Using ‘USDP Functional Model’ in Function Point Analysis

Iwona Dubielewicz
Informatics Department, Technical University of Wrocław, Wrocław, Poland

Bogumiła Hnatkowska
Informatics Department, Technical University of Wrocław, Wrocław, Poland

Abstract

The paper presents the steps of function point (FP) counting for information system model, constructed according to USDP recommendations. At the beginning the short characteristics of function point counting (using Albrecht’s and MKII FP analysis) is presented and afterwards the brief USDP presentation is given. The FP counting case study is performed for the USDP “functional model” of the simple computerized system for electrical kiln of clay pots firing.

1. Introduction

More than 70 % of the software projects are over cost and/or time planed for them. Such a situation need changing so one can easy understood demands for enhance the methods used to estimate more precisely these parameters.

These demands has been mapped into requirements for more accurate software complexity sizing as man fill that software complexity strongly influences on the project parameters mentioned above. In 1979, when A. Albrecht proposed the Function Point (FP) as the measure for sizing software, it was not obvious that his approach, called FP analysis, would get so wide acceptance as today it has. It has happened because FP technique quantifies the functions (contained within software) from the user point of view and it is easy to use as it considers only data structures which system functions are working on. Moreover, FP measure is recommended for use in others business measures e.g. productivity and effort can be derived when the former is known.

The function points analysis is independent of the technology used for developing any application. Additionally, it has been recognized that function points counting can be done early in the software life cycle, what is very fruitful because the sooner the project can be quantify the sooner is under control. Actually, it is recommended to count these points once the requirements have been finalized.

Up today the FP analysis (FPA) has been applied mainly to the business system, developed with structured approaches. But now, when object oriented approach becomes more and more popular and many o-o methodologies has been arisen, the way, how to conduct the FP analysis in that new reality should be proposed.

The main aim of this paper is to show, how function point approach can be used to size software functionality during the modeling stage in the USDP.

In the first part of the paper the short presentation of FP techniques and Unified Software Development Process (USDP) [5] methodology is done. The notion of functional model is described, and its interpretation and construction in USDP environment is given. The process of creating functional model according to USDP guidance is illustrated by simple example of "soft" real- time type. computerized system.

The second part contains the case study. This study presents applying described sizing methods for a system, using standard (Albrecht's) FP counting and MKII FP counting. Additionally, possibility of function sizing in the manner proposed by COSMIConsortium [7] is briefly discussed.

In the conclusion the advantages and drawbacks of the presented counting are discussed and some remarks dealing with sizing the system during the designing process are given.

2. FP analysis characteristic.

The FP analysis uses the algorithmic model based on counting so called 'function points'.

The starting point of construction of this model in Albrecht's approach, is to determine a number of items (also named 'functions') occurring in the system:

Data function (types)

External files (EIF) – deal with all machine readable interfaces to other systems

Internal files (ILF) – are the master (logical) files in the system

Transaction function (types)

External inputs (EI) – are the inputs from the user that provide distinct application oriented data; they change values of system data

External outputs (EO) – the data directed to the user (messages, reports etc.)

User inquiries (EQ) – interactive inputs requiring the response; they do not change any data in the system

Additionally, to accommodate the various level of complexity of each function it has been introduced the subjective complexity three-level rating: simple, average, complex; these levels are mapped on numerical scale as is presented in table 1.

Table 1. Rating for function of FP analysis

Function	**Simple**	**Average**	**Complex**
External interface files (EIF)	7	10	15
Internal logical files (ILF)	5	7	10
External inputs (EI)	3	4	6
External outputs (EO)	4	5	7
User (external) inquiries (EQ)	3	4	6

The original Albrecht's approach – called Unadjusted Function Count (UFC) – is treated as a size measure of software functionality and is determined as:

$$UFC = \Sigma\, function_i * w_i \qquad (1)$$

where w_i is the score taken from the Table 1 for each function (after fixing the level rate for the function).

To fix the level rate the other tables [3] should be used. These tables are various for each function category and give the rate which depends on the numbers of data (element) types (DET) and record types (REC) – for ILF, EIF– or on the numbers of record types and file referenced (RF) (for EI, EO, EQ).

Albrecht proposed also to count the (adjusted) Function Points FP, defined as FP= UFC *TFC, where TFC is the so-called technical complexity factor. This metrics considers the nonfunctional user requirements (its computation is done using the experimentally derived formula which the 14 technical factors contributing to the overall notion of complexity). This counting is often criticized as being very subjective and the same opinion is presented in ISO 14143.

Another approach, which has gained much popularity in counting function points was proposed by Symones, 1991 [2]. The functional size of the application is defined as the sum of sizes for each Logical Transaction understood as the lowest level of self-consistent process. Functional size of a Logical Transaction is the weighted sum of its three components i.e. input, processing and output, and the formula used for function point counting is as follows:

$$\text{FP Index} = wo * \text{OutDET} + wi * \text{InDET} + we * \text{EnR}, \tag{2}$$

where *wo, wi, we* are the industry-calibrated weights equal to 0.58 (for per Input Data Element Type - InDET), 1.66 (for Entity Type Referenced -EnR), 0.26 (for Output Data Element Type – OutDET).

Both, Albrecht's as well as Symones propositions are often criticized because they do not accommodate other features of software e.g. complexity of algorithms, processing of control signals etc. To pass through these drawbacks some practitioners have proposed the other measures (feature points, 3D, etc.), but they have resulted in inconsistencies among various sizing methods (N.B. none of them has gained much popularity).

To overcome these different interpretations of the original idea, the new concept called Functional Size Measurement (FSM) was introduced and some fundamentals of it has been defined in ISO 14143 [4]. This concept should be treated as some kind of generalization of the approaches used before. The ISO standard introduces the notion Base Function Component which is understood as elementary unit of functional user requirements defined by and used by an FSM method. The newest proposal of Functional Size measurement based on this conception has been formulated by COSMIC team [7]. They have proposed to measure functional size at the level of „Data Movement Type (DMT)"and this size expressed in DMT is as follows:

$$\text{DMT} = N * (\text{no. DET's})$$

where N=1 and no. DET's is the number of unique Data Element Types on Data Entry or Read, which have to be processed, or on Data Exit or Write – of the obtained results. This counting is done for each transaction Identification of the logical movement types as Entries, Exits (in/from process), Read and Write (from/ to data files) is relatively simple and unambiguous in functional user requirements.

As one can easy notice all described methods have used (maybe in some subtle different meaning) three basic notions in the sense like these:

(logical) transaction – describes user interaction with a system and allows user to input data or control information, which change data stores in the system, or retrieve data stored inside the system or send control information to the outside world

logical files / entities – represent a data stores the contents of which can be access using the proper transactions (store or change or retrieve)

data element type – uniquely processed item of information that is indivisible for the purpose of transaction

3. Methodology characteristic

Unified System Development Process (USDP) is a methodology recommended for the object-oriented approach to the information system development. It has been elaborated as the response to the demands which could be used for the specification and designing of the software-intensive systems. It has been defined by three world-known men Rambaugh, Jacobson and Booch, so it is based on concepts, ideas and notations used in their very popular object-oriented methodologies: OMT, OOSE and Booch.

USDP uses 7 kinds of models to describe modeled system from different point of view and at a given level of abstraction, i.e. use-case model, analysis model, design model, deployment model, component model, implementation model and test model.

The first two models are the most important as they constitute (form) the ‘functional environment’ unnecessary for counting the function points. The use-case model contains actors and use-cases. It stands external view of a modeled system. The analysis model is a “conceptual realization” of the use-case model and it is described in terms of collaboration groups. Each collaboration group is presented by set of interaction diagrams. The other models are out of our interests as they are the results of later phases of software development.

Analysis model and use-case model include all the elements required for functional sizing, i.e. transaction description, and entity classes with domain attributes, so it seems that these two models are enough detailed to stand the basis for ‘first-cut’ estimation of application functional size. These two models are assumed to be the “USDP functional model”.

4. Counting function points in USDP – case study

In the USDP the modeling process begins with the requirements specification expressed by the use case model (based on the problem description in natural language as presented below) [5]. This specification is often accompanied with the vocabulary containing the definitions of the main notions and objects encountered in the problem (part of which is presented in domain model).

4.1. Problem specification

It is needed the computerized system which will support the managing of an electrical kiln for clay pots firing. The kiln consists of n heaters, ventilator and automatically moved conveyor belt on which pots are set. The temperature of the kiln is controlled by a programmable regulator and read by k temperature sensors. When the kiln starts or ends the heating, belt is pull by an electrical engine [1].

There are several types of firing, each with a fixed duration and an expected temperature profile. During a firing the temperature of the kiln is automatically recorded at regular intervals and the temperature readings are displayed graphically. The temperature readings associated with earlier firing may be displayed at any time. During a firing, if the actual temperature differs from the expected temperature by more than a pre-set amount an audible alarm will be sounded (by the regulator) to alert the operator.

The system may be used to pre-set a program of future firings. The schedule of the firings may be run or stopped at any time (a current firing must be finished). Administrator is a special kind of the operators and can modify firing types, test the kiln and the engine.

So Functional model in USDP from the perspective of Operator user has the features, presented in Fig. 1 and Fig. 2. The functions the operator can perform are Schedule Firing and Schedule Management (Fig.2).

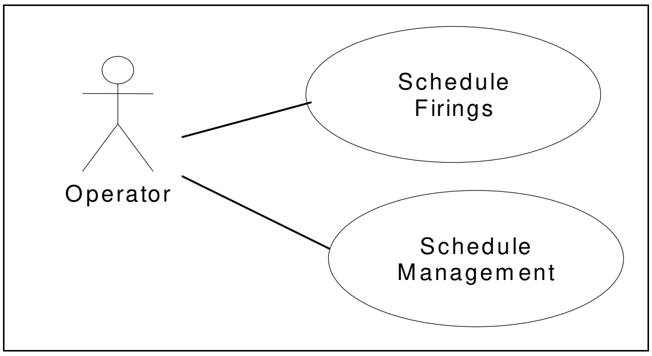

Figure 1: Fragment of a use-case model

For FP counting we have chosen the first, Schedule Firing, use case because its operations are dominated by storage and retrieval of data.

Figure 3 with a class diagram shows static aspect of a system.

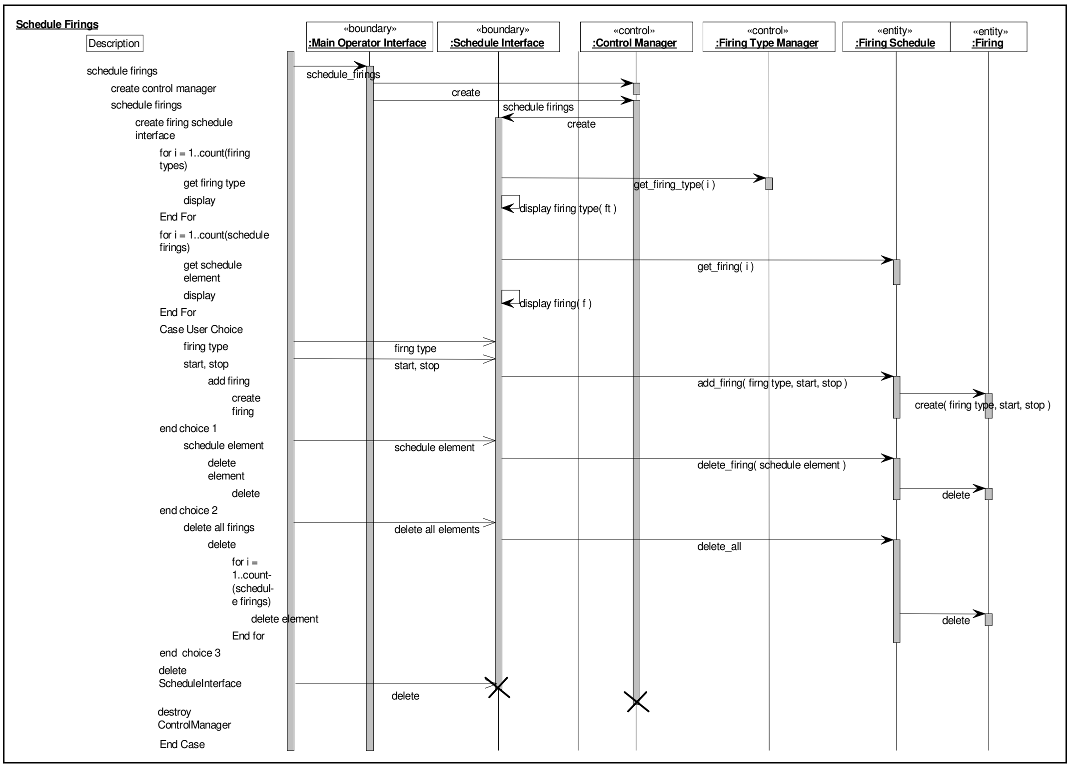

Figure 2: Sequence diagram for Schedule Firings use case

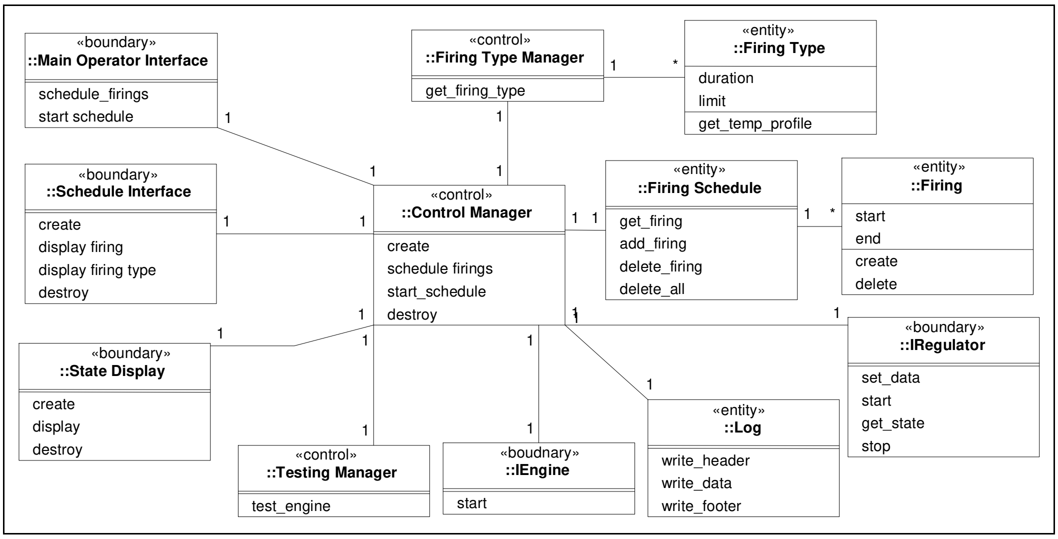

Figure 3: Operator view class diagram

4.2. FP Counting

FP counting has different stages depending on the chosen approach. But to apply any of them, the transactions (business functions) should be identified

In the UML a use case is a specialization of classifier, so it can provide a set of operations. Analyzing scenario (Fig.3) for the *Schedule Firings* one can recognize that it consist*s* of the three operations: *add firing, delete firing, delete all firings*. They can be treated as transactions.

But these transactions can have some internal structure, inner operations, and some of them can also be treated as transactions:

add a firing

+ display firing type (*External Output*)

+ display data of firings schedule (*External Output*)

+ choose firing type (*External Inquiry*)

+ input stop and end time for the new firing (*External Input*)

+ create and schedule a new firing (*External Input*)

delete a given firing

+ display firing type (*External Output*)

+ display data of firings schedule (*External Output*)

+ choose a firing (*External Inquiry*)

+ delete a firing (*Internal Logical File*)

delete all firings

+ display firing type (*External Output*)

+ display data of firings schedule (*External Output*)

+ delete all previously scheduled firings (*Internal Logical File*)

4.2.1 Albrecht's approach to size software

To obtain FP index in Albrecht's approach the following activities are recommended:

1. *Files (ILF and EIF) identification* – using USDP analysis model it is enough to investigate class diagram. The classes with stereotype <<entity>> pretend to be Internal Logical Files, and some of those with <<boundary>>

stereotype can be classified as External Interface Files. For our example ILF=2 (Firing Type and Firing) and EIF=2 (Regulator, Engine).

2. *Function (transaction- EQ, EI, EO) identification* – in USDP analysis models they can be obtained by investigation of sequence diagrams for each use case; for our example the identified functions and its categories are shown in 4.2.and there are 2 inquiry functions, two input functions and 5 output functions.

3. *Counting UFP* - Functional Points can be counted in two ways.

The first, rough, is as follows:

complexity rate is assumed for system as the whole – it means that one complexity rate is assigned to all system functions – and the system complexity is counted by multiplying value (taking from Table 1) by the number of the function of each category; for use cases described in 4.1, when the modeled system is treated as that of the average complexity, UFP is equal to:

UFP = 4* (EQ=2) + 4*(EI=2) + 5*(EO=2) +10*(EIF=0) +7*(ILF=2) = **40**

Such counting is done when no details concerning system structure and its data are given.

The second, more precise way of counting follows the standard procedure. For each system function FP is determined independently. For each function the data element types and record types it uses, and the files it referenced to are identified. When using USDP this identification is easy for performing due to the class attributes (DET), messages/operation parameters (DET, REC), and classes (REC) – elements defined in analysis model. Level rate, presented for each function in [3] allows for determination its complexity (low, medium, high). Obtained complexity is then transformed in scores given in table 1.

Afterwards obtained results are summed – according to formula (1); for our example:

UFP = 3* (EQ=2) +4*(EO=2) +5*(EI=4) +2*(ILF=2) = 48

The calculation done in that way seems to be more precise then this resulting from totaling the elements. The latter approach requires much more detailed description of system models but it can be quite easy done when using USDP.

4.2.2 MKII FP approach to size software

To obtain FP index in Symone's approach the following steps are needed:

1. *Determination of counting boundary*. In USDP the counting boundary is very easy for determining (classes with stereotype <<boundary>> define it).

2. *Identification of logical transaction* (defined as lowest level of self-consistent process). In USDP analysis model all messages send to and from system boundary classes, or from an actor or other inner classes, pretend to be classified as logical transaction (it is shown in 4.2). It is worth to notice that two ‘camouflage’ transaction has been recovered (‘display firing’ and ‘display firing type’) due to precise description of the use-case realization (in sequence diagram form). Each transaction is driven by event (transaction of E type), or by user inquiry (transaction of Q type).
3. *Calculation for each transaction* its input and output DET’s and Entity types it references to. Entity Types can be determined after analyzing the class diagram. Classes with stereotype <<entity>> are candidates for them. It can be also easy to determine the ‘non-primary’ (system) entities as they have few attributes changing very rare (in the example Log class is of this category).
4. *Calculation for each identified transaction* its FP index – according to formula (2).

The results obtained for the Schedule Firings use case size are presented in Table 2 – the numbers of Input and Output DET’s (placed in the table) are derived from the operation signatures specified in class description.

Table 2. Calculation of FP index for Schedule Firings use case

Transaction name	**Transa-ction type**	**No of Input DET**	**Response**	**No of Output DET**	**Entity Referenced to**	**No of ER**	**MKII FP**
Add Firing	E	4	OK./Error	2	FiringType, Firing	2	6.42
Delete Firing	E	1	OK./Message	2	FiringType, Firing	2	4.68
Delete all Firings	E	1	OK./Message	2	Firing	1	3.18
Display Firing Type	Q	1	FiringType/ Message	4	FiringType	1	3.8
Display Firing	Q	1	Firing/Message	5	Firing	1	4.32
Display all Firings	Q	1	List/Message	7	Firing	2	7.54

5. Conclusion

The function point technique provides an objective, comparative measure which can be effectively used in evaluation, and management (e.g. planning, control) of software production. But function point measure is effected by some degree of subjectivity. To reduce this influence one has to employ the detailed counting rules and applies it in coherent manner.

FP analysis is widely accepted technique. The paper presents the calculation of this size measure following two different approaches, i.e. Albrecht's and Symones. The third calculation proposed by COSMIC is not presented in the paper, but it seems to be possible to do it when using USDP. This technique may gain much popularity. The work on this conception is still in progress, and no measuring data are known. Therefore no comparison and discussion of the obtained results can be done.

Using USDP models in the way presented in section 4, function point measure sizes only the core of application (business logic). The FP calculation is independent of environment in which application will be situated as well as of technologies used to produce this software (e.g. CASE tools). It is also independent of the type of HC interface connected to the resulting software. FP can be easy calculated for any additional functionality and changes introduced to application.

References

[1]. Dubielewicz I., Hnatkowska B., Applying UML to information systems modeling, Conf. Proceedings MOSIS'99, Czech Republic.

[2]. The Functional Model and Its Role in Function Point Analysis, www.charismatek.com.

[3]. IFPUG , Function Point Counting Practices Manual v.4.1, www.ifpug.com

[4]. ISO/IEC DIS 14143-1:1996, Information Technology - Software Measurement - Part 1: Definition of functional size measurement

[5]. Jacobson I.,. Booch G, Rambaugh J.. Unified Software Development Process, Addison-Wesley, 1999.

[6]. MKII Function Point Analysis, Counting Practices Manual v.1.3.1, September 1998 UKSMA Metrics Practices Committee

[7]. Symones Ch., Rule P., One size fits all "COSMIC' – Aims, Design Principles and Progress', www.cosmicon.com