UQÀM

Software Engineering Management Research Laboratory

Software Engineering Laboratory in Applied Metrics

FUNCTIONAL SIZE MEASUREMENTS OF THE TACS SOFTWARE APPLICATION

Delivered to

Captain Mark Jennings
Military and Training Systems Life Cycle Application Manager
Land Software Engineering Centre
National Defence of Canada

Prepared by

Serge Oligny, M.Sc.
Jean-Marc Desharnais, Adm. A., M.Sc.

June 1998

Copyright notice

Author biographies and affiliations

Serge Oligny

Mr. Oligny is currently Director of Technological Innovations at the Software Engineering Management Research Laboratory. He holds an M.Sc. degree from the University of Sherbrooke (1988) and has accumulated over 13 years of industry experience as a software project manager in the consulting industry and as a corporate software development manager in the pulp and paper industry. He has a long-standing professional interest in software functional size measurement and has published several papers on the topic of software engineering management in America, Europe and Asia.

Jean-Marc Desharnais

Mr. Desharnais is a Research Associate with UQAM and is also a director at SELAM. He is one of the co-authors of the Full Function Point (FFP) measurement technique. Mr. Desharnais is a specialist in software engineering metrics. He has carried out a number of software engineering research projects covering estimating, assessment, budgeting and productivity evaluation. Mr. Desharnais has also evaluated productivity levels in several organizations around the world and set up quantification programs to include the assessment, productivity, quality and budgeting of systems maintenance. He has experience in counting real-time systems with FFP in Australia, Canada and the USA. He holds a Master's degree in Computer Management - University du Québec à Montréal, a Master's degree in Public Administration - *Ecole nationale d'administration publique* and a Diploma in Administration - Laval University.

Software Engineering Management Research Laboratory - UQAM

The Research Laboratory is part of the Département d'Informatique of the Université du Québec à Montréal (UQAM). The Laboratory is mainly financed through a partnership with Bell Canada, with additional financing provided by the National Research Council of Canada. The Laboratory's mission is to develop, for the software engineering industry, analytical models and measurement instruments to enable organizations to improve their decision-making processes in order to meet their business objectives.

Software Engineering Laboratory in Applied Measurements (SELAM)

SELAM is a private consulting firm which was established in Montréal in 1993. SELAM supplies its clients with practical implementations of software measurement programs to manage, develop and maintain their information systems; offers workshops on various types of software metrics use, including Function Point Analysis and Full Function Points; sets up measurement databases with information provided by organizations which could be used to identify measurement program objectives; and collaborates with national and international research organizations on the topic of software measurement.

Full Function Points (FFP)

FFP is a software size measurement technique which has been developed jointly by UQAM and SELAM. This particular effort was also financed by Bell Canada, Nortel, Hydro-Québec and JECS Systems Co. Ltd. (Tokyo, Japan). Complete documentation and some field-test results can be found on the Web at http://www.info.uqam.ca/Labo_Recherche/Lrgl/ffp.htm.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The Land Software Engineering Centre (LSEC) maintains a significant portfolio of real-time and embedded software applications for the benefit of the Canada's Department of National Defence. A parametric cost estimation model is used by LSEC personnel to assist in project decision-making concerning this portfolio. A key input of this estimation model, the size of the software application measured in source lines of code (SLOC), is available relatively late in the software engineering process. Thus, the results obtained with this tool bear a level of risk which could be avoided by the use of a software functional size measure, available much sooner in the software engineering process. Therefore, the LSEC seeks to recalibrate its estimation model based on an adequate functional size measure. Three of these measures are of particular interest to the LSEC: Function Points and Full Function Points and Feature Points.

The Software Engineering Management Research Laboratory of the Université du Québec à Montréal (UQAM), in partnership with the Software Engineering Laboratory in Applied Metrics (SELAM), maintains a long-standing professional interest in the functional size measurement of software. Both organizations routinely perform field-testing and usage analysis of software functional size measures.

In April 1998, UQAM proposed to LSEC that UQAM conduct a comparative study of the above three functional size measures by counting them on the TACS software application maintained by LSEC. The work was performed at no cost to the LSEC, with the help of resources from SELAM. It is the value of the results of the analysis that is of interest of UQAM and SELAM. Counting was completed during the month of May 1998, and the results are presented in this report.

Key findings of this analysis demonstrates that Full Function Points is more appropriate than Function Points for the needs of the LSEC. It is therefore recommended that the LSEC adopt Full Function Points as its standard functional size measure for software applications.

It is further recommended that some effort be invested by the LSEC in developing basic tools to automate part of the counting exercise, and that training on the technique be provided to a small core of permanent LSEC personnel. Finally, some basic guidelines are proposed on the calibration of LSEC estimation models based on the historical gathering of Full Function Point counts.

1. CONTEXT AND MANDATE

This section briefly outlines the motivations behind this report and states the terms of the mandate that led to it.

CONTEXT

During the month of February 1998, the Treasury Board Secretariat of Canada initiated a mandate to prepare IT governance guidelines and to identify business requirements for the development of a government-wide IT measurement database. This mandate was awarded to the firm of Godcharles, Goulet, Fournier in Ottawa, with UQAM's Software Engineering Management Research Laboratory in Montreal (UQAM) as a subcontractor.

An essential component of this mandate was the contribution provided by a workgroup composed of representatives from a reasonable sample of the Canadian Government's IT departments and an international review board. The Land Software Engineering Centre (LSEC) at the Department of National Defence was a member of this workgroup, and was represented by Captain Mark Jennings and Vern French of PRIOR Data Sciences Ltd., a significant contractor for the LSEC.

It became clear that, in the area of software measurement, the knowledge and expertise level available at the Land Software Engineering Centre was clearly above the average level of the other departments represented in the workgroup.

Given that software measurement lies at the core of the business mission of the UQAM Software Engineering Management Research Laboratory, a mutual interest relationship quickly developed between the two organizations.

Thus, convergent business interests lie at the core of the mandate that led to the delivery of this report.

MANDATE

The LSEC has been using a parametric software cost estimation model for many years. This model is essentially based on the original COCOMO model proposed by Boehm in 1981, refined and customized for the specific needs of the Centre. The model generally performs well, although it has some weaknesses which are offset by the high degree of expertise provided by one external expert from PRIOR Data Sciences Ltd.

The main weakness of the model lies in the nature of its most important input parameter: the size of the application to be developed. As with most COCOMO-based estimation models, this one uses the number of source lines of code (SLOC), a measure of software size, as the key driver for estimating effort. From an estimation perspective, this size measure suffers from three drawbacks:

a) Its actual value is known only quite late in the development process, at the end of the programming phase (typicaly after 65% to 75% of a project effort has been expended), thus forcing the use of an early SLOC estimated size value to complete the initial cost estimate;

b) SLOC is a technical measure which is only remotely connected to the requirements of the software as perceived by its users and owners. Although some conversion “factors” have been proposed in the marketplace to translate this measure into a functional size closer to the perspective of the users and owners, these are highly controversial, both in the marketplace and the academia;

c) There is no single and generally accepted standard for counting source lines of code, which forces the use of a few of them (as illustrated in the estimation tools used at the Centre) to generate different estimation scenarios thus, introducing an avoidable degree of risk in the cost estimation process.

The LSEC is therefore seeking to reduce the risk induced by these elements on its software cost estimation process by using an alternative software size measure.

Given the actual state of knowledge in the area of software measurement, supported by a significant amount of research, it is recognized by the Centre and by UQAM that functional size measures offer the best available alternative for size measurement.

Function Point Analysis (FPA) is the most widely used functional size measure in the industry. It offers reasonable coverage of the weaknesses outlined above in the MIS domain, but has often been recognized as ill-suited to measuring the size of real-time systems. Real-time systems make up a very significant part of the LSEC’s application portfolio. An alternative functional size measure, called Full Function Points (FFP), was proposed in 1997 jointly by UQAM and the Software Engineering Laboratory in Applied Metrics (SELAM). Full Function Points was designed from the start to address the particularities of real-time systems. Feature Points is an extension of FPA which has been proposed in the marketplace for real-time systems.

As part of an ongoing effort to field-test Full Function Points, UQAM proposed to the LSEC that UQAM conduct a comparative analysis of FPA, FFP and Feature Points by counting all three on one real-time software application maintained by the LSEC. This analysis would be performed at no cost to the LSEC; the interest of UQAM and SELAM lies in the value of field-test results. The interest of the Centre lies in the insights to be gained into the selection of an appropriate alternative size measure for use within its parametric cost estimation model. This mandate has been performed by Jean-Marc Desharnais, Marcela Maya and Serge Oligny from UQAM who are all experienced Function Point counters. Mr. Desharnais and Mrs. Maya are also experienced Full Function Point counters.

2. FUNCTIONAL SIZE MEASUREMENT RESULTS

2.1 Documentation used

The documentation used for all counts was the "*software requirements specification*" of the TACS application, as at September 1, 1994. This document, in two volumes, complies with the 2167 military standard for software development; it was produced by PRIOR Data Sciences Ltd. and filed under reference number W5825-1-AC08/01-ET by the Department of National Defence.

For the purpose of measuring the functional size of this application, the documentation was very detailed. No other documents were required to adequately complete all counts, although some explanations were required from a TACS application expert to correctly interpret the content of the documentation in a functional measurement context. Adequate understanding of the documentation for measurement purposes was verified by counting roughly a third of the functionality without the presence of the application expert and, by subsequently verifying the count in his presence. This exercise was conclusive.

2.2 Establishing the TACS application boundary

Establishing the correct functional boundary (what is to be counted) of an application is the first key step in most functional size measurement techniques, including Function Points and Full Function Points. Based on the available documentation, no particular problems were encountered in establishing the functional boundary of the TACS software application.

The documentation used for the counts can be divided into three parts. A first part (essentially Volume 1) contains the specifications of the TACS software from a user perspective. A second part (some in Volume 1 and the rest in Volume 2) contains the specifications for the TACS database initialization, a process that is transparent to the users of the application. A third part (Volume 2) contains technical specifications as they relate to the type of equipment and telecommunications links to be used.

Since an FPA count is restricted to the functionality directly available to the users, and in order to compare the counts for the same object, the counts were restricted to the specifications found in Volume 1 of the documentation. Some effort was required to gather a single, non-redundant list of processes from the documentation in order to eliminate duplicate counts. Potential redundancies in the data counts were eliminated in the same fashion.

2.3 TACS Project: Selected measurements

Table 2.3.1 below presents some key project measurements from the TACS project. All figures presented in Table 2.3.1 are actuals which were compiled after project completion. It is to be noted that actual project cost and effort do not include the planning and requirements phase, completed prior to project initialization.

Measure	Value	Unit
Project cost (1)	563 000	$
Project duration	14	months
Project effort (1)	993	person-days
Project max. team size	7	persons

(1): excluding the planning and requirements phase

Table 2.3.1 – TACS project key measurements

It is to be noted that although the TACS software application cost to the Department of National Defence is 563 000$, the actual cost to the supplier is usually considered higher, based on exchange of information between the DND and the supplier. Since the TACS software application was developed from a fixed price contract, it is not possible to establish how much more cost was incurred by the supplier in this project. This fact does not change the value of the asset to the DND but it does provide a base for the interpretation of the performance measures derived from cost. Therefore the above cost figure must be interpreted as a “minimum cost”.

2.4 Feature Point measurement results

The distinguishing characteristic of Feature Points is the count of “*algorithms*”. These algorithms were counted for a few processes. It was observed that the definition of an algorithm, according to available documentation, did not offer a degree of rigor comparable to that of other definitions proposed by FPA or FFP. Feature Point counts could not therefore offer the level of reliability afforded by either FPA or FFP. Consequently, the Feature Point count was dropped.

This weakness of Feature Points has been documented in the literature. It was the first opportunity the authors had to observe it directly on a non-academic software application.

2.5 Function Point measurement results

Table 2.5.1 below presents the Function Point count summary for the 25 elementary processes identified within the TACS software application. A detailed table of the count for each identified process is presented in Appendix B.

ELEMENTARY PROCESSES		FPA COMPONENTS			FPA
ID	DESCRIPTION	EI	EO	EQ	PTS
321	System initialization	0	2	0	8
3221	Operator interface initialization	1	0	1	6
3222	Menu display	0	0	1	3
3223	Menu input	0	0	1	3
3224	Message monitor	0	0	0	0
322511	Target manipulation command processor initialization	0	0	0	0
322512	Target verification test command processor	2	0	1	9
322513	Target elevation test command processor	1	1	0	7
322514	Radio link test command processor	0	1	0	4
322521	Sequence manipulation command processor initialization	0	0	0	0
322522	Sequence creation/modification command processor	2	1	1	13
322524	Battlerun creation/modification command processor	2	2	1	17
322525	Battlerun execution command processor	0	1	0	4
322526	Manual run execution command processor	0	0	0	0
322527	Sequence simulation command processor	0	2	0	8
322531	TACS database command processor initialization	0	0	0	0
322532	Copy sequence definitions command processor	0	2	0	8
322533	Copy battlerun definitions command processor	0	0	0	0
322534	Delete sequence definitions command processor	0	0	0	0
322535	Delete battlerun definitions command processor	1	0	0	3
322536	Delete hit data command processor	0	0	0	0
322541	Hit data manipulation command processor initialization	0	1	0	4
322542	View hit data results command processor	0	0	0	0
322543	Generate hit data report command processor	0	0	0	0
32255	Exit command processor	0	0	0	0
	Error messages display	0	1	0	4
	TOTAL	9	14	6	**101**

Table 2.5.1 – Summary of FPA count for TACS elementary processes

Table 2.5.2 below presents the Function Point count for the Logical Data Files identified within the TACS software application.

ID	DESCRIPTION	RET	ILF	EIF	FPA COUNT
	Menu file	1		1	5
	TACS	1	1		7
	TACS lookup	1	1		7
	TOTAL		2	1	**19**

Table 2.5.2 – Summary of FPA count for TACS logical data files

Table 2.5.3 below summarizes the results of the Function Point count for the TACS software application.

FPA COUNT ITEMS	Occ.	Points	% Tot. FPA
EI - External Inputs	9	27	23%
EO - External Outputs	14	56	47%
EQ - External Inquiries	6	18	15%
ILF - Internal Logical Files	2	14	12%
EIF - External Interface Files	1	5	4%
TOTAL		120 FPA	

Table 2.5.3 – Summary of FPA count for the TACS application

2.6 Full Function Point measurement results

Table 2.6.1 below presents the Full Function Point count summary for the 25 control processes identified within the TACS software application. A detailed table of the count for each identified subprocess is presented in Appendix A.

CONTROL PROCESSES		SUB_PROCESSES OCCURRENCES				FFP COUNT
ID	DESCRIPTION	ECE	ECX	ICR	ICW	PTS
321	System initialization	3	4	3	1	11
3221	Operator interface initialization	1	2	2	3	8
3222	Menu display	1	1	0	0	2
3223	Menu input (user modifiable)	0	1	1	0	2
3224	Message monitor	1	1	1	0	3
322511	Target manipulation command processor initialization	1	1	0	0	2
322512	Target verification test command processor	2	3	2	2	9
322513	Target elevation test command processor	1	2	2	2	7
322514	Radio link test command processor	2	2	1	1	6
322521	Sequence manipulation command processor initialization	1	1	0	0	2
322522	Sequence creation/modification command processor	1	9	2	2	14
322524	Battlerun creation/modification command processor	2	3	3	2	10
322525	Battlerun execution command processor	1	4	1	1	7
322526	Manual run execution command processor	3	12	4	2	21
322527	Sequence simulation command processor	3	6	3	1	13
322531	TACS database command processor initialization	1	1	1	0	3
322532	Copy sequence definitions command processor	2	4	2	1	9
322533	Copy battlerun definitions command processor	1	5	3	2	11
322534	Delete sequence definitions command processor	1	2	1	1	5
322535	Delete battlerun definitions command processor	1	2	1	2	6
322536	Delete hit data command processor	1	2	1	1	5
322541	Hit data manipulation command processor initialization	1	1	1	0	3
322542	View hit data results command processor	2	5	3	0	10
322543	Generate hit data report command processor	2	3	2	0	7
32255	Exit command processor	1	1	1	0	3
	TOTAL	36	78	41	24	**179**

Table 2.6.1 – Summary of FFP count for TACS control processes

Table 2.6.2 below presents the Full Function Point count for the Data Control Groups (DCG) identified within the TACS software application.

ID	DESCRIPTION	RET	ICG	ECG	FFP COUNT
	Menu file	1		1	5
	TACS	1	1		7
	TACS lookup	1	1		7
	TOTAL		2	1	**19**

Table 2.6.2 – Summary of FFP count for TACS data control groups

Table 2.6.3 below summarizes the results of the Full Function Point count for the TACS software application.

FFP COUNT ITEMS	Occ.	Points	% Ctl. Proc.	% Tot. FFP
ECE - External Control Entries	36	36	20%	
ECX - External Control Exits	78	78	44%	
ICR - Internal Control Reads	41	41	23%	
ICW - Internal Control Writes	24	24	13%	90%
ICG - Internal Control Groups	2	14	n/a	
ECG - External Control Groups	1	5	n/a	10%
TOTAL		**198 FFP**		

Table 2.6.3 – Summary of FFP count for the TACS application

3. FUNCTIONAL SIZE MEASUREMENT ANALYSIS

Based on the documentation described under section 2.1, this section presents notes and comments relating to the FPA and FFP counts.

3.1 Counting effort

The effort devoted to the counting exercises was, in the author's opinion, slightly higher than usual. This fact is attributed to the high level of detail in the documentation used for the counts. This level of detail derives from the use of a specific documentation standard (2167 military standard) which is significantly more exhaustive than the type of documentation generally encountered in the industry. Furthermore, some effort was expended in transferring basic knowledge of Full Function Points to the application expert in order to further compare FPA and FFP through discussion.

3.2 Function Point count

Elementary processes

The correct identification of the elementary processes of an application is critical to the reliability of a Function Point (FPA) count. According to IFPUG *Counting Practices Manual (v. 4.0)*, an elementary process is defined as "*the smallest unit of activity that is functionally meaningful to the end users*". This manual expands on the definition by specifying that an elementary process must be *self-contained* or *complete* from a user's perspective and it must leave the application's data in a functionally coherent state. For most of the functionality within the TACS application, the definition of elementary process was not respected during the count.

Example

Two external inputs (EI) and one external output (EO) were counted for the TACS application process 3.2.2.5.1.2 (Target verification test command processor). As stated above, each elementary process (EI or EO) must be *self-contained*. In this example, only part of an EI process was counted based on the following specifications:

- OP50 states: "*If a flashing TEU number is selected, the TEU number shall be removed from the list of TEUs present on the target range.*"
- OP51 states: "*If a TEU number that is not flashing is selected, the TEU number shall be added to the list of TEUs present on the target range.*"

In both cases, the process is complete only when the result is displayed on the video screen.

- OP 52 states: "*If a TEU is removed from the list of TEUs present on the target range, the displayed TEU number shall be shown in normal video.*"
- OP 53 states: "*If a TEU is added to the list of TEUs present on the target range, the displayed TEU number shall be shown in flashing video.*"

A *complete* or *self-contained* process would require that the application display the TEU number using a normal or flashing video message after the TEU number has been added or removed. A *complete* elementary process does NOT consist in just adding or removing a TEU number, it also includes the display of that number. Therefore, counting two different function types in this case (one EI for add/remove and one EO for display) does not comply with the definition of an elementary process.

Distinguishing EI, EO and EQ

A further impediment to the Function Point count was the segregation of EI, EO and inquiry (EQ). The IFPUG *Counting Practices Manual (v. 4.0)* requires that elementary processes be segregated according to their predominant functional role of either *inputting (including update)* OR *extracting* data. This predominant functional role is then used to determine the type of each elementary process (EI, EO or EQ). At the external application level, such a model usually works well for MIS and commercial software. However, it is often ill-suited when applied to embedded software which monitors or controls hardware equipment and operates in real-time, as documented in the literature. The TACS software application is, essentially, such a real time system.

Points assignment

Under the provisions stated above, no particular problems were encountered in the assignment of points (actual Function Points) to each elementary process given the amount of detail in the individual specifications (OPs).

3.3 Full Function Point count

Control processes

The correct identification of the control processes of an application is critical to the reliability of a Full Function Point (FFP) count. According to *Full Function Points: Counting Practices Manual (November 1997)*, a control process is defined as "*a process that controls, directly or indirectly, the behavior of an application or a mechanical device.*" This definition is expanded by stating that control processes must be identified from a *functional perspective*. A functional perspective is further defined as "*the point of view of the functionality delivered by the application; it excludes implementation and technical considerations.*"

All relevant TACS application processes, as defined by the functional hierarchy found in the documentation, could be mapped exactly to this definition.

Generic sub processes

The next step in an FFP count is to identify the generic control sub processes within each control process. There are four generic control sub processes: external control entries (ECE), external control exits (ECX), internal control reads (ICR) and internal control writes (ICW). Typical key words (read, display, etc.) found in the specifications (OPs) of each identified control process of the TACS application were used to locate and count the generic sub processes. No particular problems were encountered during this exercise.

Points assignment

No particular problems were encountered in the assignment of points (actual Full Function Points) to each generic control sub process, given the fairly low level of detail in the individual specifications (OPs). Furthermore, since generic control sub processes are at the lowest functional level, the assignment of points is straightforward and very rarely requires the counter to refer to point assignment tables. No such lookup was necessary while the TACS application was counted.

3.4 Comparing the FPA measurement method with FFP

This section discusses two key issues arising from a comparison between the measurement method proposed by FPA and the one proposed by FFP.

Coverage of the counting methods

The FPA *Counting Practices Manual* indicates that only the elementary processes interacting directly with the users of a software application are to be counted. This counting rule provides generally adequate coverage of the elementary processes for most MIS software applications. The functional nature of many real-time or embedded software applications is usually such, however, that many functional elementary processes will not interact directly with the users. The FPA counting method will therefore ignore these processes which, in the eyes of software engineers, can amount to significant pieces of the measured application.

Furthermore, according to the FPA *Counting Practices Manual*, a process occurring several times in an application must be counted only once. While this approach offers the benefit of producing a "pure" functional count, it also reveals a drawback: the ability to measure the functional size of potentially reusable functionality is lost.

The combined effect of these two characteristics of the FPA counting method is well illustrated for the TACS software application in Appendix B. The table clearly shows which processes of the TACS application were not accounted for in the FPA count by displaying a shaded box in the count column.

The FFP measurement method does not suffer from these two limitations. It can therefore be said that the FFP measurement method offers a better coverage of the functional size of a software application than the FPA measurement method.

Precision of the counting methods

By definition, the user's perspective of a software application adopted by an FPA count does not explicitly account for the internal processing of the elementary processes. This aspect of elementary processes is accounted for indirectly since, for instance, an external input (EI) entails a logical file (ILF) update. Obviously, to deliver that EI to the users of an application, some internal processing is required for information to be written to the ILF. Conversely, an external output (EO) entails some internal processing to permit reading from either an ILF or an EIF. Since such relations between external processes and internal processes are implicit, the FPA measurement technique does not afford their precise quantification. The complexity adjustment factor associated with an EI, for instance, will not allow an elementary process requiring the update of 4 ILF to be distinguished from another requiring the update of 15 ILF.

Although this characteristic has not been a subject of debate when MIS software applications are counted, it has been a significant handicap in the eyes of real-time and embedded software developer mainly because the number of such internal processing pieces a) varies a lot from process to process, and b) can often be very considerable in many real-time processes.

By favoring a pure functional perspective of a software application, FFP counts make that relation between external processes and internal processes explicit. The external control entries and exits (ECE and ECX) account for the external characteristics of the identified processes, while the internal control reads and writes (ICR and ICW) account for the internal characteristics of those same processes. Table 3.4.1 below illustrates this relationship using the TACS FFP count; roughly two-thirds of the processing of this application is "external", while the remaining third is "internal". Deriving such a proportion from the FPA count of the TACS application would be much more speculative.

FFP PROCESSES ITEMS	Occ.	Points.	Proportion
ECE - External Control Entries	36	36	
ECX - External Control Exits	78	78	64%
ICR - Internal Control Reads	41	41	
ICW - Internal Control Writes	24	24	36%
TOTAL		179	100%

Table 3.4.1 – Proportion of internal vs. external processing in the TACS application

The ability to quantify this relation more precisely using FFP enables its users to better understand possible future variations in the size of the application and, most probably, to offer better explanations for effort, costs or schedule variations. Previous field tests have also shown that, from a software engineering perspective, the FFP counting technique offers a more accurate measurement of the functional size of real-time or embedded software applications.

3.5 Selected development ratios

Many ratios can be calculated by combining functional size measurements with project measurements. Three of them usually provide a reasonable picture of a project from an overall economic/process perspective: unit cost, unit effort and average rate of delivery.

Unit effort

Unit effort is defined as the average amount of labor (measured in person-hours) to deliver one unit of functional size. Unit effort can be calculated for projects involving individual software applications or averaged out for an entire portfolio of projects. It usually includes all direct labor and the project's specific overhead (management, technical support, etc.). In any case, meaningful interpretation of unit effort, for benchmarking purposes for instance, must take into account what it includes and what it does NOT include.

As stated in section 2.3 above, the observed unit effort (actual) of the TACS project includes all direct labor expended after completion of the plans and requirements and up to the turnover of the application to the maintenance team, including project management. It EXCLUDES the labor expended to complete the plans and requirements.

Unit effort figures for the TACS project are presented in Table 3.5.1 below, in both person-hours/FPA and person-hours/FFP, assuming that one person-day is equivalent to 7,5 person-hours.

UNIT EFFORT BASIS	SIZE	EFFORT	UNIT EFFORT
FFP - Full Function Points	198 FFP	7 448 ph	37,6 ph / FFP
FPA - Function Points	120 FPA	7 448 ph	62,1 ph / FPA

Table 3.5.1 – TACS project unit effort

Unit cost

Unit cost is defined as the average cost (in dollars) to deliver one unit of functional size. Unit cost can be calculated for projects involving individual software applications or averaged out for an entire portfolio of projects. It usually includes all direct costs (manpower and material) and the project's specific overhead (management, technical support, etc.). Depending on internal accounting practices, some organizations might add an additional general administrative cost that is often allocated on the basis of the amount of capital mobilized by a project, thus yielding a *fully burdened unit cost*. In any case, meaningful interpretation of unit cost, for benchmarking purposes for instance, must take into account what it includes and what it does NOT include.

As stated in section 2.3 above, the observed cost (actual) of the TACS project includes all direct costs incurred after the completion of the plans and requirements and up to the turnover of the application to the maintenance team, including project management. It EXCLUDES general administration costs and the cost incurred to complete the plans and requirements.

Unit costs of the TACS project are presented in Table 3.5.2 below, in both $/FPA and $/FFP. It is the first time that the authors have been able to provide a figure in $/FFP since development costs were not available during previous field-testing of this functional measure.

UNIT COST BASIS	SIZE	COST	UNIT COST
FFP - Full Function Points	198 FFP	563 000 $	2 843 $ / FFP
FPA - Function Points	120 FPA	563 000 $	4 692 $ / FPA

Table 3.5.2 – TACS project unit costs

It is difficult to provide a meaningful interpretation of a unit cost figure in the context of benchmarking, since most data available today do not unambiguously specify **a)** which costs are included and which are excluded, **b)** what the exact nature is of the work included and/or excluded from a given figure, and, **c)** to a lesser degree, since it can be conceived of as an inherent aspect of an economic perspective, software development being a labor-intensive process, there is a wide variance in labor rates (wages) from an international perspective.

Given the current state of knowledge on the topic, though, unit cost can be meaningful for the purpose of internal benchmarking (within the same organization) since knowledge of the three factors stated above would generally be easier to obtain if not already available.

Average calendar delivery rate

Average calendar delivery rate is defined as the average ratio of delivered functionality (measured in functional size units) per unit of calendar time (usually measured in months). It usually includes the amount of functionality turned over to the maintenance team and thus excludes the temporary functionality that might have been required during development (like IV&V "scaffolding" functionality, for instance). It is usually considered a reasonable measure of the "speed" of the development process, although it lacks the refinement necessary to evaluate some aspects of the development effort. For instance, it does not represent variations in the architectured, designed and programmed functionality which could be significant variables in the analysis of project efficiency when a high degree of requirement variation is encountered during development.

Average rates of delivery of the TACS project are presented in Table 3.5.3 below, in both FPA/calendar month and FFP/calendar month.

DELIVERY BASIS	SIZE	DURATION	RATE OF DEL.
FFP - Full Function Points	198 FFP	14 mth	14,1 FFP / Mth
FPA - Function Points	120 FPA	14 mth	8,6 FPA / Mth

Table 3.5.3 – TACS project average calendar delivery rate

4. RECOMMENDATIONS

In the light of the context described in section 1 and based on the results presented under section 1 and the analysis presented in section 3, this section sets out four specific recommendations formulated for the Land Software Engineering Centre (LSEC).

4.1 Parametric cost estimation models

The original motivation of the LSEC in pursuing the work underlying this report was to eventually replace the KSLOC software size measure used in its estimation model by a functional size measure. In order to afford a minimum of reliability to an estimation model using functional size, some historical data must be gathered.

It is recommended that data from a minimum of 15 historical projects be gathered before attempting to recalibrate an effort estimation equation based on a functional size measure. Successive refinements of the parameters of this equation could then be produced as more historical data becomes available.

4.2 Standardization of functional measurement

Given that:

- A significant portion of the portfolio of software applications maintained by the LSEC contains real-time or embedded software,
- FFP have been shown to offer a more precise measure of the functional size of real-time or embedded software,
- FFP have been shown to offer a better functional coverage of real-time or embedded software applications,
- FFP can be used to measure the functional size of both real-time and the MIS type of software applications,

it is recommended that Full Function Points be standardized as the method for measuring the functional size of software applications.

4.3 Count automation

Throughout the counting of the TACS application, it was observed that, based on the detailed specifications available and the level of standardization of these specifications (2167 military standard), it would be interesting to automate significant parts of the count by using tools already currently available in word processors.

It is therefore recommended that the development of simple tools that would at least assist in producing an experienced counter with a rough draft of the count be explored. Such tools would contribute to lowering the counting effort, while providing some means to standardize the identification of the elements to be counted.

Given the size of the portfolio of applications maintained by the LSEC, the impact of such tools on the cost of the counting effort is deemed to be non-negligible.

4.4 Training

Throughout the counting of the TACS application, it was observed that very few resources at the LSEC had the expertise to perform functional size counts using FPA on an autonomous basis. No resources had the expertise to perform FFP counts.

Although the expertise required to perform such counts can be bought from external suppliers, it is strongly recommended that an internal core of expertise on these techniques be established by having some LSEC staff trained as functional size counters. Taking recommendation 4.1 into account, the training of personnel for counting FFP should be organized first.

4.5 Further work

The analysis presented here sheds some light on two issues that are clearly outside the scope of this report, but are nonetheless pursued on an ongoing basis by the authors.

The first issue relates to the concept of the application boundary and its practical usage for real-time or embedded software. Any serious counting exercise of a portfolio of such applications should not be undertaken without careful consideration of this topic.

The second issue relates to the quantification of the functional re-use potential of software measured using the FFP technique and its economic implications for the project decision-making process and the portfolio management process.

APPENDICES

APPENDIX A – Detailed TACS FFP control process count

Seq.	Process ID	Process Description	Sub process type	Sub process Description	FFP
1	3.2.1	System initialisation	ECE	Message prompt	1
			ECE	Initialisation 5 sec.	1
			ECE	Control TACS app.	1
			ECX	Display copyright	1
			ECX	Error In 5-6-7	1
			ECX	Error IN 9-10	1
			ECX	Fire up IN-11	1
			ICR	IN 1 and 3	1
			ICR	IN 4	1
			ICR	IN-8	1
			ICW	Reconstruct TACS	1
2	3.2.2.1	Operator Interface	ECE	Modify date, time, conf.	1
			ECX	List OP 6-op 7	1
			ECX	Display status op 11-12-13-14	1
			ICR	Read date, time op 400	1
			ICR	Read conf. Menu op 1	1
			ICW	Write date, time	1
			ICW	Write conf. Menu	1
			ICW	OP 8 unknown	1
3	3.2.2.2	Menu Display	ECE	Cursor Position	1
			ECX	Display menu	1
4	3.2.2.3	Menu Input	ECX	OP 28, 30, 31	1
			ICR	Read menu file	1
5	3.2.2.4	Message Monitor	ECE	OP36	1
			ECX	OP38, 39, 40, 41	1
			ICR	OP37	1
6	3.2.2.5.1.1	Target manipulation	ECE	OP43, 44	1
		initialization	ECX	OP43, 44	1
7	3.2.2.5.1.2	Target verification	ECE	OP45	1
			ECE	OP54	1
			ECX	OP49, 46, 47	1
			ECX	OP52, 53	1
			ECX	OP56, 57, 58, 59, 60	1
			ICR	OP50	1
			ICR	OP51	1
			ICW	OP50	1
			ICW	OP51	1
8	3.2.2.5.1.3	Target elevation	ECE	OP62, 63	1
			ECX	OP62, 63	1
			ECX	OP61, 72	1
			ICR	OP64	1
			ICR	OP389, 390, 68	1
			ICW	OP389, 390, 68	1
			ICW	OP71	1
9	3.2.2.5.1.4	Radio link test	ECE	OP74	1
			ECE	OP76, 388	1
			ECX	OP379, 380, 381	1
			ECX	OP70, 83	1
			ICR	OP75	1
			ICW	OP79, 80, 81, 82	1
10	3.2.2.5.2.1	Sequence	ECE	OP84, 85	1
		manipulation	ECX	OP84, 85	1
11	3.2.2.5.2.2	Sequence creation	ECE	OP89	1
		modification	ECX	OP89	1
			ECX	OP413	1
			ECX	OP90, 92, 93	1
			ECX	OP95, 96, 97, 100	1
			ECX	OP98, 99, 115, 116	1
			ECX	OP102, 103, 104, 105	1
			ECX	OP117 to 121 incl.	1
			ECX	OP126, 127	1
			ECX	OP132, 133	1
			ICR	OP86, 87, 88	1
			ICR	OP86, 87, 88	1
			ICW	OP86, 87, 88	1
			ICW	OP86, 87, 88	1
12	3.2.2.5.2.4	Battlerun	ECE	OP 149, 178	1
		Creation / modif.	ECE	OP 150, 151, 175, 177	1
			ECX	OP149, 178	1
			ECX	OP150, 151, 153, 175, 177	1
			ECX	OP158	1
			ICR	OP 150, 151, 154	1
			ICR	OP149	1
			ICR	OP156, 175, 177, 181	1
			ICW	OP 150, 151, 154	1
			ICW	OP156, 175, 177, 181	1
13	3.2.2.5.2.5	Battlerun	ECE	OP 191, 212	1
		execution	ICR	OP 192, 198	1
			ICW	OP199	1
			ECX	OP188, 211	1
			ECX	OP191	1
			ECX	OP193, 194, 195, 196	1
			ECX	OP 198, 202, 214, 203, 215, 216, 204, 378, 396, 397, 398, 208, 211	1

Seq.	Process ID	Process Description	Sub process type	Sub process Description	FFP
14	3.2.2.5.2.6	Manual run	ECE	OP 220	1
		Execution	ECE	OP 231, 232	1
			ECE	OP 249	1
			ICR	OP 220, 235, 236, 239	1
			ICR	OP 226	1
			ICR	OP 250	1
			ICR	OP 404	1
			ICW	OP 253, 255, 256, 257, 258	1
			ICW	OP 403	1
			ECX	OP 261	1
			ECX	OP 260	1
			ECX	OP 220, 248	1
			ECX	OP 221, 222, 223, 224, 391	1
			ECX	OP 414	1
			ECX	OP 226, 227, 228	1
			ECX	OP 231, 232	1
			ECX	OP 241	1
			ECX	OP 243, 244, 392	1
			ECX	OP 404, 405	1
			ECX	OP 250, 251	1
			ECX	OP 254	1
15	3.2.2.5.2.7	Sequence	ECE	OP 262	1
		simulation	ECE	OP 273	1
			ECE	OP 277	1
			ICR	OP 262	1
			ICR	OP 268	1
			ICR	OP 277, 281	1
			ICW	OP 277, 281	1
			ECX	OP 262, 263, 264	1
			ECX	OP 268, 269, 270, 271	1
			ECX	OP 273	1
			ECX	OP 277, 278, 279, 280	1
			ECX	OP 282	1
			ECX	OP 285	1
16	3.2.2.5.3.1	Database	ECE	OP 282	1
		commands	ICR	OP 286	1
			ECX	OP 282	1
17	3.2.2.5.3.2	Copy	ECE	OP 287	1
		sequence	ECE	OP 290, 291, 292, 293	1
		definition	ICR	OP 287	1
			ICR	OP 288, 297, 298, 305	1
			ICW	OP 288, 297, 298, 305	1
			ECX	OP 287	1
			ECX	OP 288, 289	1
			ECX	OP 294, 295, 296, 297, 298, 299, 300, 301, 302, 303	1
			ECX	OP 306	1
18	3.2.2.5.3.3	Copy	ECE	OP 307, 310	1
		battlerun	ICR	OP 307	1
		definition	ICR	OP 308, 309, 324	1
			ICR	OP 317, 318, 325	1
			ICW	OP 308, 309, 324	1
			ICW	OP 317, 318, 325	1
			ECX	OP 307	1
			ECX	OP 308, 310, 314	1
			ECX	OP 315, 316	1
			ECX	OP 319, 320, 322	1
			ECX	OP 326	1
19	3.2.2.5.3.4	Delete	ECE	OP 327	1
		sequence	ICR	OP 327	1
		definition	ICW	OP 330	1
			ECX	OP 327, 386	1
			ECX	OP 331, 332	1
20	3.2.2.5.3.5	Delete	ECE	OP 333	1
		battlerun	ICR	OP 333, 334, 335	1
		definition	ICW	OP 333, 334, 335	1
			ICW	OP 332	1
			ECX	OP 333, 387	1
			ECX	OP 337, 338	1
21	3.2.2.5.3.6	Delete	ECE	OP 339	1
		hit data	ICR	OP 339	1
			ICW	OP 345	1
			ECX	OP 348, 341, 342, 385, 343, 344, 345	1
			ECX	OP 346, 347	1
22	3.2.2.5.4.1	Hit data	ECE	OP 348	1
		manipulation	ICR	OP 348	1
			ECX	OP 348	1
23	3.2.2.5.4.2	View hit data	ECE	OP 349	1
		results	ECE	???	1
			ICR	OP 349	1
			ICR	OP 352	1
			ICR	OP 355	1
			ECX	OP 349	1
			ECX	OP 350, 393, 351	1
			ECX	OP 352	1
			ECX	OP 355	1
			ECX	OP 359, 300, 361, 362, 364, 365, 366	1
24	3.2.2.5.4.3	Generate hit data	ECE	OP 367	1
		report	ECE	OP 368	1
			ICR	???	1
			ICR	OP 368	1
			ECX	OP 367, 394	1
			ECX	OP 368	1
			ECX	OP 371, 372, 373	1
25	3.2.2.5.5	Exit	ECE	OP 374	1
			ICR	OP 374	1
			ECX	OP 374, 375	1

APPENDIX B – Detailed TACS FPA elementary process count

Seq.	Process Number	Process Description	Identified Elementary processes and occurrences	FPA CNT.
1	3.2.1	System initialisation	EO (x 2)	8
2	3.2.2.1	Operator Interface	EI EQ	3 3
3	3.2.2.2	Menu Display	EQ	3
4	3.2.2.3	Menu Input	EQ	3
5	3.2.2.4	Message Monitor	See 3.2.2.2	
6	3.2.2.5.1.1	Target manipulation initialization	Navigation	
7	3.2.2.5.1.2	Target verification	EI (x 2) EQ	6 3
8	3.2.2.5.1.3	Target elevation	EO EI	4 3
9	3.2.2.5.1.4	Radio link test	EO EQ (See OP 63)	4
10	3.2.2.5.2.1	Sequence manipulation		
11	3.2.2.5.2.2	Sequence creation modification	EQ EO (OP 98, 99, 100) EI (x 2) Create, modify	3 4 6
12	3.2.2.5.2.4	Battlerun creation / modification	EI (x 2) EQ EO (x 2)	6 3 8
13	3.2.2.5.2.5	Battlerun execution	EO (OP188), See OP 178 EO (OP195), See OP 63 OP198 EO	 4
14	3.2.2.5.2.6	Manual run execution	EO (OP 221), See OP 63 EO (OP 232), See OP 198 EQ (OP 242), See OP 63 OP261, Navigational	
15	3.2.2.5.2.7	Sequence simulation	EO (OP168), See OP 63 EO (OP 273) EO (OP 278) EO (OP 285), See OP 187	 4 4
16	3.2.2.5.3.1	Database commands	Navigational	
17	3.2.2.5.3.2	Copy sequence definition	EO (OP 288) HHC EO (OP 294), See OP 288 EO (OP 295-296)	4 4
18	3.2.2.5.3.3	Copy battlerun definition	EO (OP 300), See OP 288 EO (OP 310), See OP 288 EO (OP 314), See OP 63 EO (OP 316), See OP 296	
19	3.2.2.5.3.4	Delete sequence definition	EO (OP 386), See OP 288	
20	3.2.2.5.3.5	Delete battlerun definition	EI (OP 387), See Seq. No. 12, DEL	3
21	3.2.2.5.3.6	Delete hit data	Already exist	
22	3.2.2.5.4.1	Hit data manipulation	EO (OP 348), HIT data report	4
23	3.2.2.5.4.2	View hit data results	See OP 348	
24	3.2.2.5.4.3	Generate hit data report	EO (OP 367), See OP 156	
25	3.2.2.5.5	Exit	Already exit	
		Error msg. Display	EO	4
			TOTAL	**101**