similarity matrices, the average functional similarity of each product and the Average Similarity Reflective Functional Size (ASR) were calculated based on the formulas given in equations (9) and (10). Santillo and Abran had suggested the calculation of average similarities for the identification of the reuse potentials of the products. However, a formula is not presented by them in their work; we used the formula given in (9). We took the average similarity calculation a step further and developed the formula given in (10) and used average similarities of the products to calculate the ASRs.

$$\text{Average Similarity} = \frac{\text{Sum of the whole matrix} - \text{\# of Functional Processes}}{\text{\# of Functional Processes}^2 - \text{\# of Functional Processes}} . 100 \qquad (9)$$

$$ASR = PS. \frac{(100 - Average\ Similarity)}{100} \qquad (10)$$

After this step, effort values to develop, manage and maintain the case products were gathered from the ISBSG questionnaires in which the students filled out. The effort details of the case products can be found in the section 4.2.1 where the cases are described.

The case products' effort values except for KN were in terms of man–hours. Therefore, KN's effort values were converted from man-days to man-hours, by multiplying the efforts with the utilization factor which is assumed as "5" as can be seen on Table 13.

Since the total effort of each case product does not include the supporting processes, the effort values of the supporting processes were removed in the case products CN, SN, TN and KN, to make the comparisons among the case products to be consistent as can be seen on the 4th column of Table 18. For the case products DN and MN, the Lines of Codes values were used for the comparison instead of the total effort.

The last stage of the case study research is the calculation of the productivity ratios, since the productivity is a measure, from which the relation between the size and effort can be observed. The details of products subject to the case studies can be found in section 4.2.1.

4.2.1 Description of the Case Products

Eight case products were evaluated in the context of the multiple case studies. The details and characteristic of the cases are explained in this section.

Organization#1

Three case products in this section were developed within the same organization by the same team. All of the products had web-based graphical user interfaces and are developed using the waterfall life-cycle model. Software Requirements Specification (SRS) documents of the products were conformant with the IEEE Standard 830-1998 (IEEE, 1998).

The software tools and programming languages used throughout the software life cycle were as follows (Urgun, 2008): JAVA as the programming language, IBM WebSphere Application Developer as the development environment, Borland Together Architect as the analysis and design tool, CA All Fusion Harvest as the change management and version controlling tool and Telelogic DOORS as the traceability tool. Database Management Systems were DB2 in the products.

Case Product-1: CN

CN is a support tool that provides a paperless flow of information for change management activities in design processes. It is possible to initiate, review and approve change requests; organize configuration control board meetings and analyze change effects by using the tool.

The project was initiated in April 2007 and completed in June 2007. The project staff consisted of 8 people; 1 Project Leader, 1 Software Quality Assurance Representative, 1 Configuration Manager, 1 System Analyst, 1 System Designer, 1 System Developer, 1 Tester, 1 Database Administrator.

Total Effort required to develop this project is 1200.42 person-hours. Details of the effort utilized are given on Table 9.

Table 9 Development Effort of the CN

Software Development Life Cycle Phase	**Effort (person-hours)**
Development Processes	502.74
Software Requirements Analysis	102.6
Software Design	71.82
Software Coding & Unit Testing	184.68
Testing	143.64
Management	410.4
Supporting Processes	287.28
Total	**1200.42**

Case Product-2: SN

SN, a Stationery Requisition System project, was developed for the purpose of managing the requests of stationary material purchase of departments throughout an approval workflow, on an electronic, paperless environment (Urgun, 2008). With the help of the tool, human effort on the purchase process is minimized, possible errors are handled and request approval mechanism is automated.

The project started in May 2007 and was completed in December 2007. The project staff consisted of 9 people; 1 Project Leader, 2 Software Quality Assurance Representative, 1 Configuration Manager, 1 System Analyst, 1 System Designer, 1 System Developer, 1 Tester, 1 Database Administrator.

Total Effort required to develop this project is 1256.36 person-hours. Details of the effort utilized are given on Table 10.

Table 10 Development Efforts of the SN

Software Development Life Cycle Phase	**Effort (person-hours)**
Development Processes	532.41
Software Requirements Analysis	129.72
Software Design	70.93
Software Coding & Unit Testing	161.74
Testing	170.02
Management	207
Supporting Processes	516.95
Total	**1256.36**

Case Product-3: TN

TN is a system that is used to follow up the Letters of Credit, received or sent by the Finance and Accounting Management, Material Planning and Procurement Management and Facilities Management Departments of the organization (Urgun, 2008). TN provides the interaction among General Accounting, Purchase Management, Authorization, and Human Resources Systems of the organization. Received Letters of Credits can be registered to the system; registered or sent letters of credits can be followed up; reports are provided to the related users throughout the system.

The project started in January 2007 and completed in January 2007. The project staff consisted of 8 people; 1 Project Leader, 1 Software Quality Assurance Representative, 1 Configuration Manager, 1 System Analyst, 1 System Designer, 1 System Developer, 1 Tester, and 1 Database Administrator.

Total Effort required to develop this project is 1400.08 person-hours. Details of the effort utilized are given on Table 11.

Table 11 Development Efforts of the TN

Software Development Life Cycle Phase	**Effort (person-hours)**
Development Processes	774.15
Software Requirements Analysis	171.09
Software Design	120.27
Software Coding & Unit Testing	118.58
Testing	364.21
Management	354.89
Supporting Processes	271.04
Total	**1400.08**

Organization # 2

AN and BN were developed within the same organization. These two case products were web based projects developed with JAVA and generated with AJAX and Struts. Database Management Systems were Oracle 9i and 10i used in the products.

Case Product # 4:AN

AN was developed to provide management interfaces for water subscribers. System has interfaces with outer system components like hand terminals for reading water meter, and banks for payment information (Ergüden, 2008).

The project started in March 2006 and completed in March 2007. The project staff consisted of 3 people; 1 Project Manager who is also Technological Leader, 1 Senior Software Engineer, and 1 Junior Software Engineer. Total Effort required to develop this project is 5950 person-hours. However, only the 32% of the whole project was subject to the measurement. Therefore the effort required to develop the measured module is 3594 person-hours. Details of the effort utilized are given on Table 12.

Table 12 Development Efforts of the AN

Software Development Life Cycle Phase	**Effort (person-hours)**
Development Processes	3038
Software Requirements Analysis	452
Software Design	2287
Software Coding & Unit Testing	64
Testing	235
Management	556
Supporting Processes	---
Total	**3594**

Case Product # 5: BN

BN was developed to manage the budgeting process automatically. BN gathers budget's expenditure information from "accounting module" through an interface and then consolidates this information with budget items (Ergüden, 2008).

The project started in January 2008 and completed in August 2008. The project staff consisted of 2 people; 1 Senior Software Engineer, and 1 Junior Software Engineer.

Total Effort required to develop this project is 1584 person-hours. However, we do not have the details of this actual effort values. The only information we have is that the total effort value does not include the effort required for supporting processes.

Organization # 3

Following two web based case products were developed within the same organization by using JAVA (J2ee) and generated with Hibernate and Struts. The applications were run on Oracle IAS 10 Application Server and Oracle 10g Database Server (Şentürk, 2008).

We could not reach the reliable effort values of the projects; therefore these two projects will be evaluated from the Lines of Code (LOC) and functional size relation perspective.

Case Product # 6: MN

MN was developed to manage the finance applications of the organization (Şentürk, 2008).

The project staff consisted of 2 people; 1 Senior Software Engineer, and 1 Junior Software Engineer.

The size of the product in LOC is determined as 1950 by measurements with the "Practiline Source Code Line Counter v1.1"(Software, 2008).

Case Product # 7: DN

DN was developed to be used for management of vehicle activities in the "General Directorate of Highways" and "Radio and Television Supreme Council" as a sub module of Human Resource Management System. The project staff consisted of 5 people; 1 Project Manager, 1 Team Leader, 1 Senior Software Engineer, and 2 Junior Software Engineer.

The total size of the product in LOC is determined as 46270 by measuring with the "Practiline Source Code Line Counter v1.1". However, the size of the part that subject to the measurement is calculated as 12087.

Organization # 4

The last case product, KN, was developed as a conceptual modeling tool with the consortium of two organizations. For the software analysis and design, Rational Software Architect tool; for the requirements management, Requisite Pro tool; and C# as the programming language have been utilized in KN. Unified Modeling Language (UML) (Group, 2005) was used for representing analysis and design, and Subversion Tool was used for configuration control (Karagöz, 2008).

Case Product # 8: KN

KN is a conceptual modeling tool development project. The tool provides a common notation and a method for the conceptual model developers in different modeling and simulation development projects, particularly in the military domain.

The project staff utilized in the projects consisted of 21 people; 1 Project Manager, 1 Assistant Project Manager, 2 Steering Committee Members, 1 Project Coordinator, 8 Researchers, 1 Software Development Team Leader, 1 Quality Assurance Team Leader, 4 Software Engineers (1 part-time), 1 Part-time Test Engineer and 2 Quality Engineers (1 part-time).

It was assumed that the project staff could work 5 hours a day by considering the work capacity. The efforts utilized for the project totaled up to 1,832 person-days which equals to 9160 person-hours. Details of the effort utilized are given on Table 13.

Table 13 Development Efforts of the KN

Software Development Life Cycle Phase	**Effort (man-day)**	**Effort (man-hour)**
Development Processes	1287	6435
Software Requirements Analysis	227	1135
Software Design	185	925
Software Coding & Unit Testing	670	3350
Testing	205	1025
Management	135	675
Supporting Processes	410	2050
Total	**1832**	**9160**

4.2.2 General Discussions on the Case Studies and the Results

The case study results and our inferences based on these results are explained in the following paragraphs.

The functional sizes of the case products were measured with COSMICv3.0. The summary of the measurement results are given on Table 14.

Table 14 Measurement Results of the Case Products based on COSMIC v3.0

Org. No	Case Product	No. of Functional Processes	No. of Entries	No. of Reads	No. of Writes	No. of Exits	Total Functional Size (CFP)
Org. # 1	CN	16	32	22	25	29	108
	SN	10	24	24	10	18	76
	TN	27	49	39	16	52	156
Org. # 2	AN	36	49	98	17	117	281
	BN	34	37	51	20	70	178
Org. # 3	DN	45	45	68	18	86	217
	MN	44	44	55	31	78	208
Org. #	KN	136	324	419	657	596	1996

To be able to compare the estimation accuracy of widely used methods, we estimated the efforts with COCOMO II. In COCOMO II, there are six scales for the cost drivers which change from very low to extremely high (see section 3.2.2). As we did not have detailed information to identify the scale factors and the effort multipliers we assumed that “normal level” is acceptable for scale factors and effort multipliers for all the case products. COCOMO II uses SLOC values to estimate effort. Since, we don’t have the SLOC values for case products CN, SN and TN; we converted their functional size results to SLOCs based on the conversion equation given in (http://www.qsm.com/FPGearing.html, April 2005). The results are depicted in Table 15. In the second column of the table, functional sizes of the case products; in the third column, LOCs of the case products; in the fourth column, actual work effort and in the last column estimated effort values using COCOMO II are listed.

Table 15 Actual and COCOMO II Based Estimated Work Efforts

Case Product	Functional Size (CFP)	LOC	Actual Work Effort (MM)	Estimated Work Effort with COCOMO II
CN	108	6372	7.5	21.2
SN	76	4484	7.8	13.5
TN	156	9499	8.7	32.9
AN	281	36154	22.4	151.3
BN	178	18269	9.9	70.6
KN	1996	91609	57.25	419.5

As can be seen from the table, there are significant deviations between the estimated and the actual work efforts. Deviations change between 5.6 man-months to 362.3 man-months. We can observe that COCOMO II overestimated the required effort. Actually, there are various factors that can have impact on the estimation of efforts of the cases. One of them is the estimation of LOC values for the first three cases. The estimated LOCs may be different from the actual ones.

COCOMO II takes into account the external reused code. None of the case products used similar code from previous projects; however, they have considerable potential for the internal reuse. We think that another reason of this failure is, not evaluating the impact of functional similarities and the reuse potential during the effort estimation.

The other objective of this study is the evaluation of the current COSMIC based functional similarity identification methods. After the functional similarity matrices had been constituted by comparing each data movement and data group tuples within the functional processes, we calculated the average functional similarity values for each case product by applying the formula given in (9) in section 4.2, which can be found in Table 16. The average functional similarity values given on the 3rd column of the Table 16 were calculated to be used in the calculation of the Average Similarity Reflective Functional Size (ASR) values to evaluate the impact of average functional similarity of the whole product in FS and effort correlation. After we had calculated the average similarities given on Table 16, we observed a problem of the approach of Santillo and Abran. In their research, Santillo and Abran, calculates the average similarities and decides to apply reuse internally based on the attained Average Similarities. In fact, we observed that the average functional similarities of the whole product do not indicate the real reuse capacity within the products, especially when the number of functional processes increases. For instance, although we easily observed the similarities among functional processes while reading the Functional User Requirements of one of the case products, KN; we only had a 12.6 % similarity for the whole product. On the other hand, if KN did not have the potential for reuse with its high similarities; it wouldn't be possible to approximate the productivity of KN to other case projects as can be seen on Figure 3 and Figure 4. The reason of the difference between the average similarity and the potential similarity of the product may be caused by the fact that when we calculate the average similarity, we take the non similar functions in to consideration. When the number of non similar functions increased which means 0% percentages, the average similarity decreased unrealistically.

Table 16 Average Functional Similarity of Each Case Product

Organization	Case Product	Average Similarities (%)
Org. # 1	CN	18.01
	SN	25.25
	TN	31.70
Org. # 2	AN	12.77
	BN	13.45
Org. # 3	DN	11.32
	MN	12.69
Org. # 4	KN	12.70

After the functional similarity matrices were constituted, we calculated three adjusted functional size values (ASR, DS and CS) in addition to Plain Functional Size (PS), which are given on Table 17. Plain Functional Size (PS) is just the COSMIC v3.0 functional size measurement results of the case products. The PS values of the case products can be seen on the 3rd column of Table 17. Average Similarity Reflective (ASR) Functional Sizes were calculated by applying the average similarity values given on the Table 16 to PS values, based on the formula given in (9). Discrete (DS) and Continuous (CS) Similarity Reflective Functional Sizes were calculated based on the highest functional similarity values of the functional processes derived from the functional similarity matrix. The only difference between DS and CS is that; the DSs were calculated considering a constant similarity interval based on the highest similarity values; on the other hand CS were calculated based on the value of highest functional similarity itself. The formulas for calculating DS and CS can be found in Section 3.2.3.

Table 17 Plain and Similarity Reflective Sizes of the Case Products

Org	Case Product	PS	ASR	DS	CS
Org. # 1	CN	108	88.55	61.43	63.0
	SN	76	56.81	40.45	42.7
	TN	156	109.97	61.10	58.4
Org. # 2	AN	281	246.86	167.80	161.5
	BN	178	154.05	103.75	101.5
Org. # 3	DN	217	192.44	103.15	99.1
	MN	208	181.60	106.98	100.9
Org. # 4	KN	1996	1742.57	682.22	601.9

One of the objectives of this study was to investigate the relation between functional size and total effort. Therefore, after we had attained the functional size measurement results; we obtained the total effort values of the case products given on Table 18. The values on the 2nd column of the table are for the total efforts in which the supporting processes are included; whereas the values on the fourth column are the total efforts in which the supporting processes are excluded. Both the values on the 2nd column and 4th column are used for the observation of the differences between these two conditions in the productivity analysis.

Table 18 Total Effort and LOC Values of the Case Products

Org.	Case Product	Effort (1) (man –hour)	Supporting Process Efforts	Effort (2) (man –hour)	LOC
Org. # 1	CN	1200.42	287.28	913.14	-
	SN	1256.36	516.95	739.41	-
	TN	1400.08	271.04	1129.04	-
Org. # 2	AN	3594	-	3594	36154
	BN	1584	-	1584	18269
Org. # 3	DN	-	-	-	12087
Org. # 3	MN	-	-	-	19690
Org. # 4	KN	9160	2050	7110	91609

We calculated the functional size per hour and the lines of code per function point values for each type of functional size of each case product, in order to observe the correlation between functional size and total effort, and the correlation between functional size and lines of code, respectively. The productivity values of the case products which are indicators of the correlation between functional size and total effort are given on Table 19 and Table 20, whereas the LOC per CFPs are given on Table 24.

Based on the effort values in 3rd column of Table 18, in which the supporting processes are included, and the functional sizes on Table 17, the productivity ratios were calculated as seen on Table 19, for every size value of each case product. The Productivity#1 indicate the plain functional size per hour; the Productivity#2 indicate the average similarity functional size per hour; the Productivity#3 indicate the discrete similarity reflective functional size per hour and finally the Productivity#4 indicate the continuous similarity reflective functional size per hour. In Figure 3, the productivity ratios for case products on Table 19 are plotted. The first three and the following two case products were developed by different organizations. It can be deduced from the figure that the variances for Productivity#1 and 2 are higher than Productivity#3 and 4. The standard deviation of Productivity#1 and 2 is about 0.0556 and 0.0508 whereas the standard deviation of Productivity#3 and 4 is about 0.0154 and 0.0127 respectively. These significant decreases on the deviations reveal that the adjusted functional sizes calculated based on the rules of SiRFuS method, provide better correlation with total effort. One of the objectives of this study was to improve the relation between functional size and development effort which is seen to be accomplished observing these facts. Since we showed up the correlation of functional size and effort with the productivity values in which the DS or CS sizes were used, the effort and the cost to develop software projects can be identified reliably using effort and cost estimation models.

Table 19 Productivity Ratios of the Cases in which the Supporting Processes Included to Total Effort (man-hour)

Org.	Case Product	PS/Effort	ASR/Effort	DS/Effort	CS/Effort
		P#1	P#2	P#3	P#4
Org. # 1	CN	0.090	0.074	0.051	0.052
	SN	0.060	0.045	0.032	0.034
	TN	0.115	0.079	0.044	0.042
Org. # 2	AN	0.079	0.069	0.047	0.045
	BN	0.112	0.097	0.065	0.064
Org. # 3	DN	-	-	-	-
	MN	-	-	-	-
Org. # 4	KN	0.218	0.190	0.074	0.066
Std. Deviation		0.0556	0.056	0.051	0.015

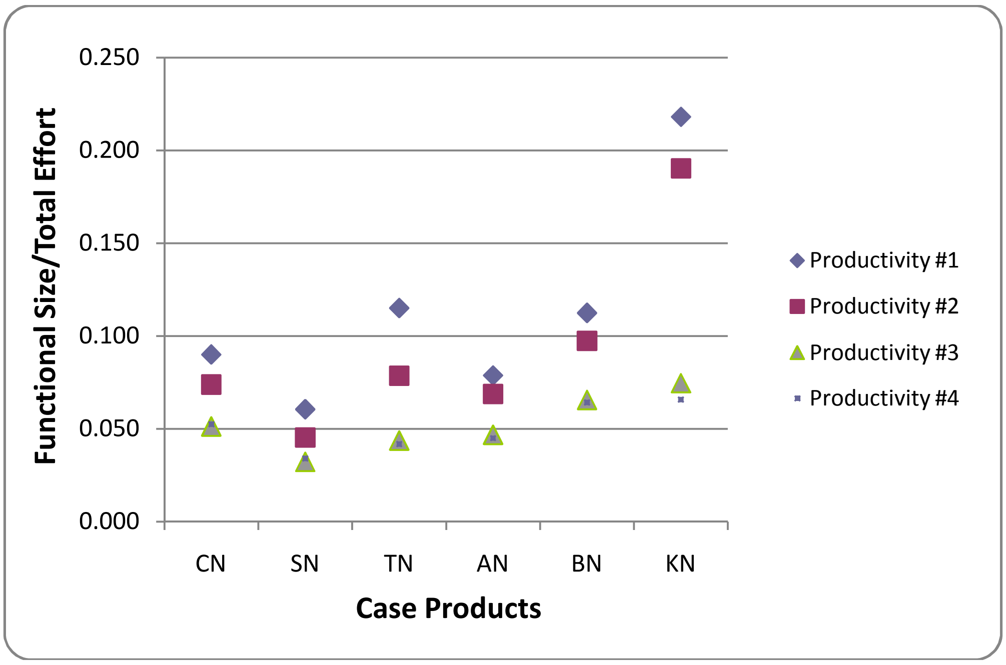

Figure 3 Distribution of the Productivity Ratios Based on the values in Table 19

For the purpose of identifying the relationship between functional size and the total effort in which the supporting processes are excluded, we calculated the productivity measures for the same case products as above, and for each type of functional size.

Based on the effort values in 5th column of Table 18, in which the supporting processes are excluded; and the Plain and Similarity Reflective functional sizes on

Table 17, the productivity ratios were calculated as seen on Table 20, for every size value of each case product. The Productivity#1 indicate the plain functional size per hour; the Productivity#2 indicate the average similarity functional size per hour; the Productivity#3 indicate the discrete similarity reflective functional size per hour and finally the Productivity#4 indicate the continuous similarity reflective functional size per hour. In Figure 4, the productivity ratios for case products on Table 20 are plotted. As emphasized previously, the first three and the following two case products were developed by different organizations. It can be deduced from the figure that, as they are in Figure 3, the variances for Productivity#1 and 2 are higher than Productivity#3 and 4. The standard deviation of Productivity#1 and 2 is about 0.0724 and 0.0655 whereas the standard deviation of Productivity#3 and 4 is about 0.0174 and 0.014 respectively. These significant decreases on the deviations reveal that the adjusted functional sizes calculated based on the rules of SiRFuS method and the efforts in which the supporting processes excluded, provide better correlation with total effort with compared to the Plain Size and Average Similarity Reflective Size. This high correlation with the correlation given on Figure 3; reveal that significant improvement was provided by identifying the functional similarities and considering the founded similarities on functional size calculation. To emphasize the improvement; the functional size values attained by using SiRFuS method, will lead to analyst better plan, monitor and control software projects as reliable inputs.

Table 20 Productivity Ratios of the Cases in which the Supporting Processes Excluded from Total Effort (man-hour)

Org.	**Case Product**	**PS/Effort**	**ASR/Effort**	**DS/Effort**	**CS/Effort**
		P#1	**P#2**	**P#3**	**P#4**
Org. # 1	CN	0.118	0.097	0.067	0.069
	SN	0.103	0.077	0.055	0.058
	TN	0.143	0.097	0.054	0.052
Org. # 2	AN	0.079	0.069	0.047	0.045
	BN	0.112	0.097	0.065	0.064
Org. # 3	DN	-	-	-	-
	MN	-	-	-	-
Org. # 4	KN	0.281	0.245	0.096	0.085
Std. Deviation		0.072	0.072	0.066	0.017

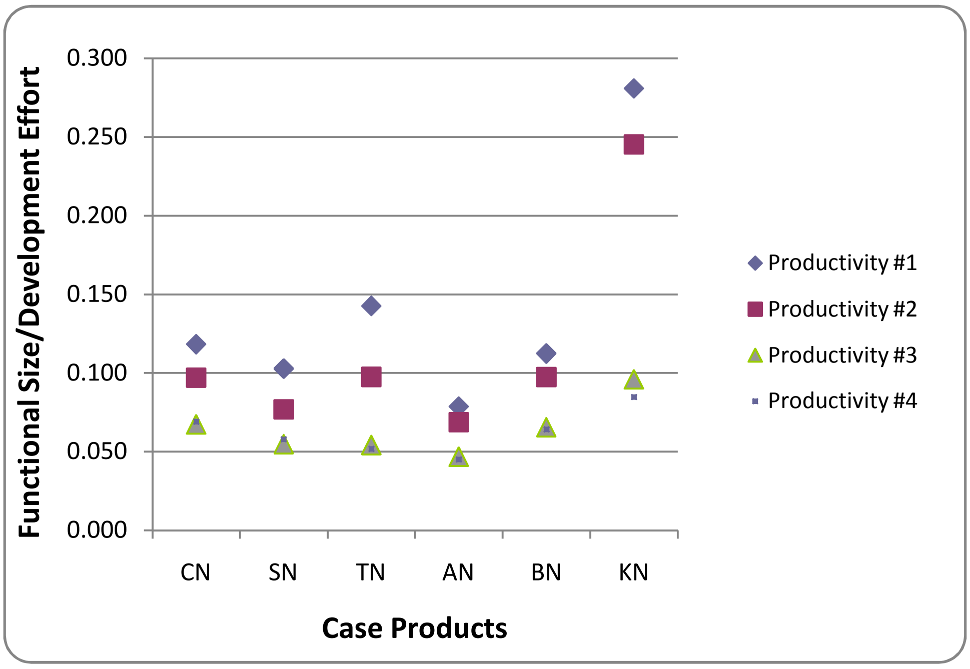

Figure 4 Distribution of the Productivity Ratios Based on the values in Table 20

To verify the accuracy of the productivity ratios given on Table 19, they were compared with the values published in (Jones, 1998). Jones mentioned that the productivity of cumulative software development activities range from 1.9 to 13.88 function points per month. The software development process he analyzed for the calculation of the productivities comprised of 25 activities such as requirements, prototyping, coding, configuration management, documentation, quality assurance, project management etc. Therefore, the values on Table 19 were preferred to be used in the comparison, since they include the efforts of the supporting process activities and there are no such considerable differences between the productivities in Table 19 and Table 20. Since the productivity values in Table 19 were calculated based on the effort values in man-hour scale; they were converted to man-month scale by assuming that total work hour per month is 160. The productivity ratios of the case products in man-month scale can be seen on Table 21. Although the measurements of Jones are performed by IFPUG and can not be directly compared to COSMIC, as their measurement scales are different, it is known that the conversion between IFPUG and COSMIC are almost one to one (Desharnais, Abran, & Cuadrado-

Gallego, 2006), (Urgun, 2008). Therefore there is no obligation to use the interval of Jones as is for comparison.

Table 21 Productivity Ratios of the Cases in which the Supporting Processes Included to Total Effort (man-month)

Case Product	PS	ASR	DS	CS
	P#1	**P#2**	**P#3**	**P#4**
CN	14.4	11.8	8.2	8.4
SN	9.7	7.2	5.2	5.4
TN	18.4	12.6	7.0	6.7
AN	12.6	11.0	7.5	7.2
BN	18.0	15.6	10.5	10.3
KN	34.9	30.4	11.9	10.5

Analyzing the values in Table 21, it was observed that the interval of Productivity#3 which was calculated based on the Discrete Similarity Reflective Functional Size values (DS) on Table 19, changes between 5.15 and 11.9 whereas the interval of Productivity#4 which was calculated based on the Continuous Similarity Reflective Functional Size values (CS) on Table 19, changes between 5.43 and 10.5. This means that both the Productivity# 3 and Productivity# 4 are in accordance with the productivity values provided by Jones (Jones, 1998). Since the interval of the productivity values provided by Jones is 1.9 to 13.88; the Productivity#1 and Productivity#2 fails for this comparison by exceeding the bounds referenced. In other words, if an organization use plain functional sizes (PS) for effort or cost estimation, it will probably overestimate the effort or the cost required to complete a project which leads to commitment of too many resources to the project.

When we analyze the Productivity#2 data which was calculated based on the Average Similarity Reflective Functional Size (ASR) values from the same perspective above; we observe that the ASR values also fail in the verification test. We detected that; consideration of the average similarity percentage values of the products is not an appropriate approach to attain an adjusted functional size, ASR. Even when we take the approach of Santillo and Abran a step further, we can not obtain satisfying results. We think the reason of this failure is that the Average

Similarities does not reflect the real similarity values of the products, since during the average similarity calculation, the value of the non similar functions decrease the similarity value as explained in the beginning of this section.

For the calculation of adjusted functional sizes, we had identified the reuse overhead as the 10% of the functional size in the beginning of this study. Later, we investigated the most suitable reuse overhead value which provides the best linear productivity line and tested whether the current reuse overhead assumption was precise or not. By changing the reuse overhead value from 0 to 0.3 with a 0.01 interval, we calculated the DS and CS sizes and the productivity#3 and productivity#4 values. At the end of this analysis we observed that we can not attain a more reliable productivity value by changing the reuse overhead. However, when the reuse overhead decreases, the deviation between the highest point and the lowest point of a productivity line decreases as well. For instance, at 0.01 point, the difference between the highest point and the lowest point of the productivity discrete functional size of the case products is 0.01 and it is 0.05 when the reuse overhead is 0.3.On the other hand, the difference is 0.3 when the reuse overhead is taken as 0.1. The reuse overhead should be in the "0.01 and 0.17" interval, to fit the interval of Jones. When the reuse overhead is identified as 0.18 or more, the productivity values exceed the acceptable boundaries. The reuse overhead values and the changing productivity values can be found in Table 22. These productivities are calculated changing the reuse overhead values on the Discrete Similarity Reflected Size formulas.

Therefore, we can not say that a 0.1 reuse overhead is the most suitable reuse overhead value. Although a 0.1 reuse overhead is acceptable, further analysis should be performed with larger data sets to analyze the accurate overhead value.

Table 22 Productivities of the Cases Calculated Based on DS and Varying Reuse Overhead Values

Reuse Overhead	CN	SN	TN	AN	BN	KN
0.01	0.049	0.031	0.034	0.042	0.059	0.051

Table 22 (Cont.)

0.02	0.049	0.032	0.035	0.042	0.060	0.052
0.03	0.050	0.032	0.036	0.042	0.060	0.054
0.04	0.050	0.032	0.037	0.043	0.061	0.056
0.05	0.050	0.033	0.038	0.043	0.061	0.057
0.06	0.051	0.033	0.039	0.043	0.062	0.059
0.07	0.051	0.033	0.039	0.044	0.063	0.061
0.08	0.052	0.033	0.040	0.044	0.063	0.062
0.09	0.052	0.034	0.041	0.045	0.064	0.064
0.1	0.053	0.034	0.042	0.045	0.064	0.066
0.11	0.053	0.034	0.043	0.045	0.065	0.067
0.12	0.053	0.035	0.043	0.046	0.065	0.069
0.13	0.054	0.035	0.044	0.046	0.066	0.071
0.14	0.054	0.035	0.045	0.046	0.066	0.073
0.15	0.055	0.036	0.046	0.047	0.067	0.074
0.16	0.055	0.036	0.047	0.047	0.067	0.076
0.17	0.055	0.036	0.047	0.048	0.068	0.078
0.18	0.056	0.036	0.048	0.048	0.068	0.079
0.19	0.056	0.037	0.049	0.048	0.069	0.081
0.2	0.057	0.037	0.050	0.049	0.069	0.083
0.21	0.057	0.037	0.051	0.049	0.070	0.084
0.22	0.058	0.038	0.052	0.049	0.071	0.086
0.23	0.058	0.038	0.052	0.050	0.071	0.088
0.24	0.058	0.038	0.053	0.050	0.072	0.089
0.25	0.059	0.038	0.054	0.051	0.072	0.091
0.26	0.059	0.039	0.055	0.051	0.073	0.093
0.27	0.060	0.039	0.056	0.051	0.073	0.095
0.28	0.060	0.039	0.056	0.052	0.074	0.096
0.29	0.060	0.040	0.057	0.052	0.074	0.098
0.3	0.061	0.040	0.058	0.052	0.075	0.100

The other objective of this study was to evaluate the efficiency of the functional similarity quantification methods and assess the effort required to identify functional similarities.

When we first started to apply the method of Santillo and Abran to the case product KN, as a part of the first case study; our aim was to observe the applicability of the method in large projects; since the defined methodology of Santillo and Abran was verified only small scale projects. Although only for the initial part of the case, which was approximately 75 CFP, similarities were identified among functional processes by evaluating the measurement results (which does not include the data preparation time) took 7 hours. Since KN had 136 functional processes and 1996 DM-DG tuples; 1996*1995 numbers of comparisons is required for the evaluation of the functional processes. This means approximately 103 hours are required for comparison of 1996*1995 tuples when our initial productivity is considered. As Santillo and Abran emphasized, this process is very time consuming; besides, it is impossible to accomplish such a job without errors.

In order not to evaluate 1996*1995 action types one by one, the best solution was to develop a tool that automatically calculates the similarity percentage of the functional processes. With the automation of the process, comparison time was decreased to seconds and possibility of occurrence of an error was decreased. Since the SR Tool provided a significant improvement to constitute a similarity matrix; we extended its applicability to be calculating the DS and CS sizes automatically.

Although we developed the SR tool and provided a significant improvement on the time required for detection of similar functions and constitution of the similarity matrices; it still takes time to prepare the measurement results to be the input for SR Tool, if they are not constructed based on the constraints of the Matlab Code initially. Measurement results are arranged so as to include the “functional process name”, “data movement number”, “data group description”, and “data movement type” information in the cells of an Excel table. The important point to care about is to write all of the information within the cells without any space.

The efforts required for Functional Similarity calculation for each case product is given on the 4th column of Table 23. The effort required for functional size measurement highly depends on the experience of the measurer, complexity of the measured product and well defined requirement documents; therefore, we should search for a correlation between the magnitudes and the efforts for FS Calculation of the products instead of the measurement time and effort of functional similarity calculation. Even if a linear correlation is not observed between functional size and effort for functional similarity calculation; we observed that when the size of the product increases the effort required for FS calculation increases.

Table 23 Efforts Required for Functional Similarity Calculation and Functional Size Measurement

Case Product	Functional Size	Effort for Functional Size Measurement (minute)	Effort for Functional Similarity Calculation (minute)
CN	108	600	25
SN	76	480	20
TN	156	900	30
AN	281	480	45
BN	178	240	30
KN	1996	5400	300
DN	217	2400	35
MN	208	1200	35

Besides the direct observations, we made some indirect observations based on the results of the case studies. One of them is the distributions of the productivity values given in Figure 3 and Figure 4. The considerable variances on the Productivity#1 and 2 in Figure 3 and Figure 4 revealed how the functional size and effort relation is a problematic area and this subject is worth to work on it.

Another indirect observation was the negligible difference between productivity values attained by the effort values in which the supporting processes are included and the effort values in which the supporting processes are excluded in Table 19 and Table 20. This negligible difference between the standard deviation values of CS on Table 19 (0.01265) and on Table 20 (0.01400) can also be used as an indicator of how the quality procedures are implemented in these organizations. For the case product KN, although the effort of supporting processes is 22.4% of the total effort, it has the highest productivity value. This can also be used as an indicator to show that the 4th organization in which the KN developed, utilizes mature processes.

The last indirect observation is related to the granularity level and the functional similarity relation. Based on the measurement structure of COSMIC, the functional similarities are identified only to the data group level. However, if the attributes within these data groups were known, the probability of analyst to make an error would be decreased, and the similarities which can be unnoticed would be detected. This situation was observed during the analysis of the case product KN. Two of the functional processes of KN are the constitution of Entities and constitution of Actor model elements. Although these two functional processes are 100% similar, they can be treated as two different processes and can be measured separately because of the granularity level that the COSMIC provided, however, the effort required to develop the second functional process is not the same as the first FP.

For the case products that the reliable effort values couldn't be attained, the LOC and functional size values were compared in order to observe the effect of the SiRFuS.

Table 24 LOCs per CFP for the Cases

Case Product	LOC/PS	LOC/ASR	LOC/DFS	LOC/CFS
AN	127.75	146.46	215.46	223.86
BN	102.64	118.59	176.09	179.99
KN	45.90	52.57	134.28	152.20
DN	55.70	62.81	117.18	121.97
MN	94.66	108.42	184.06	195.14
Std Deviation	33.98	39.31	39.57	39.21

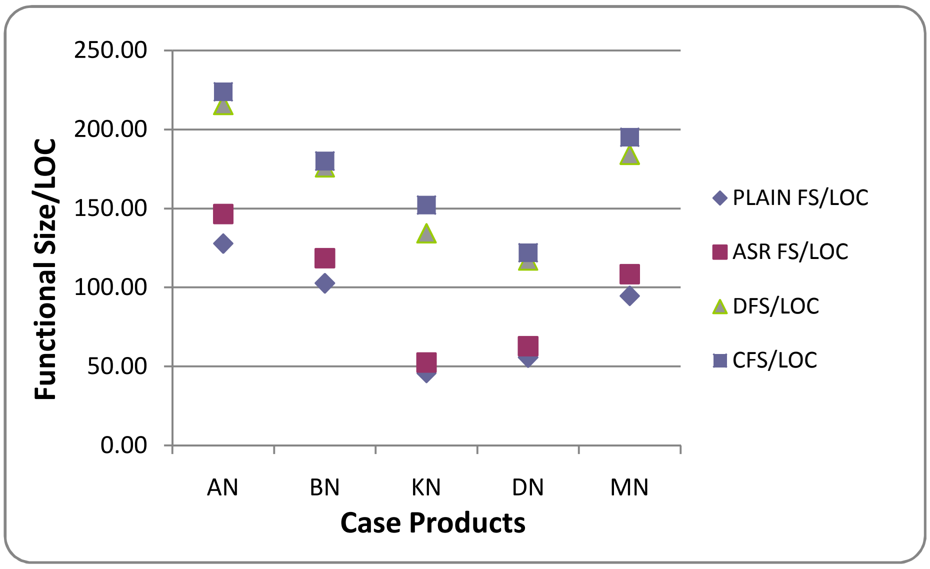

Figure 5 Distributions of the LOCs per CFPs of the Case Products

Six of the case products have been developed using JAVA and the remaining case product has been developed using C# programming language. According to the data, stated by Quantitative Software Management, Inc in (http://www.qsm.com/FPGearing.html, April 2005), JAVA and C# are assumed to have the same numbers of SLOCs for the same functionality. Therefore we used the LOC values as it is for comparison.

When the Figure 5 is analyzed, it seems as if a significant improvement couldn't not be provided on the CS and DS values which were calculated by the application of the SiRFuS method. However, the ratio between the highest point value and the lowest point value of the Plain Size and the highest point value and the lowest point value of the Continuous Functional Size was decreased from 2.78 to 1.83. The reason that we failed to observe the improvement on the correlation between LOC and the effort may base on the counting style of the LOCs of the products. Case products; DN, MN and KN includes no comments, blank lines or library; on the other hand, we don't know if the LOC of the case products AN and BN were counted based on the same rules.

Based on all of the discussion above we can conclude that it is a necessity to take into consideration the functional similarities if the correlation of functional size and development effort is desired which leading to successful planning, monitoring and controlling of the software projects successfully. Besides this significant improvement, functional similarity identification method can be used for the organizations which wonder the similarity of their products.

CHAPTER 5

CONCLUSIONS

In this chapter, the results and the contribution of our study is explained briefly.

5.1 Conclusions

In this thesis study, we studied the problem of functional size and effort correlation, since the underestimated and overestimated effort values are one of the major causes of failure of a software project (Tucker & Boehm, 2002).

Since the functional size is the primary input for effort estimation of a product, we deal with the problem of the adjustment of COSMIC functional size results. We thought that one of the reasons of the underestimated and the overestimated effort for the software development is, overlooking the important issues that have impact on the functional size such as functional similarities, varying complexity of the algorithms, differences in the software development environment, expertise of the developers etc...

We have conducted a single case study to analyze the problem and identify the improvement opportunities. With the enlightenment of the results of the case study, we decided that similarity of the functions enlarges the functional size, leading to unrealistic effort estimation.

Our solution to the problem is based on the adjustment of the COSMIC functional size measurement results by calculating the functional similarities within product by evaluating the functional size measurement results. We have developed the

Similarity Reflective Functional Size Methodology, SiRFuS, which involves identification of the similar functions, based on the method of Santillo and Abran and calculation of the similarity reflective functional sizes; Discrete and Continuous Functional Size.

Discrete Similarity Reflective Functional Size (DS) of a product is calculated using constant functional similarity percentage values, based on the interval in which the functional similarity value belongs. The constants used within the DS formulas are derived from the software enhancement approach of NESMA (NESMA, 2001). On the other hand Continuous Similarity Reflective Functional Size (CS) is calculated using variable functional similarity percentage values which are derived from the functional similarity matrix.

When we used the COCOMO II Model, we observed that estimated efforts have significant deviations from the actual efforts. Deviations change between 5.6 man-months to 362.3 man-months. We think there might be three reasons for the failures of effort estimation: the impact of functional similarities, problems during conversion from functional size to LOC, and differences between the assumptions and the real situations for the scale factors and effort multipliers.

To verify the applicability of our method, we evaluated the method in eight case products. Although we had very few points for the comparison of the relation between functional size and the effort, we were able to observe significant improvements on this relation. The improvements have been observed by comparing the productivity values of the case products which were calculated using both adjusted and unadjusted functional size values. The productivity values which were calculated by using adjusted functional size values, DS and CS, had an interval from 5.15 to 11.9 function points per month and from 5.43 to 10.5 function points per month respectively. On the other hand; the productivity values which were calculated by using unadjusted functional size values, PS, change from 9.6 to 34.8. The reference productivity values provided by Jones (Jones, 1998), range from 1.9 to 13.88 function points per month. As can be observed, the productivity values attained using DS and CS lie within the interval given by (Jones, 1998), while the productivity values attained using PS are out of the bounds. Although the ASR is an adjusted functional size, it is not in the bounds of the reference values, since the interval of the productivity attained by using ASR changes from 7.2 and 30.4; which also supports the idea that the average functional similarities does not reflect the real similarity potential of the product.

The case study results revealed that there is not a big difference between the DS and CS sizes; however, we recommend the usage of Continuous Similarity Reflective Functional Size for the determination of the required effort. Because, although the functional similarities of the case products approximated to the constant percentage values in this study; this is not a general rule. When the functional similarities show significant changes based on the complexity of the application, the constant similarity values which are derived from the NESMA enhancement approach, may not converge with the continuous similarity values.

We assumed that a reuse overhead is essential to calculate similarity reflective DS and CS sizes. Because of this is; even if two functional processes are 100% similar, an effort is still required for the investigation and the reuse of one of the functional processes. Therefore, at first we had identified the reuse overhead value as 0.1. After that we investigated the most suitable reuse overhead value which provides the best linear productivity line. After the analysis, we observed that none of the reuse overhead values between 0.01 and 0.17 provided superiority to another, although, the difference between the highest point and the lowest point of the productivity discrete functional size of the case products decreased, as the reuse overhead decreased. Although a 0.1 reuse overhead value is acceptable, further analysis should be performed with larger data sets to analyze the accurate overhead value.

One of the challenges of the functional similarity identification is that the implementation process is error prone and requires too much effort. Observing this problem, we developed the SR Matlab Tool to automate the process and to decrease the possibility of occurrence of an error. In addition to this, the comparison time was decreased to a few seconds. The program takes the COSMIC functional size measurement results as input, and generates the functional similarity matrix, DS and CS results of the product being analyzed.

Although the method of Santillo and Abran is suitable for the identification of functional similarities; their research was limited with the evaluation process. Their approach is based on the identification of the functional similarities and calculation of the average, minimum and maximum similarity percentages for the whole product which are later used for the internal reuse decisions. Since we observed that average similarity of the product does not reflect the real potential of the product for reuse; we evaluated a new similarity reflective functional size value for every functional process in our study. By identifying adjusted functional sizes for every functional process; we made possible the identification of the adjusted functional sizes of each work package. When the goal is; to plan, execute and monitor software projects successfully, the size of each work package has a considerable significance. Since the functional size can be defined more reliably with the method of SiRFuS, the effort required for the work packages will also be estimated more reliably.

The methodology SiRFuS has also some weaknesses, in addition to the strengths explained above. In our study we assumed that the cases are developed under the same conditions, although some of them are not. When the SiRFuS method is applied within organizations in the long term, and organizations' own historical data are collected, the formulas and the structure of the method can be validated more precisely with larger data sets.

Although the SR Tool, provided a significant improvement on the process of evaluation and calculation of the functional similarities, it is still error prone. Since, it identifies the similarities by comparing the names within the cells, the names should be written identically and without any space. If the analyst leaves a space by mistake between the functional processes or the data group descriptions, the tool perceives each separate name as separate functional process or data group, leading to incorrect results. Therefore the results should be checked carefully.

Although significant improvements were observed on the deviations of the productivity values, they still vary. The reasons of the variances may be the varying complexity of the algorithms, differences in the software development environment, and different level of expertise of the developers, which are not in the scope of thesis study.

As a result, this study has three major contributions to the field of software project management; indication of the significance of the identification of the functional similarities for the adjustment of the functional sizes; the development the SiRFuS methodology, which provides the correlation of the functional size; and effort and the development of the SR Tool, which partially automates the method and decreases the analysis time.

5.2 FUTURE RESEARCH SUGGESTIONS

We have used the CHAR method to define the functional domains of the case products in this research. However, we had a chance to evaluate the SiRFuS only on Information Systems and Complex Data Driven Systems Projects, since the data we have was limited. Therefore, the applicability of the method should be verified on the other functional domains defined in the CHAR Method, such as Controlling Calculation Systems, Scientific Information Systems, and Scientific Controlling Data Processing Systems.

Although we have improved the correlation between functional size and effort, there is still need to identify other factors obstructing the exact correlation. More research is needed to be conducted to identify these factors. One of these factors can be the complexity of the algorithms within the functional processes. The granularity level of functional size measurement methods does not evaluate this level of information. Therefore, SiRFuS should be refined to consider the complexity of the algorithms besides functional similarities.

The accuracy of the method can be verified within the organizations with much more data with various functional domains.

In the scope of the thesis we only evaluated the internal reuse of the similar functions, however, the external reuse can be evaluated within the projects in the same organizations with the SiRFuS Method.

The functional similarities were identified only considering the functional processes, however, some of the functions may be more similar when they are compared with clustering (Ozcan Top, et al., 2008).

We assumed that the similar functions will be used within the product; however the effectiveness of the reuse of the similar functions should be analyzed.

The correlation of the adjusted functional size values and the effort and cost models should be analyzed.

The method which is based on the measurement of the case products by COSMIC can be extended to be used with other common functional size measurement methods such as IFPUG and MkII.

REFERENCES

Abran, A., & Desharnais, J. M. (1995). Measurement of Functional Reuse in Maintenance. *Journal of Software Maintenance: Research and Practice, 7*(4), 263-277.

Abran, A., & Maya, M. (1997). Measurement of Functional Reuse. *WISR8, Ohio State University, Columbus, Ohio, USA, March*, 23-26.

Albrecht, A. (1979). Measuring Application Development Productivity. *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, 83*, 92.

Albrecht, A., & Gaffney Jr, J. (1983). Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering, 9*(6), 639-647.

Anderson, J., Branch, E., Luedtke, T., Carson, S., Falconi, J., & Janda, R. (1999). *Parametric Estimating Handbook*: Reinvention Laboratory, DOD, NASA.

Banker, R., Kauffman, R., & Zweig, D. (1993). Repository evaluation of software reuse. *IEEE Transactions on Software Engineering, 19*(4), 379-389.

Boehm, B. (1981). *Software engineering economics*: Prentice-Hall Englewood Cliffs, NJ.

Boehm, B., Abts, C., Horowitz, E., & Madachy, R. (2000). *COCOMO II Model Definition Manual*: Center for Software Engineering, USC.

Consortium, T. C. S. M. I. (2005). Guideline for Sizing Business Applications Software Using COSMIC-FFP, Version 1.0.

Cruickshank, R., & Gaffney, J. (1992). A software cost model of reuse within a single system. *MITRE-Washington Econ. Analysis Ctr. Conf. on Analytical Methods in Software Eng. Econ. II, Washington, DC, July*.

Demirors, O. (2008). Software Management Lecture Notes.

Desharnais, J., Abran, A., & Cuadrado-Gallego, J. (2006). Convertibility of Function Points to COSMIC-FFP: Identification and Analysis of Functional Outliers. *MENSUR A*.

Dillibabu, R., & Krishnaiah, K. (2005). Cost estimation of a software product using COCOMO II. 2000 model?a case study. *International Journal of Project Management, 23*(4), 297-307.

Ergüden, E. (2008). *Application of the Unification Model For Functional Size Measurement Methods: A Case Study in WEB Applications.* Middle East Technical University, Ankara.

Fenton, N. (1991). *Software Metrics: A Rigorous Approach*. London, UK: Chapman & Hall, Ltd

Frakes, W., & Terry, C. (1996). Software reuse: metrics and models. *ACM Computing Surveys (CSUR), 28*(2), 415-435.

Gencel, C. (2005). *An Architectural Dimensions Based Software Functional Size Measurement Method.* Middle East Technical University, Ankara.

Gencel, C., & Demirors, O. (2008). Functional size measurement revisited. *ACM Transactions on Software Engineering and Methodology, 17*(3).

Group, O. M. (2005). Unified Modelling Language (UML) v2.0.

Ho, V., Abran, A., & Oligny, S. (2000). Using COSMIC-FFP to Quantify Functional Reuse in Software Development: ESCOM-SCOPE.

Horowitz, E. (1994). USC COCOMO Reference Manual.

http://sunset.usc.edu/csse/research/COCOMOII/cocomo81.htm (2008). COCOMO 81

http://sunset.usc.edu/research/COCOMOII/expert_cocomo/expert_cocomo2000.html (2000). COCOMO II with Heuristic Risk Assessment

http://www.qsm.com/FPGearing.html (April 2005). LOC compatibility: Quantitative Software Management Function Point Programming Languages Table.

IEEE (1998). IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications.

ISBSG (2007). Data Collection Questionnaire- New development, redevelopement, or enhancement (COSMIC FFP) v5.10.

ISO/IEC (1998). 14143-1: Information Technology – Software Measurement - Functional Size Measurement - Part 1: Definition of Concepts.

ISO/IEC (2002a). 14143-2: Information Technology – Software Measurement - Functional Size Measurement - Part 2: Conformity Evaluation of Software Size Measurement Methods to ISO/IEC 14143-1:1998.

ISO/IEC (2002b). 14143-4: Information Technology – Software Measurement - Functional Size Measurement - Part 4: Reference Model.

ISO/IEC (2002c). IS 20968:2002: Software Engineering - MK II Function Point Analysis - Counting Practices Manual.

ISO/IEC (2003a). 14143-3: Information Technology – Software Measurement - Functional Size Measurement - Part 3: Verification of Functional Size Measurement Methods.

ISO/IEC (2003b). 19761:2003: Software Engineering - COSMIC-FFP: A Functional Size Measurement Method.

ISO/IEC (2003c). IS 20926:2003: Software Engineering - IFPUG 4.1 Unadjusted Functional Size Measurement Method - Counting Practices Manual.

ISO/IEC (2004). IS 14143-5 Information Technology – Software Measurement - Functional Size Measurement - Part 5: Determination of Functional Domains for Use with Functional Size Measurement.

ISO/IEC (2005a). IS 14143-6: Guide for the Use of ISO/IEC 14143 and related International Standards.

ISO/IEC (2005b). IS 24570:2005: Software Engineering – NESMA functional size measurement method Ver.2,1- Definitions and counting guidelines for the application of FPA.

ISO/IEC (2008). 29881:2008 Information Technology-- Software and systems engineering—FİSMA 1.1 functional size measurement method.

Jones, C. (1998). *Estimating Software Costs*: McGraw-Hill Companies.

Jørgensen, M. (2004). Top-down and bottom-up expert estimation of software development effort. *Information and Software Technology, 46*(1), 3-16.

Jørgensen, M., Indahl, U., & Sjøberg, D. (2003). Software effort estimation by analogy and "regression toward the mean". *The Journal of Systems & Software, 68*(3), 253-262.

Karagöz, A. (2008). *A Framework For Developing Conceptual Models of the Mission Space For Simulation Systems.* Middle East Technical University, Ankara.

Kemerer, C. (1987). An empirical validation of software cost estimation models. *Communications of the ACM, 30*(5), 416-429.

Krueger, C. (1992). Software reuse. *ACM Computing Surveys (CSUR), 24*(2), 131-183.

Leach, R. (1996). Methods of Measuring Software Reuse for the Prediction of Maintenance Effort. *Journal of Software Maintenance*.

Leung, H., & Fan, Z. (2002). Software Cost Estimation. *Handbook of Software Engineering, Hong Kong Polytechnic University*.

Masse, R. (1997). Software Metrics: An Analysis of the Evolution of COCOMO and Function Points, from http://www.rogermasse.com/papers/software-metrics/

MathWorks (2007). MATLAB R-2007.

Meli, R. (2000). Functional and technical software measurement: Conflict or integration. *FESMA-AEMES Conf*, 18-20.

NESMA (2001). Function Point Analysis for Software Enhancement.

Ozcan Top, O., Tunalilar, S., & Demirors, O. (2008). *Evaluation of the Effect of Functional Similarities on Development Effort*. Paper presented at the EuroMicro SEAA.

Ozkan, B., Turetken, O., & Demirors, O. (2008). *Software Functional Size: For Cost Estimation and More.* Paper presented at the EuroSPI, Dublin, Ireland.

Santillo, L., & Abran, A. (2006, May 10-12). *Software Reuse Evaluation based on Functional Similarity in COSMIC-FFP Size Components.* Paper presented at the Software Measurement European Forum, SMEF, Rome, Italy.

Santillo, L., & Della Noce, I. (2005). *A Worked Function Point model for effective software project size evaluation.* Paper presented at the SMEF 2005, Rome(Italy).

Software, P. (2008). Practiline Source Code Line Counter v1.1.

Tucker, A., & Boehm, B. (2002). Point/Counterpoint: On the Balance between Theory and Practice/Software Engineering Is a Value-Based Contact Sport. *IEEE Software, 19*, 94-97.

Turetken, O., Demirors, O., Ozcan Top, O., & Ozkan, B. (2008). The Effect of Entity Generalization on Software Functional Sizing: A Case Study *Product-Focused Software Process Improvement* (Vol. Volume 5089/2008, pp. 105-116): Springer Berlin / Heidelberg.

Urgun, E. (2008). *A Detailed Evaluation COSMIC-FFP AND IFPUG-FPA Conversion Approaches.* Middle East Technical University, Ankara.

Valerdi, R., Chen, Y., & Yang, Y. (2004). *System Level Metrics for Software Development Estimation*. Paper presented at the International Symposium on Empirical Software Engineering, ISESE.

Şentürk, O. (2008). *A Case Study on the Unification Model for Functional Size Measurement Methods: Enterprise Solutions Web Applications* Middle East Technical University, Ankara.

APPENDICES

APPENDIX A: MOVIE MANAGER

The application shall maintain the following information:

Movies: The application shall maintain a unique id, the movie title, year of production, Production Company & genre of the movies. The genre can be of the following type *or* a combination of these types: Comedy, thriller, animation, documentary, science-fiction, action, horror, drama, musical and western.

Movies shall also have director, producer, writer and cast information where all can have more than one records each.

Person: The application shall maintain a unique id, name of the person and date of birth & place of birth.

A person might be acting as an actress/actor, or might be a producer, writer or director of the movie. In relation to a movie, it is also possible for a person to be all or a combination of these (both writer and director, etc.).

- If a person is an actress/actor in a movie, the application shall also maintain the character name in the movie.
- A producer shall also be noted whether he/she is the co-producer, executive producer or just the producer.

- A writer shall also be noted whether he/she is the story writer, screenplay writer or both?
- There is no additional attributes to be maintained for directors.

The functional requirements to be measured:

1. The application shall enable the entry and update of persons. For updates, first the application shall provide a list of all persons. Once a person is selected, the application shall display the details of the person on an editable form.
2. The application shall enable the entry of movie information. Genre shall be entered via a drop-down list. Similarly, for producer, director, writer and cast information, persons shall be selected among the ones in the application via drop-down lists.
3. The application shall enable an enquiry of movies over the title and the year the movie is produced. The application shall list the title and the year of the movies that match with the query parameters. Once user selects a specific movie, details of the movie shall be listed. The output shall include the following information:
    a. title, year of production, production company, genre(s)
    b. director(s),
    c. producer(s) [co-/executive],
    d. writer(s) [story/screenplay/story & screenplay],
    e. cast (person name, character name)
4. The application shall enable an enquiry of persons over the name. The application shall list the name of the persons that match with the query parameter. Once the user selects a specific person, details shall be listed. The output shall include the following information:

a. name, date of birth & place of birth

b. movies directed,

c. movies produced (with co-/executive/producer indicated),

d. movies written (with story/screenplay/story & screenplay indicated),

e. movies acted (with character name indicated)

5. The application shall enable the deletion of movies. First the application shall provide a list of all movies. Once a movie is selected, the application shall delete all related information from its database and return to the list as a confirmation.

Table 25 COSMIC Measurement Results of Movie Manager

FP ID	FP Name	Number	DM	DG
1	AddPerson	1	Entry	Personinfo
	AddPerson	2	Write	Personinfo
	AddPerson	3	Exit	Error/Confirmation
2	ListPersons	4	Entry	Listpersonsrequest
	ListPersons	5	Read	Personinfo
	ListPersons	6	Exit	Personinfo
3	RetrievePerson	7	Entry	Retrivepersondetailsrequest
	RetrievePerson	8	Read	Persondetailsinfo
	RetrievePerson	9	Exit	Persondetailsinfo
4	UpdatePerson	10	Entry	Personinfo
	UpdatePerson	11	Write	Personinfo
	UpdatePerson	12	Exit	Error/Confirmation
5	AddMovie	13	Entry	MovieInfo
	AddMovie	14	Read	Personinfo
	AddMovie	15	Exit	Personinfo
	AddMovie	16	Entry	Writerinfo
	AddMovie	17	Entry	Producerinfo
	AddMovie	18	Entry	Castinfo
	AddMovie	19	Entry	Directorinfo
	AddMovie	20	Write	MovieInfo
	AddMovie	21	Write	Writerinfo
	AddMovie	22	Write	Producerinfo
	AddMovie	23	Write	Castinfo
	AddMovie	24	Write	Directorinfo
	AddMovie	25	Exit	Error/Confirmation
6	QueryMovie	26	Entry	QueryParameters1
	QueryMovie	27	Read	MovieInfo
	QueryMovie	28	Exit	MovieInfotitleyear

Table 25 (Cont.)

7	ListMovieDetails	29	Entry	Selectionofthemovie
	ListMovieDetails	30	Read	MovieInfo
	ListMovieDetails	31	Read	Writerinfo
	ListMovieDetails	32	Read	Producerinfo
	ListMovieDetails	33	Read	Castinfo
	ListMovieDetails	34	Read	Directorinfo
	ListMovieDetails	35	Read	Personinfo2
	ListMovieDetails	36	Exit	MovieInfo
	ListMovieDetails	37	Exit	Writerinfo
	ListMovieDetails	38	Exit	Producerinfo
	ListMovieDetails	39	Exit	Castinfo
	ListMovieDetails	40	Exit	Directorinfo
	ListMovieDetails	41	Exit	Personinfo2
8	QueryPerson	42	Entry	QueryParameters2
	QueryPerson	43	Read	Personinfo
	QueryPerson	44	Exit	Personinfo (name)
9	ListPersonDetails	45	Entry	Selectionoftheperson
	ListPersonDetails	46	Read	Personinfo
	ListPersonDetails	47	Read	Writerinfo
	ListPersonDetails	48	Read	Producerinfo
	ListPersonDetails	49	Read	Castinfo
	ListPersonDetails	50	Read	Directorinfo
	ListPersonDetails	51	Read	MovieInfo
	ListPersonDetails	52	Exit	Personinfo
	ListPersonDetails	53	Exit	Writerinfo
	ListPersonDetails	54	Exit	Producerinfo
	ListPersonDetails	55	Exit	Castinfo
	ListPersonDetails	56	Exit	Directorinfo
	ListPersonDetails	57	Exit	MovieInfo
10	ListMovies	58	Entry	Requestforalistofmovies
	ListMovies	59	Read	MovieInfo
	ListMovies	60	Exit	MovieInfo
11	DeleteMovie	61	Entry	Selectionofthemovie
	DeleteMovie	62	Write	MovieInfo
	DeleteMovie	63	Write	Writerinfo
	DeleteMovie	64	Write	Producerinfo
	DeleteMovie	65	Write	Castinfo
	DeleteMovie	66	Write	Directorinfo
	DeleteMovie	67	Read	MovieInfo
	DeleteMovie	68	Exit	MovieInfo

APPENDIX B: SR TOOL PROGRAM CODE

```
function sr_discrete_continuous(filename)
fid = fopen(filename);
C = textscan(fid,'%s%s%s%s');
fclose(fid);
a=size(C{1,1});
lenght=a(1);
a1=C{1,1};
a3=C{1,3};
a4=C{1,4};
m1=zeros(lenght,1);
m1(1)=1;
m2=eye(lenght,lenght);
temp=1;
k=1;
for i=2:lenght
   if(strcmpi(a1(i-1),a1(i))==0)
   k=k+1;
   end
   m1(i)=k;
end
t=1;
for i=2:lenght
  as=m1(i)-m1(i-1);
  if (as==0)
     temp=temp+1;
  end
  if (as==1 || i==lenght)
```

```
        say(t)=temp;
        temp=1;
        t=t+1;
   end
 end
m3=zeros(m1(lenght),m1(lenght));
 m2=inv(m2);
%
for i=1:lenght
   for j=i:lenght
    if(strcmpi(a3(i),a3(j))==1 && strcmpi(a4(i),a4(j))==1)
                 if (i~=j  )
             m3(m1(i),m1(j))=m3(m1(i),m1(j))+m2(i,i);
        m3(m1(j),m1(i))=m3(m1(i),m1(j));
     end
     end
   end
end
 for i=1:m1(lenght)
m4(i,:)=m3(i,:)./say(i);
m4(i,i)=1;
end
m5=zeros(1,m1(lenght));
%
m5(1)=say(1);
 %%%%%%%%     continuous   %%%%%%%%%%
 for i=2:m1(lenght)
mak=max(m4(i,1:i-1));
 if (mak==1)
  m5(i)=say(i)*0.1;
elseif (mak==0)
```

```
   m5(i)=say(i);
else
   m5(i)=say(i)*mak*0.1 + say(i)*(1-mak);
end
 end
 %%%%%%%%      discrete      %%%%%%%%
 m6=zeros(1,m1(lenght));
%
 m6(1)=say(1);
 for i=2:m1(lenght)
mak=max(m4(i,1:i-1));
 if (0<mak && mak<=0.34)
   m6(i)=say(i)*0.25*0.1 + say(i)*0.75;
elseif (0.34<mak && mak<=0.67)
      m6(i)=say(i)*0.5*0.1 + say(i)*0.5;
elseif (0.67<mak && mak<1)
            m6(i)=say(i)*0.75*0.1 + say(i)*0.25;
elseif (mak==1)
   m6(i)=say(i)*0.1;
else
   m6(i)=say(i)*1;
              end
       end
m5
m6
 dlmwrite('similarity_result.txt',m4,'\t');
dlmwrite('continuous_result.txt',say,'\t');
dlmwrite('continuous_result.txt',m5,'-append','delimiter','\t');
dlmwrite('discrete_result.txt',say,'\t');
dlmwrite('discrete_result.txt',m6,'-append','delimiter','\t');
```