A UML-based Simple Function Point Estimation Method and Tool

Geng Liu, Xingqi Wang, Jinglong Fang
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou, China
email:{liugeng, xqwang, fjl}@hdu.edu.cn

***Abstract*—Function Point Analysis (FPA) is used to measure the size of functional user requirements of software applications. However, the measurement process of FPA is slow, expensive and complex. The Simple Function Point (SiFP) method has been proposed as a replacement of FPA that is much faster and cheaper to apply. However, no tools supporting Simple Function Point measurement have yet been proposed. In this paper, we aim at building a tool to facilitate SiFP measurement. Specifically, we propose a measurement based on UML models of requirements, including use case diagrams and domain model (class diagrams). The proposed methodology –including a set of guidelines for domain modeling and the mapping between SiFP measure components and UML elements – makes SiFP measurement much easier to perform. In fact, the proposed methodology is usable in the early requirements definition stage, when only use case diagram and the primary class diagram illustrating the domain model (including classes' names and relationship among classes) are available. We used 17 academic sample applications to validate our proposal. The result shows that our method and tool can be used to replace manual Simple Function Point measurement in the early phases of the software development cycle to measure the functional size of software project.**

Keywords— Functional Size Measures; Simple Function Point; SiFP; UML; Object-oriented measures.

I. INTRODUCTION

Function Point Analysis (FPA) [1][2][3] aims at measuring the size of Functional User Requirements (FUR) of software applications. Being based on FUR, which are available in the early phases of development, these measures are widely used to estimate the effort required to develop software applications. FPA was originally introduced by Albrecht to measure data-processing systems by quantifying the functionality the software provides to the user, from the information view, by quantifying the volume of data flow and the storage [4].

The basic idea of FPA is that the "amount of functionality" released to the user can be evaluated by taking into account the data used by the application to provide the required functions, and the transactions (i.e., operations that involve data crossing the boundaries of the application) through which the functionality is delivered to the user. Data are user identifiable groups of logically related data, and are classified as Internal Logical Files (ILF) or External Interface Files (EIF). A transaction is a set of actions seen as one cohesive unit of work. FPA differentiates three types of transactions: External Input (EI), External Output (EO), and External Inquiry (EQ). The size of each data function depends on the type of contents; the size of each transaction depends on the number of data files used and the amount of data exchanged with the external. The sum of the sizes of data and transactions is the size of the application in Unadjusted Function Points (UFP).

Organizations that develop software are interested in Function Point measurement process that is reliable, rapid and cheap, and that fits well in their development processes. However, performing FPA requires a thorough exploration of FUR, to identify and possibly weigh basic functional components. Therefore, the measurement process can be quite long and expensive. In fact, FPA performed by a certified function point consultant proceeds at a relatively slow pace: between 400 and 600 function points (FP) per day, according to Capers Jones [5], between 200 and 300 FPs per day according to experts from Total Metrics [6]. Consequently, measuring the size of a moderately large application can take too long, if cost estimation is needed urgently. Also, the cost of measurement can be often considered excessive by software developers.

In addition, at the beginning of a project, size estimation would be necessary for bidding and planning. But, FURs have not yet been specified in detail and completely, namely the available information is often incomplete and insufficient. So the customer is only able to do approximate measurements. The accuracy of a FP measure grows with the completeness and precision of FUR specifications. When we can measure with the highest accuracy, we no longer need that measure. The situation is described by the paradox illustrated in Fig. 1.

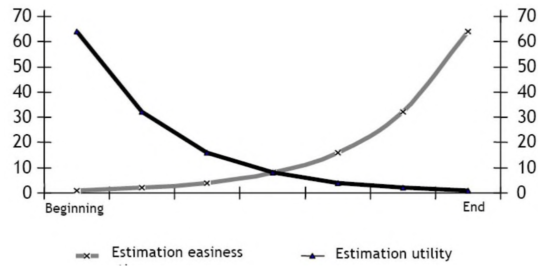


Fig. 1. Paradox of estimation and informations about estimation

Given the above situation, many simplified methods, such as Early & Quick Function Points（E&QFP) [7], Estimated NESMA [8], Simplified FP[9], ISBSG [10], ILF model [11], and Early FP [12], have been proposed. The SiFP method [13][14][27] is different from the other methods mentioned above, as it does not aim at providing approximate estimation of FP measures; rather, it defines a brand new functional size measure, to be used in place of traditional FP.

In this paper, we propose some rules for building UML models in a SiFP-oriented way. Since SiFP counting is based on the identification of Unspecified Generic Elementary Process (UGEP) and Unspecified Generic Data Group (UGDG), which basically correspond to system data and process, we exploit the ability of UML to represent such information by establishing an explicit relation between SiFP elements and UML language constructs. We also define some rules to measure the SiFP size of an application from use case diagrams and the domain model, and develop a tool to automatically measure SiFP on the base of XMI/XML files abstracted from UML model. Throughout the paper we take for granted that the reader knows at least the basics of FPA measurement and is familiar with basic UML concepts.

The rest of the paper is organized as follows: Section II explains the background knowledge about SiFP. Section III describes the empirical study. The validity of the study is discussed in Section IV. Related work is presented in Section V. Finally, Section VI draws some conclusions and outlines future work.

II. BACK GROUND KNOWLEDGE-SIFP

This section presents a brief summary of the SiFP method. For full details and explanations of the method, see the reference manual [13].

SiFP method was proposed by the Simple Function Point Association, Italy. Its basic idea is that a notion of complexity based on the number of logical data file or cross reference among transaction and file or subgroup of data in a file is not significant to the goal of representing functional size and of estimation effort or cost. In order to measure the functional size of an application, it is not necessary to identify several types of transactions and files.

The SiFP method defines the generic software model as shown in Fig.2, which highlights the components related to the functional requirements of "moving" data, "processing" data and data "storage".

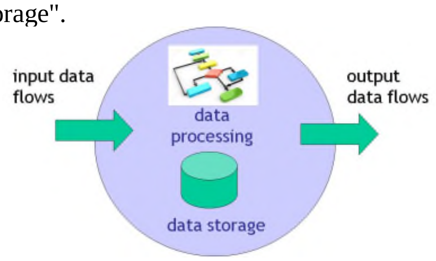


Fig. 2. Theory of SiFP [13]

The SiFP method defines and uses only two basic functional components (BFCs): UGEP and UGDG, see Fig.3. An UGEP is defined as: "*An atomic set of functional user requirements conceived for processing purposes. It refers to an informational or operational goal considered significant and unitary by the user and includes all automated, mandatory and optional activities needed to meet the goal. After an UGEP is concluded, the measurable software application (MSA) to which it belongs must be in a logically consistent state.*" [13] An UGDG is defined as: "*An atomic set of user requirements having a storage purpose. It refers to a single logical data set of interest to the user, for which information must be kept persistently.*"[13]

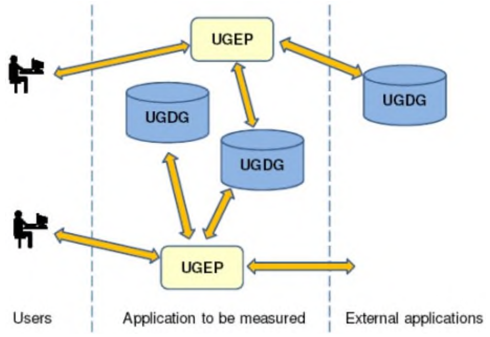


Fig. 3. BFC Types [14]

In the case of the UGEP, the term "unspecified" highlights that it is not necessary to distinguish whether a process is mainly for input, or output, or what is its primary intent of data processing. Similarly, in the case of the UGDG, it means that it is not necessary to distinguish between internal and external logical storage with respect to the boundary of the MSA.

On the other hand, the term "Generic" indicates that for any BFC there is no need to identify subcomponents in order to determine BFC's complexity: all the BFCs weight equally within the same type of BFC. Future developments of the methodology may lead to define different functional weights for each specific BFC depending on elements related to the processing component of transactional BFCs that, at present, is not quantitatively taken into account.

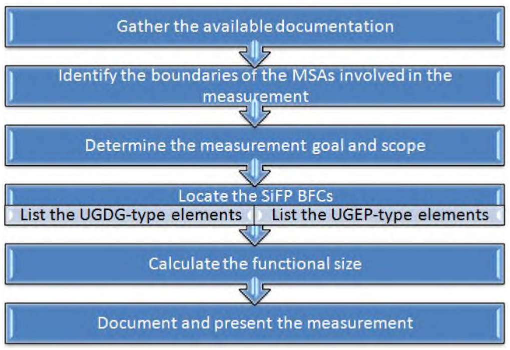

Gather the available documentation
Identify the boundaries of the MSAs involved in the measurement
Determine the measurement goal and scope
Locate the SiFP BFCs
List the UGDG-type elements
List the UGEP-type elements
Calculate the functional size
Document and present the measurement

Fig. 4. SiFP measurement process [13]

The SiFP measurement process is represented in Fig.4. It is a 6-step process:

− Gather the available documentation;

− Identify application boundary;

− Determine the measurement goal and scope;

− Locate elementary processes (UGEP) and logical data files (UGDG);

− Calculate the function size using function SiFP = 4.6 UGEP + 7 UGDG;

− Document and present the measurement.

III. THE EMPIRICAL STUDY

In this section, we introduce UML-based SiFP estimation method through a case study and present briefly the Tool SiFPOOTool developed by us.

A. *The case introduction*

We use as the case a reduced version of a real Information System by Lavazza [15], since it is concise and its size is appropriate. Its functional size in FP is already measured, so we do not need to do it again. In our case, a system class diagram that involves composition and specification/generalization meets our needs. The only drawback of this system for our study is that the use case diagram is relatively simple; the relationships among the use cases just involve the general association. But, overall, it is suitable for our objectives.

This GymIS is an information system for Gym management. The application offers annual and monthly subscriptions. The client who subscripts the annual service only needs to pay 12 times the cost of a month but have the right of receiving 13 months service. The client and subscription data are stored in the system database. The former is characterized by name, age, profession, address, and SSN. Clients can also be updated, but, once inserted, they are never removed from the system. A subscription is characterized by the duration, the subscription date, the subscribing client, the set of optional services to which the client subscribed (their cost adds up to the cost of the basic subscription). Among the optional services there is the possibility to get a monthly medical check.

The functions that the application must provide are the following: record a new client, update the client data, record a new subscription, record the payment by a given client for a given month, compute and print how much is due by every client for the previous months, compute the number of subscriptions that include the given service in a given period, and record the results of a health check. The detailed requirements for the transactions are not presented here. The complete FURs of the GymIS can be found in [15]; they were measured according to FPA rules on the basis of a traditional description. The result was that the application is 67 FP.

B. *SiFP-oriented modeling*

UML-based SiFP estimation method works well only if the given models incorporate all the required information at the proper detail level and the modeling and measure rules are defined according to the SiFP theory. In this sub-section we define the SiFP-Oriented modeling methodology as a set of guidelines. For the purpose of modeling, we use UML as defined in [16]. We do not define extensions or force the semantics of the language. This choice contributes to minimizing the impact of the measurement-oriented modeling on the development process, and to make the adoption of the method as seamless as possible.

Usually, the activity of creating OO models is not sequential; rather, it is often iterative, with several refinements, deletions, extensions, and modifications of the model. In order to keep the presentation clear, we present the modeling methodology as a sequence of conceptual steps.

<u>Step 1: Present application boundary</u>

The first objective of the model is to represent the application boundaries and the external elements that interact with the system to be measured.

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case diagram contains four components. The boundary defines the system of interest in relation to the world around it. The actors are usually individuals involved with the system defined according to their roles. The use cases are the specific roles played by the actors within and around the system. The last component is the relationships between and among the actors and the use cases. UML provides use case diagrams, which are well suited for our purposes. The boundary of the use case diagrams can be directly taken as the boundary of the MSA.

The correspondence between the SiFP concepts and the elements of UML use case diagrams is schematically described in Table I.

TABLE I. MAPPING OF THE ELEMENTS BETWEEN SIFP AND UML

SiFP	UML
Application boundary	Boundary of the object that owns the use cases
UGEP	Use case
User	Actor
UGDG locating out of the system boundary	Actor

<u>Step 2: Present UGEP using use case</u>

Use Case Diagrams indicate –as actors– the elements outside the boundary with which the application interacts; most important, use case diagrams show the transactions. We represent each UGEP as a use case.

Rule 1: Each use case must represent the smallest unit of activity that is meaningful to the user(s), and must be self-contained and leave the business of the application being counted in a consistent state.

Rule 2: Relationship among the use case, extension, include, generalization, must be correctly presented.

Rule 3：A use case that cannot be instanced must be noted as "abstract" stereotype. The base use case of a cluster of use case formatted by generalization must be noted as "abstract" stereotype.

By applying the rules above to the GymIS the use case diagram reported in Fig. 5 is obtained.

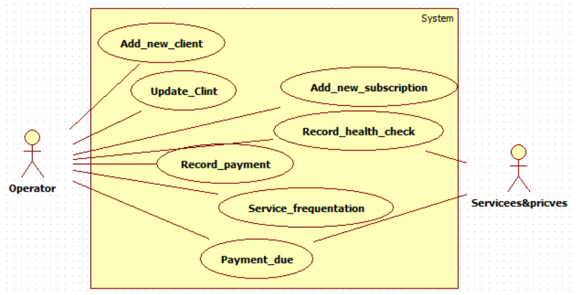


Fig. 5. Use case diagram of the GymIS

<u>Step 3: Present UGDG using domain class</u>

Usually, the methods proposed in the literature for measuring the functional size of UML models map the concept of data functions onto (sets of) classes. The difficulties in matching classes and logic files are exemplified very well in [18], where four different manners of identifying logical files are defined, according to the different possible ways to deal with aggregations and generalization/specializations relationships.

Although in several cases it is possible to identify a class as a logic file, it is also quite common that a single logic file corresponds to a set of classes.

In object-oriented development process, such as ICONIX processes [17], the modeling process of static model can be split into three stages: 1) requirements definition, 2) analysis, conceptual design and technical architecture, 3) design and coding. The obtained models are domain model, updated domain model and Class model -as shown in Fig. 6- which separately correspond to three types of diagram: domain class diagram, updated domain class diagram and class diagrams.

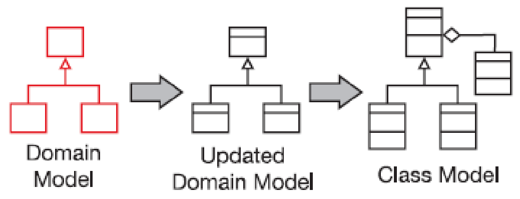


Fig. 6. Static domain model of OO development using ICONIX process

Information presented by domain diagram contains names of the entity objects, and the relationships among these entity objects; updated domain class diagram is added boundary objects and controllers. Also the attributes of each entity class abstracted from use case specification are equipped; Class diagram contains all the information mentioned above, and some controllers are changed into one or more operations and those operations are assigned to corresponding class. Analysis and comparison about different types of objects at different stages is shown in Table II. Through the above analysis we can see the domain class diagram already fully meets the demand for measuring the data "storage" part of SiFP except that it doesn't have the ability to present the UGDG located outside the system boundary.

TABLE II. ANALYSIS ADN COMPARISON ABOUT DIFFERENT TYPES OF OBJECTS

		Domain Model	Update Domain Model	Class Model
Stereotype of Class	Entity	Yes	Yes	Yes
	Controller	/	Yes	Yes
	Boundary	/	Yes	Yes
Information about entity class	Class Name	Yes	Yes	Yes
	Relationship	Yes	Yes	Yes
	Attributes	/	Yes	Yes
	Methods	/	/	Yes
Suitable for SiFP measure		Yes	Yes	Yes

Since for any BFC there is no need to identify subcomponents in order to determine BFC complexity. We define some rules as following:

Rule 4：SiFP does not distinguish between internal and external UGDG, but in order to facilitate the later statements, we divided UGDG into two types: external UGDG and internal UGDG. Internal UGDG is the UGDG that locates inside of the system boundary, external UGDG locates outside of the system boundary.

Rule 5: Entity classes that appear in the domain model diagram are the candidates for UGDG. Entity classes appear in domain model must be complete, namely, no entity class be missed. Each entity class should have its name, and the relationships among the entity classes should be complete.

Rule 6: Each entity class must be noted as stereotype <<Entity>>.

Rule 7: In general, a UGDG corresponds to an entity class (see the class User and Payments in Fig.7). A relevant exception is given by clusters of classes that are connected by composition relations (see the set classes consist of HealthRecord and Result in Fig.7), or generalization relations (see the classes Subscription, MonthScription and AnnualScription in Fig.7). A cluster of classes that are connected by composition or generalization relations is defined as one UGDG.

TABLE III. MAPPING OF THE ELEMENTS

SiFP	UML	Class(es)	#UGDG
UGDG	association	1	1
UGDG	aggregation	1	1
UGDG	composition	a cluster of	1
UGDG	generalization	a cluster of	1
UGDG locating out of the system boundary	logic data component	1	1

Rule 8: When necessary, add to the domain model one or more special class(es) to present the external system logical data: these class(es) are named as external UGDG(s) and are stereotyped <<XUGDG>> (see class otherSystem in Fig.7). A class diagram with added special classes is called an extended class diagram.

By applying the rules above to the Gym IS, the extended class diagram reported in Fig. 7 was obtained.

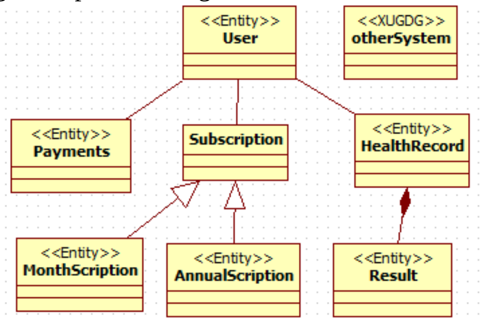


Fig. 7. Extended Class diagram of the Gym IS

C. Counting and summing

Here our SiFP counting procedure is redefined with respect to the UML model with the following goals: it must be coherent with the principles reported in SiFP reference manual [13]; it must be precise, without ambiguities, thus leaving no space to the measurer for interpretation; it must require little effort; it must be executable by people without big skill in FP counting and with little knowledge of the requirements.

As mentioned earlier, a UGEP is represented as a use case, but not every use case should be counted as a UGEP. By analyzing the role and the characteristics of each use case belonging to a set of use cases connected by include, extension or generalization relations (see Table IV), and according to SiFP rules, we define rule 9.

TABLE IV. COMMON ELEMENTS FROM GENERAL MODEL AND FPA ORIENTED UML MODEL

Type of UC	Role of UC	Complete	Abstract	for measure unit
Include	Base UC	Yes	No	Yes
	Inclusion	Yes	No	Yes
Extension	Base UC	Yes	No	Yes
	Extension	Yes	No	Yes
Generalization	general UC	No	Yes	No
	Specialized UC	Yes	No	Yes

Rule 9：In general, a use case is counted as a UGEP. A relevant exception is that the use case noted as abstract is not counted as a UGEP.

Rule 10: As defined by the rules 4-8, whether it is a single class or a group of classes, as long as it is defined as a UGDG, it is counted as a UGDG.

Rule 11：A class stereotyped <<XUGDG>> is counted as a UGDG.

Once the UGEP and UGDG lists are complete, the scores are assigned to the individual BFCs and added together as shown below. The scores to assign to each individual BFC are: UGDG = 7.0 SiFP and UGEP = 4.6 SiFP.

So the size of a whole application is:

SiFP=M(UGEP)+M(UGDG)= #UGEP*4.6 + # UGDG*7.0.

Here #X means the number of X.

According to the conversion between SiFP and UFP defined in the SiFP reference manual, we can draw the following equation to calculate the FPA functional size from the SiFP value:

UFP = #SiFP /0.998

D. Measure Tool for SiFP

There are several UML modeling tools which support OO modeling, such as Visio, Rational Rose, Power Designs, EA and StarUML. These tools not only provide a graphical modeling function, but also export the model as XMI and/or XML file. Measurement tools can be designed by parsing XMI/XML document and using measurement rules. We designed a measure tool SiFPOOTool to automatically measure the SiFP size of an application by its UML model, precisely use case diagram and class diagram. The high-level structure of the tool is shown in Fig.8.

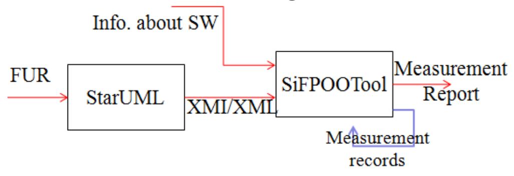
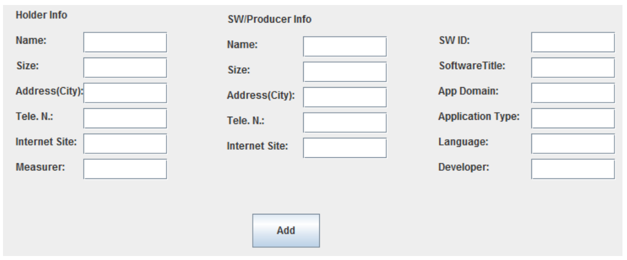


Fig. 8. Theory of SiFPOOTool

The tool provides some functions, such as, reading and parsing XML file derived from UML model, recording and reporting the measure result. Moreover, the related information about the application being measured, the company which holds the application (see Fig. 9), the measurer that carries out the measurement are all recorded by the tool to meet the needs for analysis and inquiries.



Holder Info		SW/Producer Info			
Name:		Name:		SW ID:	
Size:		Size:		SoftwareTitle:	
Address(City):		Address(City):		App Domain:	
Tele. N.:		Tele. N.:		Application Type:	
Internet Site:		Internet Site:		Language:	
Measurer:				Developer:	

Fig. 9. Information input interface of the tool

We measure the GymIS software application using our tool SiFPOOTool: 5 UGDG and 7 UGEP were identified, thus the total size is 67.2 SiFP.

IV. EMPIRICAL VALIDATION

We aimed to validate the two issues: the first one is whether the tool can be used to replace the manual SiFP measurement, when a UML requirement model is available. The second is to validate whether our SiFP-oriented UML modeling rules are correct. The validation overview is shown in the Fig.10.

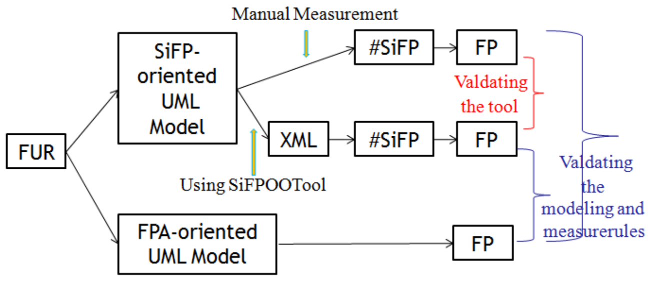


Fig. 10. Validation overview

We used 17 projects' models mainly prepared during previous work [19]. The FURs, UML models (use case diagram, class diagram, components diagram, and sequence diagram) and size measures (in UFP) of those projects are available.

The experimental validation procedure was organized as follows:

−Firstly, for each project, the use case diagrams are reviewed and modified according to the rules 1, 2 and 3 defined in Section III.B.

−Second step: the class diagrams are reviewed and modified according to the rules 4-8 in Section III.B.

−Third step: The activities involved in steps 1 and 2 are repeated until all the projects' use cases and class diagrams comply with the rules 1-8 in Section III.B. Using StarUML, the XMI/XML files are exported from UML model.

− The fourth step: those 17 projects are manually measured using the SiFP method: the results are given in columns 2-4 of Table V. The correspond SiFP and UFP are also calculated automatically and inserted in the 4th and 5th columns of the Table V. The UFP values are computed according to the function SiFP = #UFP*0.998 described in the reference manual [13].

− Then we use our tool SiFPOOTool to measure each model XMI/XML file obtained at step 4. The results and their corresponding UFPs are inserted in columns 6-8 of Table V. To automatically obtain the UFP values, the previous function SiFP=#UFP*0.998 was used in our tool.

− Finally, we copied into Column 10 the functional size measures in UFP manually measured in the previous work.

When all the preparatory work was finished, we performed three paired sample t-Tests on the datasets of manual measurement (Column 5), of the measurement based on SiFPOOTool (Column 9) and of UFP values (Column 10) obtained in the previous work. As usually the level of significance is set as 5%. Test results are as follows: on the datasets of manual measurement (Column 5) and of the measurement based on SiFPOOTool (Column 9), the two-tailed test p-value is approximately 0.104. For the datasets of the manual measurement(Column 5) and the UFP(Column 10), the datasets of measurement based on the tool(Column 9) and UFP(Column 10), both the two-tailed test p-values are approximately 0.001. Then we analyzed the average of absolute value of the ratio of UFP based on the tool(Column 9) and the UFP(Column 10), it is approximately 9.95%, which is less than 10%, so the results obtained based on the tool is acceptable. Our approach (based on UML model) belongs to the third level, detailed measurement level, of the six accuracy levels for software sizing defined in [20][21].

In conclusion, our estimation tool SiFPOOTool can be used to replace manual SiFP measurement in the early phases of the software development cycle, namely domain modeling phase, to measure the functional size of software project. As it turns out, our modeling and measure rules (Rules 1-11 presented in Section III. B, C and D) lead to good experiment results.

TABLE V. DATASETS OF MEASUREMENTS BY HAND, USING SIFPOOTOOL

P.ID	Manual Measurement				Measurement Using SiFPOOTool				UFP	Ratio of 5th/9th column	Ratio of 5th/10th column	Ratio of 9th/10th column
	#UGEP	#UGDG	SiFP	UFP	#UGEP	#UGDG	SiFP	UFP				
1	15	13	160	160.3	15	13	160	160.3	160	0.00%	0.20%	0.20%
2	15	15	174	174.3	15	14	167	167.3	140	4.19%	24.53%	19.52%
3	12	6	97.2	97.4	12	3	76.2	76.4	84	27.56%	15.95%	-9.10%
4	22	10	171.2	171.5	22	11	178	178.6	163	-3.93%	5.24%	9.54%
5	20	6	134	134.3	20	6	134	134.3	128	0.00%	4.90%	4.90%
6	18	8	138.8	139.1	18	9	146	146.1	130	-4.80%	6.98%	12.38%
7	16	3	94.6	94.8	16	3	94.6	94.8	78	0.00%	21.53%	21.53%
8	15	8	125	125.3	15	6	111	111.2	107	12.61%	17.06%	3.95%
9	17	7	127.2	127.5	17	5	113	113.4	102	12.37%	24.96%	11.20%
10	7	8	88.2	88.4	7	8	88.2	88.4	79	0.00%	11.87%	11.87%
11	18	7	131.8	132.1	18	5	118	118.0	105	11.88%	25.78%	12.42%
12	28	4	156.8	157.1	28	4	157	157.1	138	0.00%	13.85%	13.85%
13	22	5	136.2	136.5	22	5	136	136.5	124	0.00%	10.06%	10.06%
14	13	2	73.8	73.9	13	2	73.8	73.9	73	0.00%	1.30%	1.30%
15	20	3	113	113.2	20	3	113	113.2	106	0.00%	6.82%	6.82%
16	27	6	166.2	166.5	27	6	166	166.5	159	0.00%	4.74%	4.74%
17	14	5	99.4	99.6	14	5	99.4	99.6	86	0.00%	15.81%	15.81%

V. RELATED WORK

The generic concepts of FPA were published in the late 1970s. Later, more detailed measurement rules were developed to improve consistency of measurement. Due to lack of good software documentation, it is not always possible to apply all the detailed rules, and measurers must fall back on approximation techniques [22].

In [22] M. Lelli and R. Meli announced this as a paradox: Size estimation is necessary when we do not have enough information (thus, early estimation methods must be used to obtain it). When we can measure with the greatest accuracy, we no longer need that information any more.

In order to figure out whether FPA in the early phases is a realistic option, the committee "FPA" in the early phases” was established in September 1989. The committee investigated whether FPA can be used to perform an indicative size estimate before a complete logical (detailed) design is available [23].

Many techniques for early size estimation have been proposed for FP, such as component sizing technique by Putnam and Myers [24] and the Early and Quick Function Point size estimation techniques by Conte et al. [25]. These methods – such as Estimated NESMA method [8], ISBSG average weights, simplified FP [13], prognosis of CNV AG [11] and so on - do not require the weighting of functions; instead each function is weighted with average values.

Some methods extrapolated the FP counts from the countable components (usually the ILFs) using statistical methods (mostly regression analysis). Some simplified methods – Mark II, NESMA's Indicative FP, Tichenor ILF Model, Prognosis by CNV AG, and ISBSG Benchmark – were constructed according to such technique.

In [15] Lavazza et al. proposed a FPA-oriented UML modeling technique that can make FPA performed in a seamless way, while yielding reliable results. In [26] del Bianco et al. introduced the model-based technique into COSMIC method and suggested a simplified model-based cost estimation models. By using the data from a large popular public dataset Lavazza and Meli confirmed that SiFP can be effectively used in place of IFPUG [14]. However, there has been no measure tool for SiFP so far.

VI. CONCLUSIONS AND FUTURE WORK

Performing Function Point measurement according to the traditional process is expensive and time consuming. The SiFP was proposed as a replacement of FPA. Functional size is mainly used for estimating development costs and project planning. Many software developers use UML, hence they are interested in basing functional size measurement on UML models. In principle, UML-based estimation can be used effectively at the earliest stage of software: our proposal makes this possibility practical and viable. Additional researches (concerning both measurement technology and measurement tools) are necessary to support functional size measurement in different stages of software development.

ACKNOWLEDGMENT

The authors thank Prof. Luigi Antonio Lavazza from the University of Insubria in Varese, Italy, for his constructive suggestions and comments on this research. The authors also thank Jun Wu for his contribution to the implementation of the first version of the tool. The research presented in this paper has been supported by the Start Project Foundation of Hangzhou Dianzi University under Grant No. KYS105614069, by the Defense Industrial Technology Development Program under Grant No. JCKY2013415C001 and Grant No. JSZL2014415B002, and by Weapon Equipment Pre-Research Foundation under Grant No. 9140A15040214DZ04221.

REFERENCES

[1] A. J. Albrecht, "Measuring Application Development Productivity", Joint SHARE/ GUIDE/IBM Application Development Symposium, pp. 83-92, 1979.

[2] International Function Point Users Group, "Function Point Counting Practices Manual - Release 4.3.1", January 2010.

[3] ISO/IEC 20926: 2003, "Software engineering – IFPUG 4.1 Unadjusted Functional Size Measurement Method – Counting Practices Manual", Geneva: ISO, 2003.

[4] A. J. Albrecht and J.E. Gaffney, "Software function, Source Lines of Code and Development Effort Prediction: a Software Science Validation", IEEE Transactions on Software Engineering, vol. 9(6), pp.639-648,1983.

[5] C. Jones, "A New Business Model for Function Point Metrics", http://www.itmpi.org/assets/base/images/itmpi/privaterooms/capersjones/FunctPtBusModel2008.pdf, retrieved: June, 2016.

[6] Total Metrics, "Methods for Software Sizing – How to Decide which Method to Use", http://www.totalmetrics.com/function-point-resources/downloads/R185_Why-use-Function-Points.pdf, retrieved: June, 2016.

[7] "Early & Quick Function Points for IFPUG Methods v.3.1 Reference Manual 1.1", April 2012.

[8] ISO/IEC 24570: 2004, "Software Engineering-NESMA Functional Size Measurement Method version 2.1 - Definitions and Counting Guidelines for the Application of Function Point Analysis", International Organization for Standardization, Geneva, 2004.

[9] J. Geraci and C. Tichenor, "The IRS Development and Application of the Internal Logical File Model to Estimate Function Point Counts,"1994. Presented at the Fall 2000 IFPUG Conference.

[10] L. Bernstein and C. M. Yuhas, "Trustworthy Systems Through Quantitative Software Engineering", John Wiley & Sons, 2005.

[11] M. Bundschuh, "Function Point Prognosis Revisited", FESMA 99, Amsterdam, The Netherlands, October 4-8, 1999, pp. 287–297. http://www.academia.edu/1024603/FUNCTION_POINT_PROGNOSIS_REVISITED, retrieved:June, 2016.

[12] R. A. Monge, F. S. Marco, F. T. Cervigón,V. G. García, and G. U. Paino, "A Preliminary Study for the Development of an Early Method for the Measurement in Function Points of a Software Product", Eprint Arxiv Cs, 2004.

[13] SiFPA, "Simple Function Point Functional Size Measurement Method - Reference Manual, V. SiFP-01.00-RM-EN-01.01", http://www.sifpa.org/en/index.htm, retrieved: June, 2016.

[14] L. Lavazza and R. Meli, "An Evaluation of Simple Function Point as a Replacement of IFPUG Function Point", in 9th Int. Conf. on Software Process and Product Measurement (Mensura) IWSM-MENSURA 2014, October 6-8, 2014, Rotterdam.

[15] L. Lavazza, V. del Bianco, and C. Garavaglia, "Model-based Functional Size Measurement", 2nd International Symposium on Empirical Software Engineering and Measurement (ESEM 2008), Oct. 9-10, 2008, Kaiserslautern, Germany.

[16] OMG–Object Management Group, "Unified Modeling Language: Superstructure", version 2.1.1, OMG formal/2007-02-05, February 2007. (available from http://www.omg.org)

[17] D. Rosenberg and M. Stephens, "Use Case Driven Object Modeling with UML Theory and Practice", Apress, Berkeley, USA, 2007.

[18] G. Antoniol, C. Lokan, G. Caldiera, and R. Fiutem, "A Function Point-Like Measure for Object-Oriented Software", Empirical Software Engineering , Volume 4, Issue 3, pp 263–287, Sept. 1999.

[19] G. Liu, "Towards Making Function Size Measurement Easily Usable in Practice", PhD thesis, University of Insubria, Varese, Italy, 2014.

[20] P. Hill, "Software early lifecycle- Function sizing", SoftwareTech, June 2006, Vol. 9, No.2.

[21] Total Metrics, "Levels of Function Points, Version 1.3", January 2004, http://www.totalmetrics.com/total-metrics-articles/levels-of-function-point-counting, Total Metrics, 2004.

[22] M. Lelli and R. Meli, "from Narrative User Requirements to Function Point", IN: Proceedings of Software Measurement European Forum-SMEF 2005, Mar. 16-18, 2005, Rome, Italy.

[23] NESMA, "The Application of Function Point Analysis in the Early Phases of the Application Life Cycle - A Practical Manual: Theory And Case Study, V. 2.0", http://www.nesma.nl/download/boeken_NESMA/N20_FPA_in_Early_Phases_(v2.0).pdf, retrieved:June, 2016.

[24] L. H. Putnam and W. Myers, "Measures for Excellence: Reliable Software on Time within Budget", Prentice Hall, UpperSaddle River, 1992.

[25] M. Conte, T. Iorio, R. Meli, and L. Santillo, "E&Q: An Early & Quick Approach to Function Size Measurement Methods", In Proceedings of Software Measurement European Forum-SMEF 2004, January 28-30, 2004, Rome, Italy.

[26] V. del Bianco, L. Lavazza, and S. Morasca, "A Proposal for Simplified Model-Based Cost Estimation Models", In Proceedings of 13th Int. Conf. on Product-Focused Software Development and Process Improvement, pp. 59-73, June 13-15, 2012, Madrid, Spain.

[27] F. Ferrucci, C. Gravino, and L. Lavazza, "Assessing Simple Function Points for Effort Estimation: an Empirical Study", 31st ACM Symposium on Applied Computing, April 4-8, 2016, Pisa, Italy.