FUNCTION POINT ANALYSIS: AN APPLICATION TO A NUCLEAR REACTOR PROTECTION SYSTEM

Nihal Kececi
Reliability Engineering
University of Maryland
College Park, MD, 20742
nkececi@eng.umd.edu

Ming Li
Reliability Engineering
University of Maryland
College Park, MD, 20742
mli@wam.umd.edu

Carol Smidts
Reliability Engineering
University of Maryland
College Park, MD, 20742
csmidts@eng.umd.edu

ABSTRACT

This paper presents an application of full function point analysis to the estimation of the size of real-time control software. The full function point counting technique is briefly described. Its usage is illustrated on a part of the Westinghouse Reactor Protection Control System and the results analyzed. We further describe a technique for the graphical representation of requirements that helps in the full function point assessment. Specifically, the technique is used for identifying the groups of data and processes as well as the application boundaries.

I. INTRODUCTION

There is a direct relationship between the success of a product and the quality of management control. Cost and effort estimation is an important aspect of the management of software development projects. Most methods for estimating efforts require an estimate of the size of the software. Moreover, size of software can be an indirect indicator of software reliability. A study on the reliability of PLC devices performed in the United Kingdom[1] found a strong correlation between software failure rate and parameters related to the size and complexity of the software.

Several software size measures exist. However the two most popular are: (1) Physical size: a count of source lines of code, (2) functional size: a measure based on the software functionality delivered to the users.

Lines of code have been used widely throughout the software industry to estimate productivity and quality. However, many practitioners and researchers have argued that the line of code metric is essentially flawed. For instance, the higher the software language level, the less lines of code can be produced in a unit of time. Further caveats relate to the use of lines of code for early prediction, i.e. at the beginning of the software life cycle process. Since the number of lines of code is basically unknown the metric can not serve as size estimate.

Function Point Analysis (FPA) is one of the best known and most widely used methods for measuring functional size. FPA analysis will typically be performed at the end of the requirement specification phase. FPA is now widely used in the Management Information System (MIS) domain. FPA was designed and refined for business applications software. However, FPA has not enjoyed the same degree of acceptance in the domain of real-time software. As noted by Abran[2], difficulties arise when using FPA to measure non-business applications since FPA uses the amount of stored data as a significant factor in determining the functional size of the application. Where data stored is simple but the processing of stored values is complex, the functional size of the application is underestimated. "Several authors concur that when FPA is applied to real-time software, the results do not constitute an adequate size measurement[2]. Seven[3, 4, 5, 6] attempts to adapt FPA to real-time software have been identified in the literature". Full Function Point (FFP), developed by Abran and his colleague's,[7, 8] seems to be one of the most promising and constitutes the basis of this paper.

In general, function point counting is performed by a team of experts including an application expert, and an expert in function point counting. It should be noted that the understanding of the requirements is a crucial step in the counting process. Graphical representation

techniques can be a useful concept for the clarification/understanding/representation of requirements as well as for the identification of function count primitives such as inputs, outputs, boundaries, internal files, etc…

This paper presents a method for the graphical representation of software and system requirements. The technique will ease the translation of function point concepts into an equivalent prepositional graphical expression of requirements.

II. FPA OVERVIEW

The first step in performing a FPA is to identify the counting boundary, that is, the border between the application being measured, and the external applications, or the user domain. The next step consists of determining the Unadjusted Function Point (UFP) count, which reflects the specific countable functionality provided to the user by the application. Calculation of the UFP begins with the identification of function types of the application.

The final step consists in the assessment of a Value Adjustment Factor (VAF) which reflects the impact of general system characteristics on the function point count. These characteristics include distributed data processing, performance, heavily used configurations and other factors.

III. FFP OVERVIEW

Full Function Points (FFP) is an adaptation of FPA to the specific functional characteristics of real-time software. The Software Engineering Laboratory in Applied Metrics (SELAM) introduced the technique in 1997. More detailed information can be found in the Full Function Point Counting Practices Manual [7]. An overview of UFP counting by using this approach is summarized in Figure 1. After identify the counting boundary, FFP analysis includes the following steps:

(1) Identify group of data. These groups are defined as follows. (a) *Management data*: same as FPA. (b) *Control data*: used by the application to control, directly or indirectly, the behavior of an application or of a mechanical device. For a control group of data, the following procedures and rules should be applied. Control is further divided into Updated Control Group data and Read Control Group data. These terms are defined

as follows: (a) Updated Control Group: A UCG is a group of control data updated by the application being counted. (b) Read-only Control Group: An RCG is a group of control data used, but not updated, by the application being counted. The point assignment differs whether single or multiple occurrences of data are concerned.

(2) Identify process. (a) *Management process*: Same as FPA. (b) *Control process*: The FFP procedures and rules should be applied. We first distinguish between the concepts of External Control Entry, Internal Control Write, External Control Exit and Internal Control Read.

The definitions follow: *An External Control Entry*: processes control data coming from outside the application's boundary. It is the lowest level of decomposition process acting on one group data. *An Internal Control Write*: is a unique sub-process. The ICW writes control data, and is the lowest level of decomposition of process acting on one group of data. *External Control Exit*: An ECX is a unique sub-process. It is identified from a functional perspective. The ECX process control data goes outside the application's boundary, and is lowest level of decomposition of process acting on one group data. *Internal Control Read*: An ICR is a unique sub-process. It is identified from a functional perspective. The ICR reads control data, and is the lowest level of decomposition of process acting on one group data.

IV. REACTOR PROTECTION SYSTEM AND SOFTWARE REQUIREMENTS

The goal of the reactor protection system is to initiate a reactor trip if the safe operating limits are exceeded and to initiate engineered safety features if an accident occurs. Sensors monitor the values of specific variables and compare them to their respective setpoints[9]. The following are the variables required to be monitored in order to provide reactor trips:

1. Neutron flux
2. Reactor coolant temperature
3. Reactor coolant system pressure (pressurizer pressure)
4. Pressurizer water level
5. Reactor coolant flow
6. Reactor coolant pump operational status (bus voltage and frequency, and breaker position)
7. Steam generator feed-water flow
8. Steam generator water level

9. Turbine-generator operational status (auto-stop oil pressure and stop valve position)

The Operational I&C system is the interface between man and process. To provide the information for the operator the actual parameters of the process have to be prepared by the I&C system.

The Generic Westinghouse Reactor Protection System Software requirements consist of 101-requirement[10]. We illustrate this process on a small section of the Westinghouse reactor protection system. Our analysis focused on a subset of requirements that related to the pressurizer water level control process.

V. NEED FOR A GRAPHICAL REPRESENTATION

As can be seen from the descriptions in sections III and IV both counting rules and process to be counted are rather complex. The process complexity resides in the number of variables involved. The complexity of the counting rules resides in the language used, in its interpretation as well as in the bookkeeping that goes along the entire analysis. The claim made in this paper is that a graphical representation of the system under study will provide a means of communication between designers of systems and function point analysts. A further advantage of a well-behaved graphical representation is that it may be amenable to an automatic count of the application. This then reduces the potential for error in the counting process.

To construct a graphical function point representation (GFPR) of the system under study, we follow the sequence of steps described below.

STEP 1. Identification of the application boundary. The system requirements of the application under count should be carefully reviewed to identify which part of the system shall be implemented by software, which part by hardware and which part is in the hands of operators. We are only interested in the software implementation and in the requirements that define this software implementation. Hardware and humans constitute interfaces that either feed data into the software or receive data from the software. At the end this step whole software requirements have been identified and the interfaces with hardware and operators are known.

STEP 2. Functional decomposition. In order to simplify the analysis, the functional purpose of the application is decomposed into functions. Each function will then be represented graphically and the graphics later recombined to yield the complete software functionality. The lowest level of decomposition is the elementary process meaningful to the end-user. At the end of this step, a list of software functions is available.

STEP 3. Requirements collection for the functions(sub-processes). All requirements involved in a function are identified. Given the output generated by a function, all inputs to the function are traced back through other functions either to the boundary of the software application or till one can not go any further. At the end of this step a list of all requirements involved in the computation of this function is available.

STEP 4. External Input/Output and Boundary Representation. The external inputs of the requirements involved in the computation of the function of interest are identified and placed on the left side of the diagram presented in Figure 2, and the external outputs (if any) are placed on the top of the diagram. The dashed rectangle in the diagram represents the boundary of the application under study.

STEP 5. Explicit representation of relationships between requirements involved in the computation of the function. STEP 3 identified requirements that were related. Two requirements A and B are related iff the input of B is an output of A. Figure 3 shows how the relationships are expressed graphically. The diagram should be read from left to right and from bottom to top. Indeed, requirements on the left support (i.e. provide input data to) requirements on the right. Lines between requirements indicate the existence of a data flow or control information flow between the requirements. Dots in the diagram represent reading relationships between requirements where a dot between requirements A and B means that requirement A writes an output which B will read. The requirements mentioned in this step can either be a requirements per say or a group of functionally related requirements.

STEP 6. Establishment of the GFPR. We obtain the final graphical representation by refining Figure 3. Each requirement or requirement group is decomposed to the level of elementary process. A process that creates an output that remains within the boundaries of the

application is represented by a single line rectangle box followed by a circle containing the number of DETs written. The default number of DETs is 1 and in this case the circle remains empty. Each elementary process that creates an external output is represented by a single lined rectangle. Read only data retrieved by an elementary process is represented by a single line rectangle that has no entry. External inputs are placed in a single line rectangle box on the left side of the application boundary. An arrow crossing the top of the application boundary signifies the presence of an external output. Double lined rectangle boxes correspond to multiple occurrences groups of data. They symbolize existence of a logical association between data. For example, a requirement such as "The system generates a log containing both the trip signal and the corresponding timestamp" links the trip signal to the timestamp logically. The word logically means that the majority of further decisions and/or computations in the application concern the group and not its individual elements. The group is considered as a single unit. The graphical representation of this group of data is a double lined rectangle.

STEP 7. Integration of all GFPRs. This step consists in integrating the GFPRs constructed for each function. This step will not be studied. We will assume here that the software application consists of a single function.

STEP 8. Counting. Rules for the automatic counting of the application are given in the next section.

VI. AUTOMATIC COUNT

Once the GFPR is established, the full function point can be performed automatically. The following paragraph discusses the rules used to count the application based upon the GFPR.

Identification and Count of UCGs. Every elementary process generates data that belongs to an UCG. External output data is an exception to this rule. It does not belong to the UCG. An application can only have one Single Occurrence UCG. The number of DETs in this UCG is the total number of DETs in circles minus the total number of DETs that belong to multiple occurrence data groups. The number of double lined rectangles is the number of Multiple Occurrences UCGs. The number of DETs in each multiple occurrences UCG is the total number of

DETs in the circles related to elementary processes that write into this UCG.

Identification of RCG. The same counting rules apply to the RCG. However, the data considered is read-only.

Identification of ICWs and ICRs. Each circle symbol increases the number of ICWs by one. Each dot increases the number of ICRs by one.

Determination of DETs in the ICRs. The number of branches placed just before a dot represents the number of DETs that dot (ICR) has.

Identification of ECEs. Each line crossing the leftmost boundary of the diagram corresponds to an ECE.

Identification of ECXs. Each line crossing the topmost boundary of the application is an ECX.

VII. GWRPS GFPR AND FUNCTION POINT COUNT

This section provides the results of the application of the procedure outlined in section VI to the GWRPS Pressurizer Water Level trip.

The requirements involved in the computation of this trip function are as follows:

Pressurizer Water Level Block
Inputs
- Press-Water-Level
- B-Press-Water-Level
- C-Press-Water-Level
- High –Press Water-Level Set-Point
- P-7
- High-Press-Water-Level-Trip

Requirement 35: The pressurizer water level signal (A- through C Press-Level) and pressurizer water level set point (High-Press-Level-Set-point) are given in percent total water level. All pressurizer water levels and set points are displayed as percents. The available space for displaying the water levels and set points is 3 digits.

Requirement 37: A circle indicator labeled "P-7" is gray when P-7 is FALSE and green when it is TRUE.

Requirement 38: A rectangle indicator labeled "TRIP" is gray when is FALSE and green when it is TRUE.

Requirement 79: Pressurizer High Water Level Trip
Inputs:
- A- Press-Level
- B-Press-Level
- C-Press-Level
- High –Press Level Set-Point
- P-7

Output**:** High-Press-Level-Trip
Logic: The logic for the trip is the following (X=A, B, C). IF (2 out of 3 x Press-Level> High Press-Level Set-point) AND (P-7= FALSE) THEN

   High -Press- Level-Trip = FALSE;
ELSE

   High -Press- Level-Trip = TRUE;
    END.

Requirement 87 (Permissive P-7)**:** The actual logic for permissive P-7 involves at least one of the first stage turbine signals to be above a set point. In the GWRPS, it is assumed that both first stage turbine pressures are above the set point and

omitted from the logic. Therefore, the logic for permissive P-7 is the inverse of P-10.
  P-7=NOT (P-10)

Requirements 86 (Permissive P-10):
Inputs:
1. A-Power-Range Flux
2. B-Power-Range Flux
3. C-Power-Range Flux
4. C-Power-Range Flux

Output: P-10
Logic: The following permissive logic for P-10 (X=A, B, C, D)

  IF (2 out of 4X Power-Range-Flux >10% Nuclear Power) THEN
   P-10=TRUE
  ELSE
   P-10=FALSE;
  END.

Requirement 100: The data generated by the GWRPS is placed into global variables. The data is then accessed by the monitor for display.

Figure 4 gives the GFPR for this specific function. The corresponding full function point count is given in Table 1.
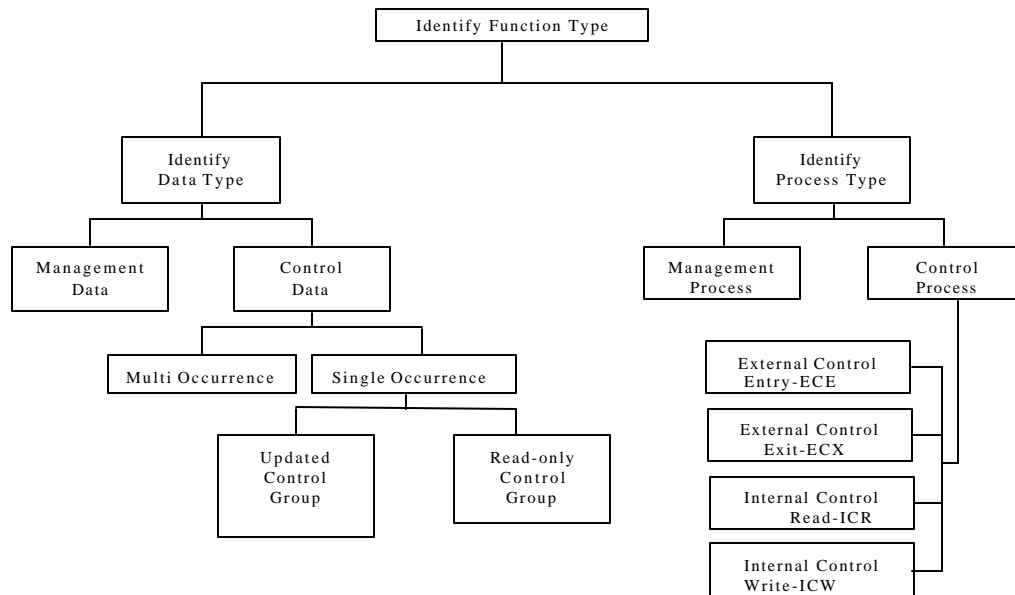

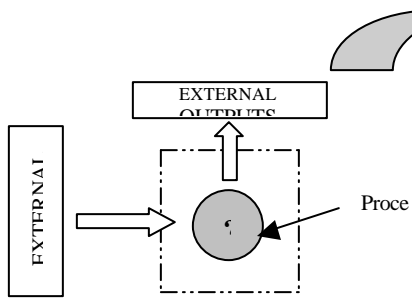Figure 1 Overview of Full Function Point
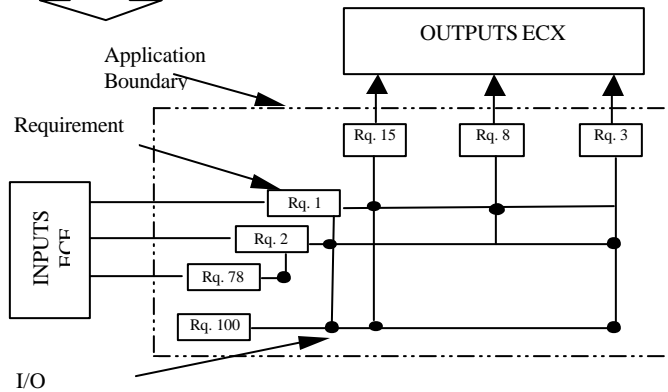
Figure 2: Identification of process

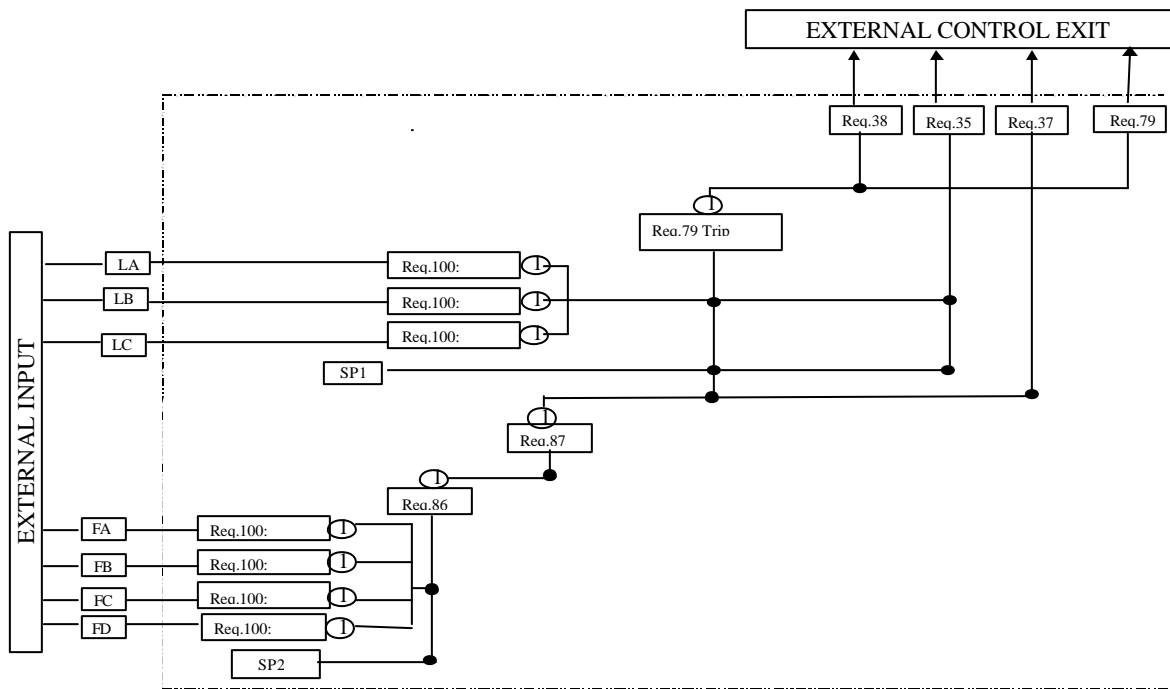Figure 3. Classification of ECE, ECX, ICW, ICR

Figure 4. Application of GFPR on Pressurizer Water Level Process Software

VIII CONCLUSIONS

This paper presents a graphical function point representation and the corresponding counting procedure for estimating the size of real –time system software. The method is based on the full function point methodology and provides a means of communication between system designer and function point counting experts.

The counting process can be easily automated once the graphical representation is available. Future research will be directed towards the automation of the entire process, i.e. from requirements to final count.

Data Function Types			
Type	Description	DET	Point
Single Occurrence UCG	Water Level A	10	7
	Water Level B		
	Water Level C		
	Flux Level A		
	Flux Level B		
	Flux Level C		
	Flux Level D		
	P7		
	P10		
	Trip Signal		
Single Occurrence RCG	Water Level Set Point	2	0.4
	Flux Level Set Point		
Sub Total			7.4
Transaction Function Types			
Type	Description	DET	Point
ECE	Water Level A	1	1
	Water Level B	1	1
	Water Level C	1	1
	Flux Level A	1	1
	Flux Level B	1	1
	Flux Level C	1	1
	Flux Level D	1	1
ECX	Trip Signal	1	1
	Display Water Level (Rq. 35)	3	1
	Display P7 Indicator (Rq. 37)	1	1
	Display Trip Indicator (Rq. 38)	1	1
ICR	Rq 35 reads Water Level Set Point	1	1
	Rq 35 reads Water Levels	3	1
	Rq 37 reads P7	1	1
	Rq 38 reads Trip	1	1
	Rq. 79 reads P7	1	1
	Rq. 79 reads Water Levels	3	1
	Rq. 79 reads Water Level Set Point	1	1
	Rq. 86 reads Flux Set Point	1	1
	Rq. 86 reads Flux Levels	4	1
	Rq. 87 reads P10	1	1
ICW	System writes Water Level A	1	1
	System writes Water Level B	1	1
	System writes Water Level C	1	1
	System writes Flux Level A	1	1
	System writes Flux Level B	1	1
	System writes Flux Level C	1	1
	System writes Flux Level D	1	1
	Rq. 79 writes Trip	1	1
	Rq. 86 writes P10	1	1
	Rq. 87 writes P7	1	1
Sub Total			31
Total			38.4

Table 1 Full Function Point Count of GWRPS

ACKNOWLEDGEMENT

REFERENCES

1. R.M. Consultants Ltd., *An Investigation into PLC Software Reliability,* A report prepared for the Health and Safety Executive, London, UK., November 1995.

2. Abran, A., Maya, M., St-Pierre D., and Desharnais, J-M, *Adapting Function Points to Real Time Software,* Universite du Quebec a Montreal, November 1997.

3. Galea, S., *The Boeing Company: 3D Function Point Extensions, V2.0, Release 1.0,* Seattle, WA: Boeing Information and Support Services, Research and Technology Software Engineering, June 1995.

4. Jones, C., Applied *Software Measurement-* Assuring Productivity and Quality, McGraw-Hill, New York, 1991, pp. 493

5. Reifer, D.J., Asset-R: *A Function Point Sizing Tool for Scientific and Real-Time Systems.* Journal of Systems and Software, Vol.11, No.3, March 1990, pp. 159-171

6. Whitmire, S.A., *3-D Function Points: Scientific and Real Time Extensions to Function Points,* Proceeding of the 1992 Pacific Northwest Software Quality Conference, Portland, OR, 1992

7. Desharnais, J-M., St-Pierre D., Maya, M., Abran, A. *Full Function Points: Counting Practices Manual Procedure and Counting Rules.* Universite du Quebec a Montreal, November 1997.

8. Serge Oligny, Alain Abran, Jean-Marc Desharnais, Pam Morris. *Functional* Size Of Real Time Software: Overview of Field Test. UQAM Software Engineering Management Research Lab.1998

9. Westinghouse Technology System Manuel United States Nuclear Regulatory Commission Technical Training Center.Rev.0690 vol.2, pp12.

10. Generic Westinghouse Reactor Protection Control System Software Requirements.