Journal of
Information Technology & Software Engineering

Research Article Open Access

Function Point Analysis FPA on A Team Planning Website Based on PHP and MYSQL

Lassen A*

Department of Computer Science, University of Copenhagen, Copenhagen, Denmark

Abstract

A function point analysis (FPA) has been carried out on a custom team planning website based on PHP and MYSQL. The FPA was done after the development was finished (retrospective) and a constructive cost model analysis (COCOMO) was carried out to asses source lines of code (SLOC). In the present study the function point analysis is based on entities of the relational database evaluated as internal logical files, and evaluation of PHP source code with dynamic SQL as either external input files or external inquiry files. The complexity for a custom team planning website was found to be 510 total adjusted function points (TAFP) (UAF=580 FP, TDI=23, VAF=0, 88, TAFP=510FP). The function point estimate was classified as an organic project in a COCOMO analysis, and it was concluded that the complexity corresponds to 27742 LOC (or 27, 4 KDSI). The estimate of 66-82 person-months project would correspond to a 4 crew team in 16-21 months. The estimate was compared to the actual source code count of 22300 LOC.

Keywords: Function point analysis; Dynamic SQL

Introduction

Function point analysis (FPA) is one way to determine the overall complexity of a system. In the current study a custom website is analyzed for complexity. Function point analysis is attributed to Allan Albright in 1979 [1] and JE Gaffney [2] and further developed in the MK2 report [3]. More recent work on function point analysis, a software tool (Unified code count (UCC)) [4,5]. Function point analysis can be evaluated using UML [6,7]. Made a literature review based on reported keywords identifying improvements to the accuracy of function point analysis [8]. They included 18 primary studies. The improvements were categorized into three categories: 1) "weights and complexities"; 2) 'technological in-dependence" of the method; and 3) calculating the 'adjusted functional size". Literature review for productivity [9]. A study of Henderson and coworkers study perception of function point analysis from a manger viewpoint and a developer viewpoint based 13 desirable properties with 3 key findings: SLOC-count is less complicated than FP; developers better perceive the benefits of FP than Managers; the difference in values between managers and developers inhibit communication necessary to reach informed decisions [10].

The FPA Allows for quantifying different properties of the system, in LOW, AVERAGE or HIGH complexity, totaling the unadjusted function points (UAF). Use SIMPLE, AVERAGE and COMPLEX, where SIMPLE and COMPLEX are well defined; and use an AVERAGE, MEDIAN, RANGE(LOW, HIGH) classification for complexity [2]. The unadjusted function points can then be adjusted for technical complexity as the total adjusted function points (TAFP). It is this measure that can be converted to project size in terms of man years based on lines of code (LOC) for the used programming language [11]. A function point measure for a list of languages. To assess PHP we use the LOC per function point for java and C++ [12].

The system examined is custom build website supporting planning tasks and in-site-postings for Danish yachtracing crews participating in international match race. The website domain myteam.dk was built and in operation in the years 2008-2014 by Hans Jacob Simonsen [13]. The system supports in-site blogging, planning, logging comments to training and events, handling expenses, sending out reminders by SMS.

The result of the function point analysis is further analyzed using constructive cost model (COCOMO) analysis by Boehm BW [14]. Bearing in mind that the COCOMO measure is the total lines of code delivered by the development team. Finally the result of the COCOMO-analysis is compared to a simple source code count of delivered source code. Comparing the Total Adjusted Function Points to Delivered Source Lines of Code (SLOC), similar to the two step work effort validation [2] (Figure 1).

Method

Persistent store

The website database was a relational database of type MYSQL version 5.3 (or lower). The tables were defined with primary keys, unique index and auto-increment. No foreign keys constraints, triggers or stored procedures. Web-tier. Most of the source files were PHP files with HTML, CSS - files and some libraries in java Script. PHP class definitions were part of the code so both structured programming and object-oriented programming was present. The model-layer was object-oriented.

The system is evaluated using function point analysis [12]. The metric is evaluated for: internal logical files (ILF), external interface files (EIF), external Input (EI), external output (EO) and external inquiry (EQ).

Internal logical files (ILF): Entity (Table 1) count in the relational database schema. The complexity of the entities graded initially as: below 8 attributes - (LOW), 8-16 attributes - (AVERAGE) and above 16 attributes (HIGH).

External interface files (EIF): Was not initially found relevant, but library calls could be considered. For example, the calendar functions.

External input (EI): PHP-files including DML-statements INSERT, UPDATE and DELETE executed as dynamic SQL. The search was a done by 'search in files' with notepad++, and visual inspection (Notepad++ 2007-2018). Server side code was considered and no stored procedures

***Corresponding author:** Anders L, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark, Tel: 4520898668; E-mail: knh487@di.ku.dk, knh@di.ku.dk, plan@lassena.dk

Received March 27, 2018; **Accepted** May 10, 2018; **Published** May 18, 2018

Citation: Lassen A (2018) Function Point Analysis FPA on A Team Planning Website Based on PHP and MYSQL. J Inform Tech Softw Eng 8: 237. doi:10.4172/2165-7866.1000237

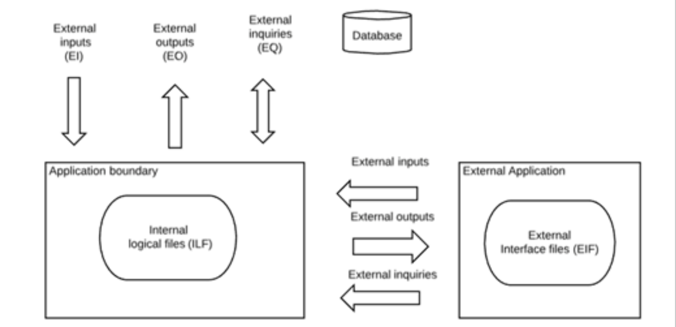

Figure 1: The boundaries of the myteam website for FPA-evaluation. The application boundary is defined as the boundary to the MYSQL database. The external application is taken as the PHP-source-code on the web-server. External inquiries (EI) are files with SELECT statements. External input are files with DML statements INSERT, UPDATE and DELETE. External outputs (EO) would be SMS and email services.

or triggers found to include. The complexity was assessed by counting and weighing the DML statements found in each file.

External inquiry (EQ): PHP files with SELECT statements as Dynamics SQL. Files with INSERT, UPDATE and DELETE statements are not counted but treated as external input files.

External output (EO): These are reports, screens, messages. SMS messaging is an example of external output considered from these weighted measures according to unadjusted function points (UAF) was calculated [1]. General system characteristics (GSC) were evaluated for their degree of influence (DOI) summing to the total degrees of influence (TDI): data communication; distributed data processing; performance; heavily used configuration; transaction rate; on-line data entry; end-user efficiency; online update; complex processing; reusability; installation ease; operational ease; multiple sites; and facilitate change. Each GSC was rated for degree of influence (DOI) on a scale from 0 to 5: not present or no influence (0); incidental influence (1); moderate influence (2); average influence (3); significant influence (4); and strong influence (5). The value adjustment factor (VAF) was calculated as 0.01 times TDI + 0.65. The total adjusted function points (TAFP) was calculated as the unadjusted function points (UAF) times the value adjustment factor (VAF)

$$VAF = (TDI * 0.01) + 0.65 \quad (1)$$

$$FP = UAF * VAF \quad (2)$$

Once calculated, the adjusted function points (FP) was used to assess the project size using the constructive cost model (COCOMO). The function point conversion table was examined and initially PHP was compared to java or C++ [1]. The conversion rate for java or C++ are the same. Average source LOC per function point is 53, and average source LOC for a 210 FP application is 11130 LOC. Since we have the source code [13], the actual LOC can be counted and compared to the estimated project size.

Results

The evaluation of the database relations was done in one pass. The evaluation of the PHP-files was done in several passes.

Count-1: Simple file count.

Count-2: Inspection of files for SELECT statements. Presence of Select in comments was disregarded and file Complexity was based on the number of SELECT-statements.

Count-3: Same evaluation based on inspection for INSERT-statements.

Count-4: Same evaluation based on inspection for UPDATE- and DELETE-statements.

Internal logical files (ILF)

Every relation in the database was considered. Some of the relations have media files. Media are implemented as attributes of type LONGBLOB or BLOB. All these relations are AVERAGE candidates.

External measures

External Interface files (EIF): No external interface files are determined at this point, unless the website configuration is to be considered. An external interface file must be generated or maintained by another system [1]. This measure is initially set to count 10 and weight AVERAGE (Table 1).

External Input (EI): These are input screen. Here the PHP code is inspected to determine user input. The initial file count done in Notepad++ by simple keyword search is tagged count-1 in Table 2. The result of the first count was 30 files with INSERT-statements, 49 files with UPDATE statements and 38 files with DELETE statements. Initially set to complexity AVERAGE.

Further passes are done with code inspection carefully. Keywords in comments and variable are discarded from counts. Files with several DML statements are judged for complexity and account for recurring files in the first count. A file with SELECT and DML modification statements (INSERT, UPDATE, DELETE) should only be classified as an external input file. EI should be reduced.

After the final pass, assessment of complexity is complete. Many overlaps of data manipulation statements are present in the same files and the total number of files included is 29. Files in the 'classes' folder are model-classes for the major relational entities and implement SELECT, INSERT, UPDATE and DELETE statements in various member functions. The final count is given in Table 2 (11 LOW, 7 AVERAGE, 11 HIGH).

External output (EO): These are reports, screens, messages. Here we know the SMS service is very important, but how many places are the SMS services called? Likewise we account for an email service. We estimate is 10 files, average complexity.

External Inquiry (EQ): Enquiry forms are listings; screens that are informational; SELECT statements. All files with keyword 'SELECT' were inspected using 'find in files' in notepad++ (Notepad++ 2007-2018). The initial file count was 86. In the most cases SELECT's would be simple, say 80%, so 86 files are divided into 16 files of AVERAGE complexity and 70 files with LOW complexity.

SELECT-statements and dropdown html are the vast candidates. Select is used in a HTML-tag for one option in a dropdown box. Select also is found in comments. Upload is a library used that is not included, even though some coding efforts must be done to facilitate upload (40 LOW, 13 AVERAGE, 5 HIGH).

Two further passes were done to inspect for data modifications statements. After inspection for INSERT. UPDATE and DELETE statements files initially classified External Inquiry (EQ) are classified as External input files and the EQ count reduced accordingly. In the last pass several UPDATE and DELETE-statements found and moved several EQ-AVERAGE and EQ-HIGH file to EI-files. The final assessment of complexity for External inquiry files (EQ) are

Relation	Attributes	Complexity (low, Average, High)	Referenced in code (notepad++)	Cardinality (tuples)
Availability	5	LOW	10 hits in 2 files	100
Comments	6	LOW	69 hits in 20 files	73
Contents (media, blob)	9	AVERAGE	112 hits in 23 files	318
Diary (media, blob)	5	AVERAGE	258 hits in 18 files	98
Equalizations	7	LOW	14 hits in 3 files	294
Events	15	AVERAGE	321 in 49 files	1175
Expenses	7	LOW	42 hits in 4 files	594
Expence ToPers	2	LOW	10 hit in 2 files	2454
Faqs	5	LOW	5 hits in 3 files	6
Gallery	8	AVERAGE	336 hits in 24 files	56
Help Table	2	LOW	2 hits in 2 files	6
Links	6	AVERAGE	44 hits in 15 files	119
Main Team (media, longblob)	23	HIGH	65 hits in 20 files	38
No SMS	2	LOW	12 hits in 4 files	1
Pers Category	4	LOW	12 hits in 6 files	123
Pics (media, longblob)	11	AVERAGE	51 hits in 13 files	981
Positions	11	AVERAGE	77 hits in 4 files	100
PosNeg	3	LOW	64 hits in 16 files	10127
Race Diary	15	AVERAGE	78 hits in 5 files	5
reminderLog	8	LOW	40 hits in 6 files	9218
Sponsor (media, blob)	8	AVERAGE	56 hits in 5 files	0
Stat	6	AVERAGE	613 hits in 61 files	382501
team (media, blob x2)	30	HIGH	Common name 1949 hits in 99 files	209
Team To Member	5	LOW	67 files in 11 files	177

Table 1: Relations and their complexity. LOW 0-7. AVERAGE 6-15. HIGH 23-30 plus binary objects. SQL-DML reference in PHP code with table name. Cardinality is the number of tuples in each relation.

DML	Complexity	Files
IUD	LOW	11
IUD	AVERAGE	7
IUD	HIGH	11
SUM		29

Table 2: DML modification statements (INSERT, UPDATE, DELETE). After the final pass many overlaps in files have been identified. The total number of files included is 29.

DML	Complexity	Files
SELECT	LOW	31
SELECT	AVERAGE	5
SELECT	HIGH	1

Table 3: The complexity external inquiry, after four passes.

summarized in Table 3 (31 LOW, 5 AVERAGE, 1 HIGH).

Computing the unadjusted function points

The unadjusted function points in Table 4 are calculated using the weights [12]. The ILF complexity is taken from Table 1. The EIF Complexity is not determined and set ad hoc to (10 files and AVERAGE complexity). EI is set to the file count with INSERT, UPDATE and DELETE, corrected for recurrence, comments and variable- and function names. EO is set to the SMS estimate (10 AVERAGE). Further inspection will change this. EQ is adjusted for recurrence of SELECT's and conflicting data manipulation statements. Total unadjusted function point = 580.

The external UAF count is higher than the internal UAF count. Initially we found that the external UAF counts out performed the internal UAF count, but final inspection has reduced this concern.

Count-4	Complexity									
	Low			Average			High			
	N	W	S	N	W	S	N	W	S	S
Internal logical files (ILF)	12	7	84	10	10	100	2	15	30	214
External Interface files (EIF)		5		10	7	70		10		70
External Input (EI)	11	3	33	7	4	28	11	6	66	127
External Output (EO)		4		10	5	50		7		50
External Inquiry (EQ)	31	3	93	5	4	20	1	6	6	119
Total Unadjusted function points (UAF)										580

Table 4: Total unadjusted function points (UAF). After finished evaluation. Weights (w) applied [12].

The PHP-source code is less than twice as complex that the MYSQL data model of the two measures of complexity set EIF and EO, EO is to some extent reasonable. Of the two output modalities have been identified and styles emails can add to complexity. Taken together they are probably overestimated.

GSC and total adjusted function points

The unadjusted function points can now be weighted with a set of general system characteristics (GSC). The 14 system characteristics and the degrees of influence of each of the General system characteristics [10] are listed in Table 5 column "Degree of influence [1]".

Add hoc setting for general system characteristics for this application was done. The unadjusted function point (UAF) of Table 5 was used to calculate the total adjusted function points. Generally the degree of influence have been reduced to a total of 23 degrees of influence. Only one system characteristic (Multiple Sites) was set to average influence (3) as the application could be refurbished to several platforms, and the site give rise to 2 code bases. Seven system characteristics were set to moderate influence (2): Data Communication, On-line Data Entry, End User Efficiency, Online Update, Reusability, Installation Ease, and Operational Ease. Five system characteristics were set to incidental influence (1): Performance, Heavily Used Configuration, Transaction Rate, Complex Processing, and Facilitate Change. One system characteristic was set to not present or no influence (0): Distributed data processing.

The value adjustment factor for this study was found to be 0.88. A lower degree of influence than presented by Jack TM [12]. The total adjusted functions points (TAFP) for this project was 510 FP.

COCOMO (Constructive cost model)

The COCOMO analysis takes the total adjusted function point measure and converts to a measure of lines of delivered source code (LOC). In our case PHP and java script are taken as the java and C++ measure of 53 LOC per function point and 11130 average source LOC for a 210 FP application [12] (equation 4). My expectation for the current 510 FP measure for TAFP would be 2.4 * 11130 lines of code = 27442 LOC (when UAF = 580 and TAFP = 510), see equation 5. Which I hope will be found to be an over estimation for the original PHP site [13]. The estimated number of lines of code can then be converted to 27,4 KDSI (1000 delivered source instructions = 1000 LOC) by dividing by 1000 (equation 6).

$$\text{Estimated LOC}_{\text{Mar}}\text{:(TAFP / 210 FP) * 11300 LOC/53FP) (4)}$$

$$\text{Estimated LOC}_{\text{E4}}\text{: (510 FP/210 FP = 2,4) * 11300 LOC = 27442 LOC (5)}$$

$$\text{Estimated KDSI}_{\text{E4}}\text{: 27442 LOC /1000 LOC/KDSI = 27,4 KDSI (6)}$$

In COCOMO a man-month is 152 hours. In COCOMO first decide if the project is organic (expect few problems), Embedded (expect problems) or semi-detached (in-between). An embedded project scales to 3.6 × KDSI$^{1.20}$. KDSI = 27,4 gives 191 Person-Months. I would go with the lower project classification (organic to semi-detached), 77-122 person-months. The estimate is 19-30 months for a 4 crew team, or 2+ years (Table 6).

General System Characteristic	Degree of influence	Degree of influence
Data Communication	3	2
Distributed data processing	2	0
Performance	4	1
Heavily Used Configuration	3	1
Transaction Rate	3	1
On-line Data Entry	4	2
End User Efficiency	4	2
Online Update	3	2
Complex Processing	3	1
Reusability	2	2
Installation Ease	3	2
Operational Ease	3	2
Multiple Sites	1	3
Facilitate Change	2	1
Total degrees of influence (TDI)	40	23
VALUE ADJUSTMENT FACTOR (VAF) VAF = (40 * 0.01) + 0.65 = 1.05 VAF = (TDI * 0.01) + 0.65	1.05	0.88
UAF	200	580
Total adjusted function points (TAFP) TAFP = 200 * 1.05 = 210 TAFP = UAF * VAF	210	510

Table 5: Calculation of the value adjustment factor (VAF) and the total adjusted function point (TAFP) or just function points (FP).

The actual lines of code counted is about 43000 LOC including 20700 LOC for 3-party code. The KDSI measure of COCOMO is a measure of delivered source code instructions. This amounts to 43000-20700= 22300 lines of delivered source code (22,3 KDSI). Compared to the result of the constructive cost model estimate of 27,4 KDSI , this is an overestimation by 23%.

Discussion

The first key question here is training. The subjective measure of complexity in a smaller custom website, compared to corporate wide systems. Does this lead to overestimation? Yes. In this function point analysis only the relational tables an their complexity was held against the PHP code as an external application. The function point analysis was calculated from a database standpoint. There are other factors that have only been touched.

It was surprising that function point analysis of a custom website developed by one programmer and operational over a period of 6 years had the estimation of 510 function points (FP) converting to an expected 27000 lines of code. I had expected less complexity. Experience with FPA will give more precise estimates for each parameter and even bring the FPA closer to the source code count. In this study the boundary elements considered were primarily entity- and transactional complexity. They seemed a tangible constraint on metrics explored. Other metrics could be considered. A metric for algorithmic complexity (AT) that also is an interesting metric, but may be more academic than operational even in systems of modest size [4].

The Java code calibration is a candidate for debate. In this study it worked out well, but I must also note some complex PHP and Java script-files were not included. This would only increase the measure. And training would cater for this. In the COCOMO analysis overestimation could also be biased by my ad hoc setting of the various degrees of influence.

A reason to retrospective make a function point analysis in this case was the author's lack of luck to debug and support the site after the creator passed away and the vendor upgraded the PHP-version, rendering the site down. This lack of skills can be attributed in some part to Fredric Brooks – 'The mythical man month' [5] but also the teachings of Peter Naur, 'Computing a human activity' and 'Programming as theory building' [15-17]. In the section 'program life, death and revival' ties well into the problem not having access to programmers with working knowledge [16]. Barry Boehm [5]and others list issues with project estimation, also covers unfamiliarity with existing source code [3,5,15]. This is a real obstacle or spike if you will, in reviewing and debugging existing code. It is the author's view that more frequently

Project type	Person-months	KDSI	Person-months	Team4-months
Organic	Person-months= 2.4 * KDSI$^{1.05}$	27,4	77	19
Semi-detached	Person-months= 3.0 * KDSI$^{1.12}$	27,4	122	30
Embedded	Person-months= 3.6 * KDSI$^{1.20}$	27,4	191	48

Table 6: Calculating person months and team months for a four person team (person-month divided by 4) based on KDSI=27,4

than admitted; it is the wiser choice to re implement the code in face of rejection of the initial strategy. Support for this argument can also be in comparing computing as text production to theory building [16]. In this case the following observations contribute to understand the current system down: Broken links. Possible missing URL resolutions; Application state could not be debugged and restored; Vendor upgrade coincided with mourning period.

Conclusion

The Internal Logical File complexity holds for the number of files/entities, but the complexity (LOW; AVERAGE; HIGH) could be overestimated for a smaller custom website.

Using DML (Data Manipulation Language) as a marker for EI an EQ in a website seems operational in a retrospective study (where the coding has been done). The initial keyword search identifies key participating potential relevant external files. A code inspection is necessary to ass's complexity and relevance.

It was found that the "myteam" custom website consist UAF=580 FP unadjusted function points, TDI=23 Total degrees of influence, Value adjustment factor, VAF=0,88; Total adjusted function points TAFP=510FP.

The COCOMO analysis showed an estimated project size of 27,4 KDSI or 27442 LOC. Based on 27,4 KDSI the project type was classified as organic to semi-detached, and project estimate of 66-82 person-months or 16-21 team-months for a four person team. An overestimation of 23% is found compared to the current count of the actual lines of code. The function point analysis explained the complexity quite well.

References

1. Albrecht AJ (1979) Measuring application development productivity. Proceedings of SHARE7GUIDE IBM Applications Development Symposium, Monterey, California pp: 83-92.
2. Albrecht AJ, Gaffney JE (1983) Software function, source lines of code and development effort prediction: A software science validation. IEEE Transaction on Software Engineering SE-9: 639-647.
3. Treble S, Douglas N (1995) Sizing and Estimating Software in Practice: Making MK II Function Points Work. McGraw Hill.
4. Henderson GS (1992) The application of function points to predict source lines of code for software development. Thesis.
5. Hira A, Boehm B (2016) Function point analysis for software maintenance. ISESEM.
6. Chen T (2008) The Application of the function point analysis in software developers' performance evaluation. 4th int. conf. on wireless communications, networking and mobile computing pp:1-4.
7. Saxena V, Shrivastava M (2009) Performance of function point analysis through UML modeling. ACM SIGSOFT software engineering notes 34: 1-4.
8. de Freitas Junior M, Fantinato M, Sun V (2015) Improvements to the function point analysis method: A systematic literature review. IEEE Trans Engineering management 62: 495-506.
9. Sudhakara GP, Patnaik AFS (2012) Measuring productivity of software development teams. Serbian journal of management 7: 65-75.
10. Sheetz SD, Henderson D, Wallace L (2012) Understanding developer and manager perceptions of function points and source lines of code. The Journal of systems and software 82: 1540-1549.
11. Dennis A, Haley WB (2000) Systems analysis and design: An applied approach. New York. John Wiley.
12. Marchewka JT (2010) Information technology management. International student edition. 3rd Edition Pp165-168.
13. Simonsen HJ (2008-2014) www.myteam.dk. A team planning website for Danish yachtracing crews participating in national and international match race. Domain www.myteam.dk.
14. James Cadle J, Yeates D (2008) Project management for information systems. 5th Edition. Person prentice Hall.
15. Brooks, Jr. FP (1975) The mythical man-month. Essays on software engineering. Addison-Wesley.
16. Naur P (1985) Programming as theory building. Micro processing and microprogramming 15: 253-261.
17. Naur P (1992) Computing: A human Activity. Wesley.