An Extended Function Point Approach for Size Estimation of Object-Oriented Software

A. Chamundeswari
Department of CSE
SSN College of Engineering,
SSN Nagar – 603 110, India
chamundeswaria@ssn.edu.in

Chitra Babu
Department of CSE
SSN College of Engineering
SSN Nagar – 603 110, India
chitra@ssn.edu.in

ABSTRACT
Early and accurate estimation of software size plays a crucial role in facilitating effort and cost estimation of software systems. One of the widely used methodologies for software size estimation is Function Point Analysis (FPA). Several approaches which adapt this methodology to Object Oriented (OO) Software have been proposed in the literature. However, these approaches lack clarity in providing precise directives for the identification of FPA components. Further, when a particular class is involved in multiple interactions such as aggregation, association and inheritance, its complexity calculation is ambiguous. In order to address these issues, this paper proposes a new and enhanced approach for OO software size estimation by providing rules that better guide the practitioners. This paper discusses a sample case study describing the applicability of the proposed approach. The developmental size predicted by applying the proposed approach for a set of sample projects correlates well with the size prediction obtained through the existing approaches. Thus, the proposed approach provides simple and unambiguous guidelines for the identification of FPA components as well as for the calculation of complexity due to each one of those components, without adversely affecting the accuracy of software size estimation.

Categories and Subject Descriptors
Software – Software Engineering – Design Tools and Techniques - Object Oriented design

General Terms
Measurement, Design

Keywords
Function points, OO software, FPA, size estimation**,** design.

1. INTRODUCTION
Software size estimation for Object Oriented (OO) software is a challenging activity at the early stage in software developmental life cycle. In general, the overall software size is the sum of the size of code developed and the size of the test cases/test drivers. The former is referred to as the developmental size of the software and the latter is known as the size of the testing component. One of the widely used developmental size estimation techniques is FPA proposed by Albrecht [2]. It estimates the size of the software based on the functionality specified in requirements specification, independent of the technology used to build the software. It contains five components namely External Interface File (EIF), Internal Logical File (ILF), External Input (EI), External Output (EO), and External Inquiry (EQ). All these five components are estimated based on the functionality. Fourteen Technical Complexity Factors (TCF) are evaluated based on the non-functional requirements. FPA has been approved by International Function Point User Group (IFPUG) and it has become a standard. It is widely accepted in software industry as a superior metric compared to the naïve lines of code counting for developmental size estimation.

With the advent of OO software, FPA has been adapted for developmental size estimation by many researchers. These adapted approaches estimate the size of the software by mapping the various key notions of object model to the FPA components. However, these adapted approaches, have specified widely varying directives to estimate the size of the OO software, and lack in setting up precise guidelines for practitioners supporting the estimation process. Further, when a class has multiple relationships with other classes, the complexity of that class increases. In order to address these issues, this paper proposes a new and enhanced approach for developmental size estimation based on object model.

The objective of this research work is to provide a new enhanced approach for developmental size estimation using object model, adapting the FPA technique. This has been achieved as follows:

- Mapping the object model components to FPA components during the analysis phase.
- Setting the rules for FPA mapping to determine the various parameter values based on the dependency relationships a given class is involved.
- Estimating the developmental size of OO software, in terms of function points, by applying this FPA mapping.

The remainder of the paper is organized as follows. Section 2 outlines the FPA based software developmental size estimation techniques available in the literature. In Section 3, the new developmental size estimation model which applies FPA mapping on object model is presented. Section 4 discusses the results. Section 5 concludes and suggests future directions.

2. RELATED WORK

The research work on quantifiable developmental size estimation has been the focus of many researchers [2,5,9,10,15,16,17,18,19,23]. It has been dealt during the various phases of the software development life cycle such as analysis [4,9,10,11,23], design [4,15,17,23] and coding [14,19,16]. As FPA technique cannot be directly used for estimating the size of OO software [4,10,11,15,23], mapping rules were framed, to adapt FPA.

Harput et al. [11] have applied FPA to OO requirements specifications. Rules were defined to specify a semi-automatic transformation from OO requirements model to FPA model. Ashman [6] applied use case based estimation model for determining the project effort. This estimation model was applied in an iterative development process to improve the accuracy of successive estimates based on repeatable measurements. Fernandez et al. [9] have applied COSMIC Full FP to estimate the size of the OO Software at early stages of development. COSMIC Full FP is a size estimation method sharing the commonalities of IFPUG and MARK II size estimation methods.

Minkiewicz [14] proposed a size estimation technique in terms of predictive object points using the design structure. Fetcke et al. [10] proposed an approach, to estimate the software by applying Jacobson methodology and by framing appropriate rules. Antoniol et al. [3,4,5] proposed Object Oriented Function Points (OOFP) from the user's perspective at analysis phase using the object model. Four selection approaches were introduced to estimate the size, such as single class, aggregation, generalization, and mixed (aggregation and generalization). Identification of components and FPA mapping rules as defined in IFPUG were applied. Lot of flexibility was given to the user to choose any one of the selection approach and apply. Validation procedure was applied to measure the performance. Several regression techniques were applied to derive a relationship between the lines of code and OOFP. Based on this relationship, a predictive accuracy of 0.337 was achieved using generalization selection method.

Ram et al. [15] proposed Object Oriented Design Function Point(OODFP) technique, from the designer's perspective at design phase using the object model for developmental size estimation. In this work, the single class selection method [5] was updated with class complexity classification. The class complexities such as low, average and high was defined with their weightage, based on data visibility. This complexity table along with FPA was used to estimate the developmental size. The complexity due to inheritance, and polymorphism were also taken into account. Zivkovic et al. [23] proposed an Iterative Estimation Technique(IET) to improve the accuracy of the estimation with more data, at three-abstraction levels in the designer's perspective. Estimation is done in three stages such as basic, comparative and final. In final stage, OO to FPA mapping[5] was followed, with modified complexity table [1] in estimating Transaction Function(TF) complexity. Effort in terms of size was empirically evaluated and an error of 9.8% was achieved.

Thus, each method has its own unique way in determining the developmental size. The formation of ILF remains the same in OOFP, OODFP, and IET. However, ILF complexity calculation in OODFP slightly differs from the OOFP method for inheritance data. The formation of TF in OOFP is not clearly stated. In OODFP technique, it is clearly stated for each return type, arguments, function type like virtual, override, etc. IET follows the same approach as OOFP, but applies the modified complexity table [1] to determine TF complexity. In all these approaches, when a given class has multiple interactions with other classes, the increased ILF complexity of the given class is not directly accounted for. Instead, this complexity is indirectly evaluated by the grouping of classes based on aggregation, association and inheritance. However, there is a lot of ambiguity in deciding the grouping of classes and this subsequently results in variations in size estimation depending upon which way the classes were grouped. Thus, in order to avoid this ambiguity, this paper proposes a new and enhanced approach for developmental size estimation of OO software.

3. PROPOSED SIZE ESTIMATION

The main objective of FPA is to determine the size based on functional requirements of the software application. The various components in the object model are mapped to FPA components so that the FPA can be applied for size estimation. Software comprises of many applications and it is essential to identify the boundaries of different applications. The boundaries are classified into two categories: internal and external. An internal boundary in the object model is the class diagram which contains different classes to perform different functionalities within the given application. An external boundary in the object model is a set of classes which is referenced by the application outside the internal boundary. Thus, internal boundary is mapped to ILF component of FPA, while external boundary is mapped to EIF component of FPA. Methods in the classes are the main transactions and they are mapped to EI components of FPA. Figure 1 gives a pictorial overview of the proposed developmental size estimation model. The complexity due to various ILF, TF and EIF components are summed up to estimate the developmental size of the software, OMFP(Object Model Function Point) based on object model.

Developmental size estimation is determined based on the following steps: The first step is the identification of files and transactions within the internal boundary. The second step is to assign weights for the files and transactions. The third step is to estimate the developmental size of the software. The following subsection discusses these steps in detail.

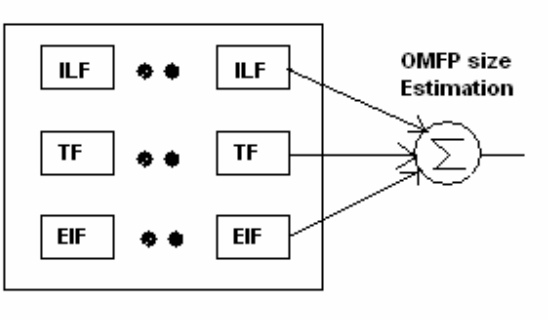

Figure 1. Size estimation model

3.1 Mapping Object Model to FPA

The classes are the main candidates in the class diagram based on which the entire size estimation is carried out. The class diagram has various parameters such as classes, identifiers, methods, and relationships, which are valuable resources for size estimation. A class can interact with another through one or more relationships such as association, aggregation, and inheritance.

3.1.1 Identification and Classification of ILF

Each class in a class diagram performs a different functionality addressed by the software. A given class can be invoked by other classes or a given class can invoke the other classes. Thus, a class plays a vital role through different relationships it has with other classes. Hence, the complexity of each class needs to be precisely estimated to predict the size of the entire software to be developed. All the classes within the internal boundary are identified and mapped to the ILF component of FPA.

Classification of ILF depends on two parameters, Record Element Type (RET) and Data Element Type (DET) as referred in IFPUG [12]. RET represents a user recognizable group of logically related data. DET represents a simple unique user recognizable, non-recursive data in RET. The rules for proper classification of RET and DET in a given class have been formulated as follows:

[1] Simple data types contained in this class such as integer, string, float, etc. are assigned one DET each.
[2] Complex data types contained in this class such as event, object, etc. are assigned one RET each.
[3] If this class is involved in a single association or simple aggregation relationship with another class, one DET is assigned.
[4] If the given class is in any one of the following relationships with another class, such as multiple association, composite aggregation, simple aggregation or multiple association with other class, one RET is assigned for each.
[5] If this class is a base class, one RET is assigned.
[6] If this class is a derived class, one RET is assigned.
[7] If this class is a derived class, one DET is assigned for each derived simple data type from its base class.
[8] If this class is a derived class, then one RET is assigned for each derived complex data type from its base class.
[9] The given class itself is accounted for one RET.

Once, RET and DET parameters are identified and classified for each individual class in a class diagram, complexity of the ILF can be estimated using the ILF complexity table as defined in IFPUG standard[12].

3.1.2 Identification and Classification of TF

Each class has one or more methods and all the transactions happen through them. Methods in a given class can be invoked by other class methods, or these methods invoke methods belonging to other classes. Only concrete methods are considered and abstract methods are not considered. Polymorphism is achieved through method overloading or inheritance. Hence, virtual, override and overload methods are considered as separate methods. Thus, these methods that activate transactions are termed as Transaction Functions (TF). They play a vital tole and hence they need to be precisely estimated to predict the developmental size of the software. These TF are identified and mapped to the EI component of FPA.

Classification of TF depends on two parameters namely File Type Referenced (FTR) and DET. FTR represents a transaction in EI. DET represents a unique user recognizable and non-recursive data in FTR. The rules for proper classification of FTR and DET in a given TF have been formulated as follows:

[10] If the return type is void, in TF, one DET is assigned.
[11] If the return type from the TF is a simple data type such as integer, string, float, etc., one DET is assigned.
[12] If the return type from the TF is a complex data type such as objects, events, etc. one FTR is assigned.
[13] If the argument is void in TF, one DET is assigned.
[14] If the argument is a simple data type in TF, one DET is assigned for each.
[15] If the argument is a complex data type in TF, one FTR is assigned for each.
[16] If this TF is involved in multiple association with another TF, one FTR is assigned.
[17] If this TF is involved in single association with another TF, one DET is assigned.
[18] The given TF in class diagram is accounted for one FTR.

Once, FTR and DET parameters are identified and classified for each individual TF in the class diagram, complexity due to that TF can be estimated using EI complexity table as defined in IFPUG standard [12].

3.2 Size Estimation

The size estimation of a software application can be determined by applying all the above eighteen proposed rules. The proposed estimation, OMFP, is calculated as shown below:

*OMFP = Unadj OMFP * TCF*
where

Unadj OMFP = EIF + ILF + TF
EIF = f (RET, DET)
ILF = f (RET, DET)
TF = f (FTR, DET)
*TCF = 0.65 + 0.01 * Σ t_i*

OMFP is determined from four components namely *EIF, ILF, TF* and Technical Complexity Factor (TCF). *TCF* is determined from fourteen characteristics (t_i,) with 'i' varying from 1 to 14, as defined by Albrecht [2]. The following section describes how the proposed size estimation technique can be applied for a sample case study.

4. RESULTS AND DISCUSSIONS

Consider a sample case study for which the class diagram is shown in Figure 2, to validate the proposed estimation model. Eighteen mapping rules as specified in Sections 3.1.1, 3.1.2, and the formula in Section 3.2 are applied on this case study. There are totally 6 classes, 10 methods, and 9 data types. There are three association relationships, one base class, two derived classes, one simple aggregation and one complex aggregation. ILF's parameter values are identified and classified by applying rules 1 to 9 to determine the values of RET and DET. TF's parameter values are identified and classified by applying rules 10 to 18. ILF and TF size for the sample case study is tabulated in Table 1 and Table 2. TCF for the 14 characteristics is calculated as 1.07.

Unadj OMFP = ILF + TF
= 45+32 = 77
OMFP = Unadj OMFP * TCF
= 77 * 1.07 = 82.39 FP

The proposed size estimation technique, OMFP is also applied on four different projects developed in software engineering laboratory. These projects were developed using OO development process. Developmental size estimation techniques such as OOFP [5] and UCM_{fp} [7] which adapt FPA during analysis phase were applied on those projects. The comparison of projects using UCM_{fp}, OOFP, and OMFP is tabulated in Table 3. The correlation coefficient obtained through the data sets in Table 3 signifies that

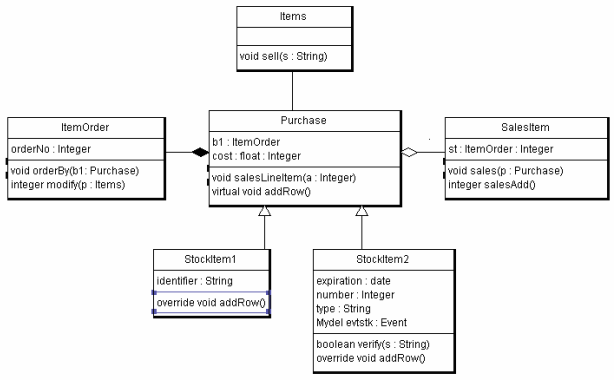

Figure 2. Sample case study.

Table 1. ILF complexity estimation

ILF	RET	DET	Complexity
Item	1	1	7
ItemOrder	2	1	7
Purchase	8	1	10
SalesItem	2	1	7
StockItem1	3	2	7
StockItem2	4	4	7
Total ILF	20	10	45

Table 2. TF complexity estimation

TF	FTR	DET	Complexity
void orderBy()	2	2	3
integer modify()	2	2	3
void salesLineItem()	4	2	4
virtual void addRow()	4	2	4
void sales(p: Purchase)	2	2	3
integer salesAdd()	1	3	3
override void addRow()	1	2	3
boolean verify()	1	2	3
override void addRow()	1	2	3
void sell	1	2	3
Total TF	19	20	32

the prediction of size estimation at the analysis stage achieved through the proposed approach compares well with the earlier approach OOFP that is based on object model and UCM_{fp} that is based on use case model. The rules that are proposed in OMFP simplify the identification of FPA components, while retaining the estimation accuracy of its counterpart approaches. Specifically, rule 4 ensures that a class that is involved in more number of interactions will have more complexity compared to a class which is involved in lesser number of interactions. This eliminates the need for ambiguous grouping of classes and the consequent disparity that arises due to the possibility of variations in grouping choices selected by various estimators.

5. CONCLUSIONS AND FUTURE WORK

A new and enhanced approach for estimating the developmental size of OO software is proposed by applying FPA on Object model. This approach involves mapping of object model components to FPA components. Classes and their various relationships are considered to estimate the size. Eighteen rules are proposed to guide the practitioners for identification and classification of FPA components for size estimation. This technique is applied on four sample projects and the corresponding OMFP is estimated. These results are compared with the function points calculated for the same set of projects by applying the existing developmental size estimation approach. The correlation coefficient achieved signifies that the size estimation using the proposed approach compares well with the results obtained through the existing approaches. Thus, the proposed approach removes the ambiguity in identification of

Table 3. Comparison of size estimation techniques

	No. of Requirements	No. of Use cases	No. of Classes	UCM_{fp}	OOFP	OMFP
Passport automation system	22	10	7	89.88	83.46	86.67
e-book management system	21	8	7	83.46	87.74	88.81
Online photo sharing and indexing	15	10	6	92.02	93.09	97.37
Foreign exchange system	21	10	7	108.07	115.56	118.77
Correlation With OOFP With UCM_{fp}						 1 0.95

FPA components and in the calculation of their complexity, while maintaining the accuracy in size estimation. Future work is to empirically validate this size estimation model by applying it on large scale projects from software industry.

6. REFERENCES

[1] Al-Hajri M A, Ghani A A A, Sulaiman M N, Selamat M.H, Modification of standard function point complexity weights system, Journal of Systems and Software, 2005, pp. 195–206.

[2] Albrecht A, "Measuring application development productivity", IBM Application Development Symposium, 1979, pp. 83-92.

[3] Antoniol G, Calzolari F, Cristoforetti L, Fiutem R, Caldiera G, "Adapting function points to object oriented information systems", Lecture notes in computer science, Advanced Information System Engineering, Vol 1413, 1998, pp. 59-76.

[4] Antoniol G, Fietum R, Lokan C, "Object oriented function points: An empirical validation", Journal Empirical Software Engineering, Vol 8, No 3, Sep 2003, pp. 225-254.

[5] Antoniol G, Lokan C, Caldiera G, Fiutem R, "A function point-like measure for object oriented software", Empirical Software Engineering, Vol 4, Sep 1999, pp. 263-287.

[6] Ashman R, "Project estimation: A simple use-case based model", IT professional, Vol 6, No 4, Jul 2004, pp. 40-44.

[7] Chamundeswari A and Chitra B., "Function point size estimation for object oriented software based on use case model", Third International Conference on Software and Data Technologies, July 2008, pp. 114–120.

[8] COSMIC-FFP Measurement Manual, Common Software Measurement International Consortium, Vol 2.2, 2003.

[9] Fernandez N C, Abrahao S, Pastor O, "Towards a functional size measure for object oriented systems from requirements specifications", Proceedings of the Fourth International Conference on Quality Software, Vol 00, 2004, pp. 94-101.

[10] Fetcke T, Abran A, Nguyen T H, "Mapping the OO-Jacobson approach into function point analysis", Proceedings of IFPUG, 1997 Spring Conference, 1997, pp. 134-142.

[11] Harput V, Kaindl H, Kramer S, "Extending function point analysis of object oriented requirements specifications", Eleventh IEEE International Software Metrics Symposium, Vol 00, 2005, pp. 39-49.

[12] IFPUG, Function Point Counting Practices Manual, Function Point Users Group, Westerville, Ohio, 1999.

[13] MK II Function Point Analysis, Counting Practices Manual, Version 1.31, United Kingdom Software Metrics Association, 1998.

[14] Minkiewicz A F, "Measuring object oriented software with predictive object points", Proceedings of the Conference on Applications in Software Measurements, Oct 1997.

[15] Ram D J, Raju S V G K, "Object oriented design function points", Proceedings of the First Asia Pacific Conference on Quality Software, Hong Kong, 2000, pp. 121-126.

[16] Schooneveldt M, "Measuring the size of object oriented systems", Proceedings of the Second Australian Conference on Software Metrics, Metrics Association, Nov 1995, pp. 168-177.

[17] Uemura T, Kusumoto S, Inoue K, "Function point analysis using design specification based on the Unified Modeling Language", Journal of Software Maintenance Evolution-Research Practice, Vol 13, 2001, pp. 223-243.

[18] Uemura T, Kusumoto S, Inoue K, "Function Point measurement tool for UML design specification", Proceedings of the Sixth International Symposium on Software Metrics, Vol 62, 1999, pp. 62-69.

[19] Whitmire A S, "Applying function points to object oriented software models", Software Engineering Productivity Handbook, Mc Graw-Hill, New York, 1992, pp. 229-244.

[20] Zivkovic A, Hericko M, “Tips for estimating software size with FPA method”, Proceedings of the IASTED International Conference on Software Engineering, Acta Press, 2004, pp. 515-519.

[21] Zivkovic A, Hericko M, Brumen B, Beloglavec S, Rozman I, “The impact of details in the class diagram on software size estimation”, Informatica, Vol 16, No 2, 2005, pp. 1-18.

[22] Zivkovic A, Hericko M, Kralj T, “Empirical assessment of methods for software size estimation”, Informatica, Vol 4, 2003, pp. 425-432.

[23] Zivkovic A, Rozman I, Hericko M, “Automated software size estimation based on function points using UML models”, Journal of Information and Software Technology, Vol 47, 2005, pp. 881-890.