ENHANCEMENT IN FUNCTION POINT ANALYSIS

Archana Srivastava[1] , Dr. Syed Qamar Abbas[2] , Dr.S.K.Singh[3]

[1] Amity University, Lucknow, India
asahai@amity.edu, srivastavaarchana@yahoo.com,
[2]Director, Ambalika Institute of Management & Technology , Lucknow, India
[3]Professor, Amity University, Lucknow, India

ABSTRACT

Early and accurate estimation of software size plays an important role in facilitating effort and cost estimation of software systems. One of the commonly used methodologies for software size estimation is Function Point Analysis (FPA). The purpose of Software size estimation and effort estimation techniques is to provide a useful measure of the software complexities, efforts, and costs involved in software development. Despite almost three decades of research on software estimation, the research community has yet not able to provide a reliable estimation model for End-User Development (EUD) environments. EUD essentially out-sources development effort to the end user. Hence one element of the size and effort is the additional design time expended in end-user programming. This paper discusses the concept end-user programming and enhancement of FPA by adding end-user programming as an additional General System Characteristic (GSC).

KEYWORDS

size estimation, end user programming, function point analysis, end user development

1. INTRODUCTION

End-User Programming system aims to give some programmable system functionality to people who are not professional programmers. One of the most successful computer programs of all times is the spreadsheet applications. The primary reason for its success is that end users can program it without going into the background details of logic and programming. However, end user programming is rare in other applications and where it exists usually requires leaving conventional programming, for example AutoCAD provides LISP for customisation, and Microsoft applications use Visual Basic. More effective mechanism for users is to customize existing applications and create new ones as and when needed.

End-user programming is defined as “Creating a data structure that represents a set of instructions either by explicit coding or by interaction with a device. The instructions are executed by a machine to produce the desired outputs or behaviour” [12].

End-User Programming will be driven by increasing computer literacy and competitive pressures for rapid, flexible, and user driven information processing solutions. These trends will push the software marketplace toward having users develop most information processing applications themselves via application generators. Some example application generators are spreadsheets, extended query systems, and simple, specialized planning or inventory systems [15].

DOI : 10.5121/ijsea.2012.3610

End-user programmers, who will generally know a good deal about their applications domain and relatively little about computer science in contrast to the infrastructure developers will generally know a good deal about computer science and relatively little about applications [16].

Effort estimation for software projects has proven to be an elusive and expensive problem in software engineering. On one hand, stakeholders expect precise estimates in the early stages of a project; on the other hand, reliably producing those numbers is extremely difficult and may well be technically infeasible. Boehm et al. report that estimating a project in its first stages yields estimates that may be off by as much as a factor of 4. Even at the point when detailed specifications are produced, professional estimates are expected to be wrong by ±50%. [14]

Project estimation involves translating a set of business objectives or requirements into a measure of product "size." This size measure is then used to estimate the effort, duration, and quality of the final software product. The ability of a system engineer or project manager to align the business objectives with the technical estimates leads to well informed business decisions [19].

2. RELATED WORK

There are many models for software estimation available and prevalent in the industry. Researchers have been working on formal estimation techniques since 1960. Early work in estimation which was typically based on regression analysis or mathematical models of other domains, work during 1970s and 1980s derived models from historical data of various software projects. Among many estimation models expert estimation, COCOMO, Function Point and derivatives of function point like Use Case Point, Object Points are most commonly used **[20**].

While Lines of Code (LOC) is most commonly used size measure for 3GL programming and estimation of procedural languages, IFPUG FPA originally invented by Allen Albrecht at IBM has been adopted by most in the industry as alternative to LOC for sizing development and enhancement of business applications. FPA provides measure of software functionality based on end user view of application software functionality. Albrecht's view was that it is the function of the system, or what it does that is our primary interest. The number of actual lines of code taken to deliver this function is a secondary consideration. The measure which Albrecht developed is called the Function Point (FP).

The method of sizing software in terms of its function expressed in Function Points is now very widely used. It is interesting to note that FPA came about, not because a new measure of system size was requested, but because productivity was becoming increasingly important; it was out of the need to measure productivity that FPA was conceived.

2.1 Analysis of End User Programming

End user expects advanced interactive applications that are easy to use and easy to learn. As user interface become easier to use they become harder to create. User interface developers need tools which provide a rich support for development of advance user interfaces. U^2ML tool is vital for user interface and end user programming. U^2ML looks after all the specification of a modern day's end user design, development and customisation tool **[13]**.

It is foreseeable that the number of active end-user developers will soon exceed the population of professional programmers. As such, there is a growing need to understand the possible risks and benefits associated with EUD activities. Amongst others, relevant criteria include quality (robustness, usability, learnability, etc.), cost, maintainability, and fit-for-purposefulness.

Given the diversity of end-user developers, their activities and conditions, software development frameworks must be flexible enough to respond rapidly to changing contexts and requirements. This is currently infeasible with traditional Software Engineering methodologies. A more attractive solution is to empower end-users to design, integrate and adapt software to meet their changing needs. Despite warnings that EUD may lead to throw-away software **[11]**, end-users will continue to maintain their software whilst they remain motivated to do so. Several research efforts have developed new tools to help end users create effective software. These tools will have significant impact on the software size and hence on the effort required for development.

Over 64 million Americans used computers at work in 1997, and it was estimated that this number will grow to 90 million in 2012, including over 55 million spreadsheet and database users and 13 million self-reported programmers **[10]**. Existing characterizations of this end user programming becoming common in software usage but it do not provide any guidance on the impact of end user programming on estimating software size and effort.

3. END USER PROGRAMMING CHARACTERISTICS

There are basically only two target users in the real world:

- Developers, who want to see the source code and to write imperative code. These users do not trust model-driven approaches, because they feel this can reduce their freedom in application development;
- Non-developers, who want to ignore all the technical issues and have simple, possibly visual or parameter-based configuration environments for setting up their applications.

There are two general approaches to helping users to design. In one case the computer is an intelligent design assistant that tracks the user's actions and infers what might be required. This approach is based on programming-by-example (Lieberman, 2001) and extends adaptable user interfaces that automatically change to fit a user profile or react to the user's behaviour. In the other approach, initiative is left with the user and the system provides powerful tools to support design activity (e.g. Agentsheets: Repenning, 1993). End-user development is a complex field which includes different approaches to helping users instructs machines and design artifacts.

User interfaces of development tools may not be a complex theoretical issue, but acceptance of programming paradigms can be highly influenced by this aspect too. The user interface comprises, for instance, the selection of the right graphical or textual development metaphor so as to provide users with intelligible constructs and instruments. It is worth investigating and abstracting the different kinds of actions and interactions the user can have with a development environment. The end user aspects are prearranged into five phases as in the table 1. In each phase there are interrelated end user programming aspects. These will be the characteristics to be considered whenever the software developers want to estimate for the software size and effort. Hence, end user programming and online help for end user are integrated in existing FPA as GSC.

Table 1
Proposed End User characteristics Formulation

Step	**End User & Online Help Aspect**
Plan(P)	End user requirement
Design(D)	End User features, Functional Features
Code(C)	Coding of tools, review and audit
Test(T)	Review & Testing
Deploy(D)	Software installation & Monitoring

4. REVIEW OF FUNCTION POINT ANALYSIS

Function Point Analysis (FPA) [3][14] consists of two main parts in the measurement. In the first part following **function**alities are counted while counting the **function point**s of the system.

- Data **Function**ality
    - Internal Logical Files (ILF)
    - External Interface Files (EIF)
- Transaction **Function**ality
    - External Inputs (EI)
    - External Outputs (EO)
    - External Queries (EQ)

The second part is 14 General System Characteristics (GSCs) that measured from 0 to 5 nominal scales. These characteristics contribute to Value Adjusted Factor (VAF). The final function point count is obtained by multiplying the VAF times the Unadjusted Function Point (UAF). These 14 GSC's are : Data communication, Distributed functions, performance, heavily used configuration, transaction rate, online data entry, End user efficiency, Online Update, complex processing, Reusability, installation ease, multiple sites, facilitate change **[17]**. The standard equation for estimation is:

$$FP = UFP * VAF \quad (1)$$

Where UFP = Unadjusted Function Point and VAF = Value Adjusted Factor
As mentioned, the total number of UAF is accumulated from five components as in (2). The simplified equation is as follows:

$$UFP = EI + EO + EQ + ILF + EIF. \quad (2)$$

The weights are assigned to each component based on transactional and data function types. For VAF, it is calculated from the summation of 14 GSCs as in (3).

$$VAF = 0.65 + 0.01 \times \sum_{i=1}^{14} C_i$$

Where

C_i = degree of influence for each General System Characteristic
i = is from 1 to 14 representing each GSC
Σ = is summation of all 14 GSCs.
Later on function point was enhanced using the security feature as an additional GSC.

$$VAF = 0.65 + 0.01 \times \sum_{i=1}^{14} C_i + \text{security} \qquad \textbf{[9]}$$

In recent years, several research projects such as Search Computing, ResEval1, and FAST2 spent substantial effort towards empowering end users (sometimes called expert users, to distinguish them from generic, completely unskilled users), with tools and methods for software development therefore it is proposed to consider End-User Programming as individual characteristic in GSC's in FPA model.

5. END USER PROGRAMMING ENHANCEMENT FOR FPA

The restriction to 15 factors seems unlikely to be satisfactory for all time. Other factors may be suggested now, and others will surely arise in the future. A more open-ended approach seems desirable. A new factor end user programming can also be added to the existing model as end user programming practices if inbuilt in the software in the form of additional tools enhances the features of the software as well as increases the software size.

$$VAF = 0.65 + 0.01 \times \sum_{i=1}^{14} C_i + \text{security} + \text{end user programming}$$

i.e.

$$VAF = 0.65 + 0.01 \times \sum_{i=1}^{16} C_i$$

Where

C_i = degree of influence for each Genaral System Characteristic
i = is from 1 to 16 representing each GSC
Σ = is summation of all 16 GSCs.

For ease of giving degree of influence, following end user characteristics are listed in Table 2. The user can identify the number of characteristics that might take into consideration during development. The number will indicate the score for degree of influence. Table 2 and table 3 will be the additional characteristic sheet for GSC's in FPA model.

TABLE 2
EUP characteristics

S.No	EUP facilities in software
1	Programming by example
2	Creating throw away codes
3	Creating reusable codes
4	Sharing reusable code
5	Easily understandable codes
6	Security features in codes for more control by end users
7	Authentication features
8	Personnel security
9	Inbuilt feedback about the correctness
10	Verification
11	Tools for analyzing by debugging
12	Error detection tools
13	Testable codes
14	Availability of online help
15	Self – efficacy: High sense of control over the environment
16	Perceived ease of use: Apart from extrinsic motivation intrinsic motivation (enjoyment) should be present.
17	Perceived usefulness
18	Flexible codes
19	Scalability features
20	Ease of Maintenance

As in the EUP characteristics table 20 items are basically considered. The suggested score for degree of influence is as below.

TABLE 3: DEGREE OF INFLUENCE FOR END USER PROGRAMMING

Score as	Descriptions to determine Degree of influence
0	None of the above
1	1<=n<=4
2	4<n<=8
3	8<n<=12
4	12<n<=16
5	16<n<=20

Where n is the number of applicable end user programming characteristics.

6. CONCLUSION AND FUTURE RESEARCH

We project that in 2012, over 13 million workers will “do programming” in a self-reporting sense; however, based on spreadsheet and database usage, it seems likely that the number of end user programmers will be much higher. Consequently, end user programming environments will definitely have impact on software size and effort estimation for software projects. Sizing is a key estimating activity. If the sizes of major deliverables can be predicted within 5 to 10 percent, then the accuracy of the overall estimate can be quite good. The proposed enhancement of FPA is not

analysed at this stage. In the future, we will perform additional analysis with software having end user programming, as well as collect and analyse new survey data, in order to better understand the software usage of end users. We anticipate that more precise estimates and characterizations of end user practices will help researchers target further work in developing languages and tools to assist end users in programming tasks.

REFERENCES

[1] Verner, June M. and Tate, Graham, "A Model for Software Sizing", Journal of Systems and Software, IEEE Software, pp. 173-177, July 1987.

[2] Albrecht, Allan J. and Gaffney (Jnr), John E., "Software Function Source Lines of Code and Development Effort Rediction: A Software Science Validation", IEEE Transactions on Software Engineering, Vol. SE-9, No. 6, pp. 639-647, Nov. 1983.

[3] N. E. Fenton and S. L. Pfleeger, 1997. Software Metrics: A Rigorous and Practical Approach, 2nd Edition Revised ed. Boston: PWS Publishing.

[4] L. M. Laird, and M. C. Brennan, 2006. Software Measurement and Estimation: A Practical Approach, Wiley-IEEE Computer Society Pr, ISBN: 0-471-67622-5.

[5] Forselius, P., 2004. Moving from Function Point Counting to Better Project Management and Control, IWSM/MetriKon Presentation.

[6] C. R. Symons, Software Sizing and Estimating − MkII FPA (Function Point Analysis), John Wiley and Sons, Chichester, U.K., 1991.

[7] A. Abran, M. Maya, J. M. Desharnais, and D. St-Pierre, “Adapting function points to real-time software,” American Programmer, Vol. 10, 1997, pp. 32-43.

[8] C. Jones, Applied Software Measurement: Assuring Productivity and Quality, McGraw-Hill, New York, 2008.

[9] N. A. S. Abdullah1, R. Abdullah2, M. H. Selamat2, A. Jaafar2,Software Security Characteristics for Function Point Analysis, Proceedings of the 2009 IEEE IEEM

[10] Scaffidi, C., Shaw, M., and Myers, B. The “55M End User Programmers” Estimate Revisited. Technical Report CMUISRI-05-100, Carnegie Mellon University, Pittsburgh, PA, 2005.

[11] G. Fischer, E. Giaccardi, Y. Ye, A. G. Sutcliffe, and N. Mehandjiev. Meta-design: A manifesto for end-user development. Communications of the ACM, 47(9):33–37, September 2004.

[12] Contributions, Costs and Prospects for End-User Development, Alistair Sutcliffe, Darren Lee & Nik Mehandjiev

[13] “Object based designing of pattern using U2ML” in proceeding of International conference of advances in computer vision and information technology (ACVIT-2007)

[14] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R. Cost models for future software life cycle processes: COCOMO 2.0. Annals of Software Engineering, Special Volume on Software Process and Product Measurement (1995).

[15] Appendix C: COCOMO II Process Maturity led by Dr. Barry Boehm

[16] http://sunset.usc.edu/research/COCOMOII/Docs/modelman.pdf

[17] http://www.devshed.com/c/a/Practices/An-Overview-of-Function-Point-Analysis/3

[18] http://www.cs.toronto.edu/%7Esme/papers/2005/ESEC-FSE-05-Aranda.pdf

[19] http://www.crosstalkonline.org/storage/issue-archives/2011/201101/201101-Stark.pdf

[20] http://approachtoproject.com/component/k2/item/10-software-estimation-techniques.html

Author

Author Archana Srivastava has done M.Sc(Maths), MCA, M.Tech(IT) and is working as Sr. Lecturer in Amity Institute of information Technology, Amity University, Lucknow, India. She is persuing phD in software engineering. She has more than 12 years of teaching experience.

Co-Author Syed Qamar Abbas completed his Master of Science (MS) from BITS Pilani. His PhD was on computer-oriented study on Queueing models. He has more than 20 years of teaching and research experience in the field of Computer Science and Information Technology. Currently, he is Director of Ambalika Institute of Management and Technology, Lucknow. He is actively involved in academic and research work. Till date he has published over 50 research papers in national and international journals.

Co-Author Dr. S.K.Singh is Professor and Programme Director in Amity Institute of Information Technology, Amity University, Lucknow, India. He has done M.Sc(Maths), MCA, M.Tech(IT) and PhD in Applied Computer Science. He has more than 18 years of teaching experience. Till date he has published over 14 research papers in national and international journals.