Can Function Points be Mapped to Object Points?

Ayman Issa[1], Mohammed Odeh[1], and David Coward[2]
[1]Centre for Complex Cooperative Systems, CEMS Faculty, University of the West of England, UK
[2]School of Computer Science, CEMS Faculty, University of the West of England, UK

Abstract: *Object points is a new software size metric that has emerged to cope with recent developments in software engineering, and to overcome the deficiencies of the traditional lines of code and function points size metrics. Moreover, object points has been utilized as the basis for several software cost estimation models with promising improvements in the accuracy of estimates. However, the infancy of the object points size metric means that there is a shortage of object points based software historical projects, on which to base the empirical validation of the new object points based software cost estimation models. Hence, the relationship between the extensively used function points and newly invented object points size metrics have been conceptualized and utilized in a novel forward approach to convert the function points projects data into their equivalent object points data. Empirical investigations of 66 function points projects have shown high correlation and significance, 88% and 0.33, respectively, between the resulting object points effort estimates and the actual function points effort. Furthermore, the resulting object points data have been utilized to model the embodied function points-object points relationship in two specialized productivity factors and function points type dependent linear models. The resulting models have shown high fitness, R^2, values of 0.95, for both models.*

Keywords: *Software size metrics, function points, object points, software cost estimation.*

Received August 1, 2005; accepted January 292006

1. Introduction

Software system sizing has been a difficult and controversial topic for a long time. Great debates have been reported in the literature [14] on whether lines of code is a better sizing metric than Function Points (FP), and vice versa. This is dependent on several factors such as system type, programming language(s), and the software development life cycle phase in which the estimation is conducted. Several software systems size metrics have emerged to cope with recent developments in software engineering including new development paradigms, new programming language generations, and new Computer Aided Software Engineering (CASE) development tools. Object Points (OP) [1], being a size metric developed to cope with the visual widgets (objects) of recent programming language generations, is a typical example of those metrics that were intended to be used and estimated earlier and faster than lines of code and FP.

Several software cost estimation models [1, 2] have been built on top of this relatively young sizing metric and, most interestingly, their results were promising and comparable to those models that are based on the more complicated low level lines of code and FP size metrics. However, all these new models share the same problem. There is a shortage of historical project data, which significantly reduces the reliability of their empirical validation. The main reason behind this data shortage is that these new metrics are based on the most recent software development life cycles artifacts, where only a limited number of organizations have adopted and applied them in their software development projects [14]. However, the International Software Benchmarking Standards Group (ISBSG) [6] has taken the responsibility of building a large volume software project repository based on the FP size metric. On the other hand, building similar repositories based on the new sizing metrics, e. g., OP, will take a long time. This requires them to be recognized and applied by software development houses yet, until the data is available to justify a change of metric they may be loath to do so.

Thus, the promising results of the new sizing metric, e. g., OP, based software cost estimation models, and the unavailability of historical data raised a number of research questions:

- How can these models be validated and employed in the software cost estimation phases of the current software development life cycles?.
- Is it possible to generate highly reliable software project data for the new sizing metrics, e. g., OP, from the currently available data?.
- To what extent will the generated data be sufficiently reliable to be employed in building and validating software cost estimation models that are

applicable at the early stages of the software development life cycle?.

Consequently, this research is aimed at investigating these research questions that exist between the new OP and the traditional FP sizing metrics. These sizing metrics, OP and FP, have been selected for this investigation for three major reasons. First, OP has been used in earlier research [8] as a basis for a use case-OP based software cost estimation model and more historical projects data are needed to inform its reliability. Second, the ISBSG projects data repository provides a large volume of reliable FP based software projects data. Third, there appears a hidden relationship between the elements of OP and FP sizing metrics that should be bridged and modelled for further research and validation.

Section 2 surveys the most well-known software size metrics conversion literature. The process of bridging the relationship between FP and OP is explained in section 3. A forward FP-OP conversion approach is proposed in section 4. The implementation of the forward conversion approach is discussed in section 5. Section 6 briefly explains the nature of the ISBSG empirical data, and evaluates the empirical OP results. Finally, the conclusion and the outline of future work are presented in section 7.

2. Size Metric Converters

The most well known software size metrics converter in the literature is the FP-lines of code backfiring table produced by Jones's at the Software Productivity Research Centre [9, 15]. This table, as shown in the extracted sample in Table 1, approximates the number of logical source statements of each programming language/ language generation that correlated roughly with a single FP [15], as observed by analyzing several historical projects.

Table1. An extract of FP-lines of code backfiring table.

Programming Language/ Language Generation	Average Source Statements Per FP
1st Generation default	320
2nd Generation default	107
3rd Generation default	80
4th Generation default	20
5th Generation default	4
C++	55
Java	53

Smith [13] proposed a use case-lines of code backfiring table to approximate the system size in lines of code from its use case model. Smith assumed that each software solution has a structural hierarchy consisting of the following levels: System of systems, system(s), subsystem group, subsystem(s), and class(es). As detailed in Table 2, the method determines and uses typical adjacent factors between the architectural levels to approximate the lines of code system size [13].

Table2. Use case model-lines of code architectural level adjacent factors.

Architectural Level	Adjacent Factor
Operation Size	70 lines of code
Number of operations per class	12
Number of classes per subsystem	8
Number of subsystems per subsystem group	8
Number of subsystem group per system	8
Number of systems per system group	8
Number of external use cases (per system, subsystem, etc.)	10

Unfortunately, there are no available historical projects data that accommodate OP and FP information. Thus, a straightforward statistical analysis could not be performed to generate a similar FP-OP conversion table, which facilitates the process of generating OP projects data, and consequently validating the OP based software cost estimation models. Therefore, the problem should firstly be conceptualized to bridge the relationship between the elements of the two sizing metrics. Then, the available FP data could be used empirically to investigate this conceptualization.

3. FP-OP Relationship Conceptualization

Albrecht, as cited in [12], has developed the well-known FP size metric which sizes the software project based on its functionality. It counts the number of Internal Logical Files (ILF), External Logical Files (ELF), External Input (EI), External Output (EO), and External Queries (EQ). Banker *et al.* [1] have proposed the OP size metric as a replacement for FP to cope with the visual widgets of the fourth generation languages and the integrated CASE environments. OP is mainly concerned with producing a reliable early count of the application points, being the sum of the adjusted number of screens, reports, and third generation language modules that are expected to be developed to supplement the fourth generation languages code. Hence, OPs do not relate to object oriented concepts such as inheritance, encapsulation, etc, but are more closely dependent on the user interface of the system being developed. Furthermore, the OP method was adopted and generalized by the COCOMO II team [2] as a size measure in the early prototyping stage, namely the Application Composition model. Section 3.1 outlines the mapping between the elements of the original FP and OP sizing metrics, while section 3.2 explains how the different OPs with different complexity levels consume the different FPs elements based on the original FP and OP methods.

3.1. FPs-OPs Elements Mapping

In defining OP measures, Boehm *et al.* [2] identified three categories of complexity levels for screens and reports: Simple, medium, and difficult. Furthermore, OP complexity levels are determined using specialized characteristic dimensions. As detailed in Table 3, screens' complexity levels are rated according to the embodied number of views and data tables [1, 2]. Each view contains several EI, EO, or EQ items that send or receive data to or from several ILFs or ELFs. Reports are rated similarly by the embodied number of sections and data tables [1, 2]. Furthermore, as can be seen in Table 3, screen and report complexity levels are dependent on the values of their characteristic dimensions [2]. For example, a screen that contains less than 3 views and interacts with less than 4 data tables is considered as a simple screen. At the same time, a screen that contains 3-7 views and interacts with less than 4 data tables is also considered as a simple screen. Hence, in order to facilitate the subsequent phases in bridging the relationship between FP and OP sizing metrics, new derived screen and report complexity level schemes have been defined in this conceptualization phase. Table 4 presents the derived screen complexity levels rating scheme.

Table 3. Original OP method screens complexity rating scheme.

Number of Views Contained	Number and Source of Data Tables		
	Total < 4 (< 2 srvr < 3 clnt)	Total < 8 (2/3 srvr 3-5 clnt)	Total 8+ (> 3 srvr > 5 clnt)
< 3	Simple	Simple	Medium
3 – 7	Simple	Medium	Difficult
> 8	Medium	Difficult	Difficult

Table 4. Derived screens complexity levels.

Number of Views Contained	Number and Source of Data Tables		
	Total < 4 (< 2 srvr < 3 clnt)	Total < 8 (2/3 srvr 3-5 clnt)	Total 8+ (> 3 srvr > 5 clnt)
< 3	Simple 1	Simple 2	Medium 1
3 – 7	Simple 3	Medium 2	Difficult 1
> 8	Medium 3	Difficult 2	Difficult 3

When defining complexity levels for third generation language modules, Boehm *et al.* [2] have identified one complexity level for OPs. They stated that all third generation language modules OPs are rated equally as difficult as modules that are built to support fourth generation languages code. Possibilities of third generation language modules include time/event-triggered module, application programming interface/protocol based system call, and data manipulation supportive module [10, 16]. Hence, third generation language modules use EI, EO, and EQ in performing their tasks, which will incorporate some interaction with ILF or ELF to inquire about the system state or save their results. Accordingly, Figure 1 presents a suggested mapping between the elements that compose the FP and OP size metrics.

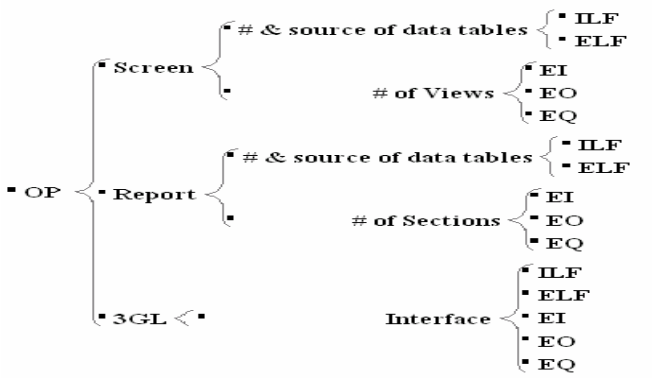

Figure1. Suggested FP-OP elements mapping.

3.2. Qualitative & Quantitative Analysis of FP-OP Relationship

According to the original OP method complexity levels described in the previous section, there are 9 distinct complexity levels of screens, 9 distinct complexity levels of reports, and 1 complexity level for third generation language modules. This focuses attention on discovering more about the nature of the FP elements that are embodied in each OP complexity level. Subsequently, several distinct FP element combinations have been constructed and matched to the different OP complexity levels taking into consideration the original characteristic dimensions of each OP complexity level, and the following FP-OP mapping and consumption guideline:

Guideline 1: Screen sections and report views should be associated with EI, EO, and EQ FP elements. Also, screen and report data tables characteristic dimensions should be associated with ILF and ELF FP elements.

Table 5 presents a sample from the constructed FP element combinations and their corresponding screen OP complexity levels. Assuming that this correspondence is supported by empirical analysis, this should deal with the qualitative aspects of the relationship between FP and OP elements. Furthermore, these FP-OP element pairings form the basis for the quantitative analysis of the FP-OP element relationship. As a result, the values of the original OP complexity levels characteristic dimensions have been utilized to define the quantitative FP consumption for each OP complexity level. In addition, the following FP-OP elements mapping and consumption guidelines have also been used to bridge the gaps between the semantics of the different FP and OP elements.

Guideline 2: The complexity of managing one ELF equals that of two ILFs due to the overhead of external systems communication.

Hence, the existence of ELFs in any FP combination raises the complexity of the corresponding OP complexity level. This may result in some OP complexity levels where the total FP consumption is

less than the original OP characteristic dimensions values.

Table 5. Screens complexity levels and their matched FPs combinations sample.

Screen Complexity Level	FP Combination
Simple 1	EI, ILF
Simple 3	EI, ELF
Medium 1	EI, EQ, ILF
Medium 2	EO, EI, ILF
Difficult 2	EI, EQ, ELF
Difficult 3	EO, EI, ILF, ELF

Guideline 3: OP characteristic dimension intervals have been distributed among the involved complexity levels uniformly to highlight the complexity differences in the resulting objects of the same interval.

In that, simple objects have been assigned the lower limit of the dimension's interval. Medium objects have been assigned the middle of the dimension's interval, and, difficult objects have been awarded the upper limit of the dimension's interval. For example, according to the original OP method complexity rating scheme, any screen that consists of 3-7 views could be considered as simple, medium, or difficult depending on the embodied number of data tables. Hence, in the defined FP-OP consumption scheme, simple screens are considered to have 3 views, medium screens are considered to have 5-6 views, and difficult screens are considered to have 7 views.

Guideline 4: According to the original OP method, any screen may contain less than 3, from 3 to 7, or greater than 8 views.

However, no mention is made of screens containing 8 views. It is not clear how they are classified and rated. Instead of extending the range of the last interval to include screens that have greater or equal 8 views, a decision has been made to extend the range of the middle interval of screens views to include screens that contain 8 views since it covers a diverse range of OP complexity levels: Simple, medium and difficult.

Guideline 5: It has been found, by surveying the literature [10, 17], that regardless of the third generation language module type, a minimum of 1 FP for each FP element will be needed to develop any third generation language module.

As a result of applying the above FP-OP mapping and consumption guidelines to the matched FP-OP combinations, three screens, reports, and third generation language modules FP consumptions schemes have been constructed. Table 6 presents a sample from the constructed screen-FP consumption scheme.

4. Forward FP-OP Conversion Approach

This has been denoted as a forward bottom-up approach compared to a backward top-down approach, being developed by the research team, as simply the latter approach starts from the assumption that the total effort when estimating using FPs is equal to the total effort when estimating using OPs.

In the forward approach, the OP counts and effort of ISBSG FP historical software projects are estimated based on the above newly defined FP-OP elements mapping and consumption schemes. The following sections detail the underlying assumptions, input and output, and workflow of the proposed conversion approach.

Table 6. Screens FP consumption scheme sample.

Screen Complexity Level	EI	EO	EQ	ILF	ELF
Simple 1	1			2	
Simple 3	3				2
Medium 1	2		1	8	
Medium 2	4	2		6	
Difficult 2	5		4		5
Difficult 3	6	5		7	5

4.1. Assumptions

The forward FP-OP conversion approach has been built on top of several assumptions that facilitate the conversion process and place the development context of ISBSG software development projects:

1. FP and OP elements are related to each other according to the defined FP-OP elements relationship.
2. The original OP method productivity scheme does not fit ISBSG FP projects.
3. The generated OP counts should cover all the OP types in their ideal proportions in real world systems.

4.2. Input and Output

For each ISBSG FP project, several input data are required by the forward conversion approach to infer its corresponding OP elements. The required input data could be partitioned into two groups as follows:

1. *FP Project Information*: This includes actual FP effort, EI, EO, EQ, ILF, and ELF FPs counts.
2. *Productivity Project Information*: This includes lower CASE tools with/without code generation used, upper CASE tools used, integrated CASE tools used, and programming language generation.

On the other hand, the forward conversion approach should produce the following OP artefacts for each project:

1. Total Adjusted OP (AOP) count.
2. Total unadjusted OP count.
3. Unadjusted OP count breakdown.
4. OP re-estimated effort.

4.3. Conversion Workflow

The workflow of the forward conversion approach consists of four phases:

1. Determine project parameters.
2. OP counting and FP consumption.
3. Project re estimation.
4. Record conversion results.

Each phase has a well-defined objective and relationship with subsequent phases. In addition, the objective of each phase is carried out by one or more specialized steps. The boundaries of the conversion phases and the embodied steps are presented in the workflow of Figure 2. The corresponding details of the conversion phases and steps are explained in the following subsections. However, a detailed numerical conversion example that explains the details of the proposed forward conversion approach is presented in [7].

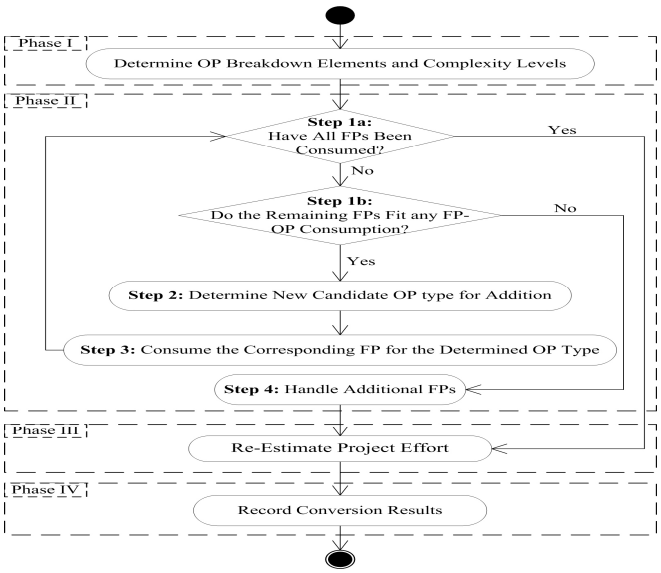

Figure 2.Forward conversion approach workflow.

4.3.1. Phase I: Determine Project Parameters

The main objective of this phase is to place the OP context of the current project to guide the FP-OP conversion process. For realistic and correct generation of OP counts, it is necessary to cover all OP types in their ideal proportions in real world systems. Hence, according to the current project type (i. e., shared or stand alone application [6, 14]), the OP types and complexity levels contribution proportions are determined from the calculated contribution proportions in Tables 7 and 8.

4.3.2. Phase II: OP Counting and FP Consumption

This phase embodies the core functionality of the forward conversion approach. It follows a bottom-up iterative approach to construct the OP counts of the current project. Furthermore, it employs the FP-OP consumption schemes to achieve the main objective of this phase by the means of four specialized steps.

Step 1: *Check FPs Availability*
The first step in this phase is the assessment of the termination condition for the OP counting iterations. It checks whether all FPs have been consumed, and if not, determines whether or not the remaining FPs fit the FP consumption scheme of any OP complexity level. The outcome to these queries determines the next step to be executed. If the remaining FPs fit the FP consumption of one or more OP complexity level, control is transferred to step 2, otherwise step 4 is executed. Alternatively, if all FPs have been consumed, control is transferred to phase III.
Step 2: *Determine New Candidate OP Type for Addition*
In this step, a new candidate OP complexity level for addition is determined on the basis of both the remaining FP topography and the current project OP proportions.
Step 3: *Consume the Corresponding FP for the Determined OP Type*
This step is specialized in using the defined FP-OP consumption schemes to consume the corresponding FPs for the determined OP complexity level.
Step 4: *Handle Additional FPs*
This step is executed when the remaining FPs do not fit the FP consumption of any OP complexity level. Hence, an additional FP handler is triggered to distribute the additional FP across several OP complexity levels that have not utilized the upper limit of their OP characteristic dimensions.

Table 7. OP contribution proportions in the different software systems types.

OP Type	Software System Type	
	Shared Application	Stand-Alone Application
Screen	38%	57%
Report	24%	43%
Third Generation Language Module	38%	0%

Table 8. OP complexity levels sub-proportions in the different software systems types.

OP Type	Complexity Level	Shared Applications' Proportions	Stand-Alone Applications' Proportions
Screen	Simple	79 %	49 %
	Medium	17 %	32 %
	Difficult	4 %	19 %
Report	Simple	78 %	48 %
	Medium	20 %	35 %
	Difficult	2 %	17 %

4.3.3. Phase III: Project Re-Estimation

The main objective of this phase is to re-estimate the project effort using the resulting AOP count as:

$$Effort = AOP/OP\text{-}Productivity$$

where AOP is the resulting count from the previous phase, and the OP-productivity, determined from the calculated programming language generation and CASE tools dependent OP productivity scheme in Table 9.

Table 9. Calculated OP productivity scheme.

Programming Language Generation	CASE Tool Type and Level				No CASE Tools
	Upper CASE	Lower CASE (no code generation)	Lower CASE (with code generation)	Integrated CASE	
Third Generation Language	5	6	7	8	4
Fourth Generation Language	7	8	9	12	6

4.3.4. Phase IV: Record Conversion Results

The main objective of the last phase of the forward conversion approach is to archive the conversion results, as specified in the input and output section, for expert analysis and evaluation.

5. FP-OP Converter System

Java, being a powerful portable language that eases the burden of platform dependent languages [4], has been adopted to implement the proposed conversion approach. Section 5.1 discusses the conversion system architectural style and the functionalities of the different system components are outlined in section 5.2.

5.1. System Architecture

A model-view-controller 3-tier architecture [14] has been adopted in the implementation of the conversion system to separate the concerns of the different system components. The first tier, the view tier, is represented by a graphical user interface component. The middle tier, the controller tier, holds the business logic component of the conversion approach. The last tier, the persistence model tier, is represented by a data management component.

5.2. Components Functionalities

Data management component is the data consumer and producer component. It consumes the data by reading the required input data project by project from the source file. First, the FP project information is retrieved and saved independently. Then, the productivity project information that is required during the conversion process is fetched. It also produces data by generating an output excel spreadsheet that contains the conversion results.

Business logic component is the main component that holds the technical application of the forward conversion approach, which is presented in section 4. In addition, it is the responsibility of this component to keep a historical record about each resulting OP and its corresponding consumed FPs.

Finally, a specialized graphical user interface component has been designed and developed, as shown in Figure 3, to facilitate the system customization and configuration to fit the different development environments. This enabled the user to overwrite the empirically calculated default values of the OP type's contribution proportions, OP complexity level contribution proportions, and input/output data files physical locations.

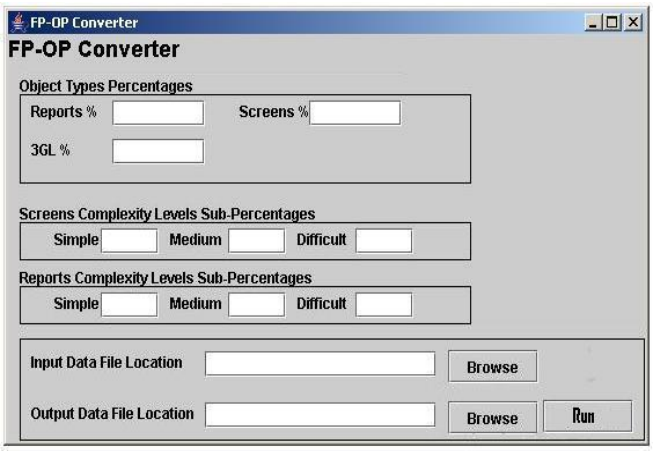

Figure 3. FP-OP converter system graphical user interface.

6. Empirical Results Evaluation

The source of the FP projects empirical data is the ISBSG Release 9 data repository [6]. Currently, it holds information about 3,024 software projects gathered from different organizations around the world. Several fidelity and sizing filtering criteria [7] have been applied to ISBSG repository to extract the most appropriate projects for the empirical and critical evaluation of the proposed forward conversion approach. This resulted in reducing the selected number of ISBSG projects to 66 as detailed in [7]. Consequently, several statistical tests, as discussed in sections 6.1 and 6.2, have been performed to investigate the reliability of the defined FP-OP relationship in addressing the research questions.

6.1. Reliability of the Forward FP-OP Conversion Approach

The relationship between the calculated and actual ISBSG data attributes has been assessed using both Spearman's correlation coefficients and paired sample T-test significance measures [5, 7]. The outcomes from investigating the relationship between the forward conversion approach estimated effort and ISBSG actual effort have shown high correlation and significance, 88% and 0.33, respectively. Further highly desirable correlation, 87%, has been discovered

between the calculated AOP and both FP types; unadjusted and adjusted. Having the high correlation and significance relationships between the main output attributes of the forward conversion approach and the ISBSG actual data can be considered as supporting the reliability of the defined FP-OP elements mapping and consumption schemes.

6.2. FP-OP Relationship Modelling

The construction of the anticipated FP-OP conversion table needs a large volume of historical data to cater for the increasing number of programming languages and programming language generations. Moreover, it should be regularly maintained to consider the emerging versions of current and new programming languages. Therefore, one could conclude that the relationship between FP and OP should be independent from any external factors that might limit its stability and reliability. Thus, the embedded relationship between FP and OP sizing metrics should be modelled using the converted OP data.

FP effort is calculated as:

$$FPEffort = Total\ AFP/FP\ Productivity$$

Similarly, OP effort is calculated as:

$$OPEffort = Total\ AOP/OP\ Productivity$$

Having available the completed and delivered ISBSG projects, allowed a comparison of the two predicted effort estimates with the actual expended effort. Given the high correlation, we can assume that the effort estimates can be deemed equal. Hence, equating FP and OP effort formulae results in one consolidated formula as:

$$\frac{TotalAFP}{FP\ Productivity} = \frac{TotalAOP}{OP\ Productivity}$$

Re-factoring this formula has resulted in obtaining a linear relationship between FP and OP as:

$$TotalAOP = TotalAFP \times \frac{OP\ Productivity}{FP\ Productivity}$$

Hence, the relationship between OP and FP sizing metrics is governed by an adjustment factor being the ratio between the OP and FP productivities, and vice versa. Accordingly, two linear curve fitting statistical studies have been conducted to investigate the fitness of two specialized productivity factors and FP type dependent FP-OP conversion models:

1. Unadjusted FP (UFP) dependent model:

$$\begin{aligned} AOP = {} & (0.66\times UFP) + (-0.27\times UFP\times Z_1) \\ & + (-0.18\times UFP\times Z_2) + (-0.29\times UFP\times Z_3) \\ & + (-0.26\times UFP\times Z_4) + 30.87 \end{aligned}$$

2. Adjusted FP (AFP) dependent model:

$$\begin{aligned} AOP = {} & (0.58\times AFP) + (-0.28\times AFP\times Z_1) \\ & + (-0.13\times AFP\times Z_2) + (-0.27\times AFP\times Z_3) \\ & + (-0.23\times AFP\times Z_4) + 33.17 \end{aligned}$$

where Z_1, Z_2, Z_3, and Z_4 are project size and programming language generation indicator variables as summarized in Table 10.

Table 10. Specialized FP-OP conversion models productivity factors indicators.

Indicator Variable	Value	Productivity Factor
Z_1	0 or 1	Small Size Project
Z_2	0 or 1	Medium Size Project
Z_3	0 or 1	Third Generation Language Project
Z_4	0 or 1	Fourth Generation Language Project

The fitness of the resulting FP-OP conversion models has been measured by R^2 statistical fitness measure. R^2 is the proportion of variation in the dependent variable, AOP [5]. The values of R^2 range from 0 to 1. Small values indicate that the model does not fit the data well. Finally, the fitness, R^2, of the resulting UFP/AFP-OP conversion models is 0.95.

7. Conclusion and Future Work

A novel approach to bridge the relationship between FP and OP sizing metrics has been proposed. The intension, as per the defined research questions, is to investigate the applicability and reliability of inferring and reusing the OP information of a software project from its available FP information. Hence, the relationship between the elements of OP and FP sizing metrics has been bridged in two phases. First, a mapping scheme between the elements of the two sizing metrics has been defined. Then, the second phase focused on quantifying the defined FP-OP relationship.

A multi-phase forward conversion approach has been proposed to infer the OP information for 66 ISBSG multi-organizational historical projects using the defined FP-OP elements mapping and consumption schemes. Consequently, the reliability of the defined FP-OP relationship has been evaluated by assessing the resulting OP attributes against the actual ISBSG FP and effort attributes. The evaluation of the conversion approach empirical results showed high correlation, 88%, between the OP estimated and FP actual efforts that generally supports the reliability of the defined FP-OP elements mapping and consumption schemes. Furthermore, a high correlation, 87%, has been found between the calculated AOP and unadjusted/adjusted FP. This confirms the hypothesis that motivated this research to reveal the existence of a hidden relationship between the two size metrics as defined in the proposed FP-OP relationship. The low level investigation of the above correlation in the performed

statistical studies showed that the lower the underlying programming language generation is, the higher the correlation between FP and OP size metrics.

The high AOP and FP correlation has been used to relate the FP and OP effort formulae to discover the embodied relationship between them. It has been demonstrated that the AOP is linearly related to the FP by an adjustment factor, and vice versa. Statistically, two productivity factors and FP type dependent linear models have fit the relationship between FP and AOP with high statistical fitness, R^2, values, 0.95, for both models.

To facilitate the conversion process, a 3-tier architecture [14] FP-OP converter has been developed using JAVA due to its diverse features that suite the purposes of the current and planned research. Future work is being planned to integrate the FP-OP converter in a multi-model software cost estimation CASE tool that suits the different stages of the software development life cycle.

Several requirements elicitation and modelling techniques have been used to model and specify system requirements [11, 14]. Use case elicitation and modelling is one of the techniques that have been used extensively in the literature. Subsequently, several use case model based software cost estimation models [3, 8] have been developed since use case models capture, relatively, an accurate representation of the users' requirements. Therefore, further work is being carried out to integrate the resulting FP-OP conversion models with earlier work [8] specialized in building a use case model and OP based software cost estimation model that is applicable at the early stages of the software development life cycle. In parallel, reusing the generated OP data in building and validating OP based software cost estimation models is being evaluated by the research team.

Acknowledgements

Ayman Issa would like to thank Philadelphia University, www.philadelphia.edu.jo, for sponsoring this research programme. Also, the authors would like to thank ISBSG and in particular Peter Hill for granting them permission to utilize the data repository in this research.

References

[1] Banker R., Kauffman R., and Kumar R. "An Empirical Test of Object-Based Output Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment," *Journal of Management and Information Systems*, vol. 8, no. 3, pp. 127-150, 1992.

[2] Boehm B., Abts C., Brown A., Chulani S., Clark B., Horowitz E., Madachy R., Reifer D., and Steece D., *Software Cost Estimation with Cocomo II*, Prentice Hall, 2000.

[3] Damodaran, M. and Washington, A., "Estimation Using Use Case Points" *in Proceedings of ISECON 2002,* San Antonio, 2002.

[4] Deitel H. and Deitel P., *Java: How to Program*, Upper Saddle River, Pearson Education International Prentice Hall, 2003.

[5] Gerber S., Voelkl K., Anderson T. W., and Finn J. D., *The SPSS Guide to The New Statistical Analysis of Data*, Springer-Verlag, New York, 1997.

[6] ISBSG, *Projects Data Repository (Release 9)*, available at: http://www.isbsg.org.au/html/index2.html, 2004.

[7] Issa A., Odeh M., and Coward D., "FP-OP Conversion," *Technical Report*, Centre of Complex and Cooperative Systems, University of the West of England, UWE-CEMS-TR-CCCS-0002, 2005.

[8] Issa A., Odeh M., and Coward D., "Using Use Case Models to Generate Object Points," *in Proceedings of the IASTED International Conference on Software Engineering,* Austria, pp. 468-473, 2005.

[9] Jones C., *Applied Software Measurement: Assuring Productivity and Quality*, McGraw-Hill, London, 1997.

[10] Lokan C., "An Empirical Analysis of Function Point Adjustment Factors," *Information and Software Technology*, vol. 42, no. 9, pp. 649-659, 2000.

[11] Major M., "A Qualitative Analysis of Two Requirements Capturing Techniques for Estimating the Size of Object-Oriented Software Projects," *Object Technology Group*, Department of CS, Clemson University, 1996.

[12] Matson J., Barrett B., and Mellichamp J., "Software Development Cost Estimation Using Function Points," *IEEE Transactions on Software Engineering*, vol. 20, no. 4, pp.275-287, 1994.

[13] Smith J., "The Estimation of Effort Based on Use Cases," *Rational Software*, available at: http://www.rational.com/products/whitepapers/TP171.jsp, 2003.

[14] Sommerville I., *Software Engineering*, Addison-Wesley, Harlow, England 2001.

[15] SPR, "Programming Languages Table," *Software Productivity Research*, available at: http://www.spr.com/products/programming.shtm 2005.

[16] Stutzke R., "Experience with the COCOMO II Application Point Model," *in Proceedings of the 15th International Forum on COCOMO and Software Cost Modeling*, Los Angeles, pp. 1-17, 2000.

[17] United Kingdom Software Metrics Association, *MKII Function Point Analysis Counting Practices Manual*, UKSMA Metrics Practices Committee, UK, 1998.

Ayman Issa received his BSc and MSc degrees in computer science from the University of Jordan in 2000 and 2003, respectively. In 2003, he joined the University of West of England, Bristol, UK as a software engineering PhD researcher and visiting lecturer. He has four years of experience in quality assurance of e-business applications. In addition, he has been appointed in several management posts. His research interests include use case modelling, use case patterns, software cost estimation, software metrics, software complexity, and software development life cycles.

Mohammed Odeh is a senior lecturer in software engineering and the leader of the Software Engineering Research Group of the Complex Cooperative Systems Centre in the Faculty of CEMS at the University of West of England, Bristol, UK. He holds PhD degree in computer science from the University of Bath, 1993 in addition to PGCert in higher education and ITL membership. He has more than 20 years of experience in software engineering including research and development, extensive project management experience in planning and leading a range of IT-related projects in addition to management posts. He is a co-investigator on EU-funded projects, and led the second work-package (the user-requirements specifications) of the MammoGrid project, an FP5 EU-funded project with collaboration from European partners such as Oxford University, Cambridge University Hospital, CERN, Udine University Hospital in Italy, and Mirada Solutions Limited. Dr. Odeh has been supervising six PhD students, with one successful completion, four expected completions in 2006, and one in 2007. His research interests are mainly focused on bridging the gap between knowledge, business, and system models in addition to software cost estimation and requirements engineering processes.

David Coward received his BSc in computing and statistics, in 1978, a PhD degree in computer science from the Open University, UK, in 1992, and also a Cert Ed (Further and Higher Education). He is a principal lecturer in computer science. Currently, he is a head of the School of Computer Science in the Faculty of CEMS at the University of West of England, Bristol, UK. He has been in higher education for more than 20 years. His research interests include software engineering in particular software validation, software metrics and cost estimation.