ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

End User Programming: An Important Cost Driver

Archana Srivastava[1], Dr.S.K.Singh[2], Dr. Syed Qamar Abbas[3]
[1] Asst. Professor, Amity Institute of Information Technology, Amity University, Lucknow, India,
[2]Professor, Amity Institute of Information Technology , Amity University, Lucknow, India, [3]Director, Professor, Ambalika Institute of Management & Technology , Lucknow, India

Abstract*— Day per day competition in the software industries is increasing; in such scenario accurate effort estimation has become an important and essential task, which is providing basis for other software activities like scheduling and, planning. Efforts estimation has become a challenge for IT industries. In a world that is not predictable, improvisation, evolution, and innovation are a necessity. Most programs today are written not by professional software developers, but by people with expertise in other domains working towards goals supported by computation. Understanding how programming fits into end users' everyday lives is central to not only the design of the software project but will also act as an important cost driver in estimating the effort required for developing project incorporating End User Development and End User Programming(EUD/ EUP). This paper is proposing a systematic framework of effort estimation, which will involve all those parameter that may affect the efforts of a software project incorporating EUD/EUP features. It also discusses about how end-user software engineering activities and the technologies may affect the overall software development effort estimation.

Index Terms— **EUD/EUP, effort estimation, cost drivers, COCOMO model.**

I. INTRODUCTION

End-user participation in the software development process is not a new phenomenon, but it has usually been limited to involvement of users in the initial phases of the process (Mørch, 1998). End-User development can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artifact. Today, some forms of EUD have found widespread use in commercial software with some success: recording macros in word processors, setting up spreadsheets for calculations and defining e-mail-filters. While these applications only realize a fraction of EUD's potential and still suffer from many flaws, they illustrate why empowering end-users to develop the systems they are using is an important contribution to letting them become active citizens of the Information Society. One promising approach is end-user development (EUD), the practice of users creating, modifying, or extending programs for personal use [1, 2]. This approach has two main benefits. One, it puts systems design in the hands of the domain experts who are most familiar with what needs should be met. Two, it scales with both a rapid increase in users and the increasing rate of change of many business processes.

II. NEED OF END USER PROGRAMMING

EUD has now found its first widespread use in commercial software, and end-users have taken it up with some success: recording macros in word processors, setting up spreadsheets for calculations and defining e-mail-filters. While these applications only realize a fraction of the EUD potential and still involve many issues, they illustrate why empowering end-users to develop the systems they are using is an important contribution to letting them become active citizens of the information society. One example for future use of EUD technology is the field of home appliances, i.e., all sorts of electronic devices that people will use at home and that will become interconnected and very flexible in the near future. This creates a mass-market where people will want to adapt systems to their specific contexts. and requirements and where they will value personalized, adaptive and anticipatory systems. Given estimates like that of Brad Myers [3] (Carnegie- Mellon-University) that in 2005 there will be 55 million end users doing EUD while there will be only 2.75 million software professionals, the importance of research on EUD becomes apparent. Not only to limit the damage caused by erroneous EUD activities but also to fully exploit the potential benefits of quick and precise system adaptations that only end-users can perform at a reasonable cost.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

According to Sutcliffe, EUD essentially outsources development effort to the end user. Because there is always some effort to learn an EUD tool, the users' motivation depends on their confidence that it will empower their work, save time on the job and/or raise productivity. In this model, the benefits to users are initially based on marketing, demonstrations and word-of-mouth. Once the technology is put into use, experience of actual benefits becomes the key motivator.

III. REQUIREMENTS FOR INCLUDING EUP FEATURES

The goal of EUD is to empower end-users to adapt IT-systems themselves as much as possible, thus letting them become the initiators of a fast, cheap and tight co-evolution between themselves and the systems they are using. To allow for this level of end-user development, IT-systems must be made considerably more flexible and they must support the demanding task of EUD in various ways: they must be easy to use, to teach, understand, and learn. Also, users should find it easy to test and assess their EUD activities. EUD has now found its first widespread use in commercial software, and end-users have taken it up with some success: recording macros in word processors, setting up spreadsheets for calculations and defining e-mail-filters. While these applications only realize a fraction of the EUD potential and still involve many issues, they illustrate why empowering end-users to develop the systems they are using is an important contribution to letting them become active citizens of the information society. One example for future use of EUD technology is the field of home appliances, i.e., all sorts of electronic devices that people will use at home and that will become interconnected and very flexible in the near future. This creates a mass-market where people will want to adapt systems to their specific context sand requirements and where they will value personalized, adaptive and anticipatory systems.

IV. END-USER ACTIVITIES PARADIGM

Two types of end-user activities are identified from a user-centered design perspective:

i) Parameterization or Customization. Activities that allow users to choose among alternative behaviors (or presentations or interaction mechanisms) already available in the application. Adaptive systems are those where the customization happens automatically by the system in reaction to observation the user's behavior.

Examples of activities belonging to the Parameterization or Customization type are:

- **Parameterization**- In this commonly occurring case, the user wishes to guide a computer program by indicating how to handle several parts of the data in a different way; the difference may simply lie in associating specific computation parameters to specific parts of the data, or in applying different program functionalities to the data.
- **Annotation**- The users write comments next to data and results in order to remember what they did, how they obtained their results, and how they could reproduce them.

ii) Program Creation and Modification. Activities that imply some modification, aiming at creating from scratch or modifying an existing software artifact. Examples of these approaches are: Programming by Example (also called

Programming by Demonstration), Visual Programming, Macros, and Scripting Languages. EUD more properly involves the second set of activities since with the first set the modification of software is restricted to strictly predefined options or formats. However, we often want to design for a “gentle slope” of increasing complexity to allow users to easily move up from the first to the second set of activities.

Examples of activities belonging to the Program Creation and Modification type are:

- **Programming by Example**- Users provide example interactions and the system infers a routine from them.[6]
- **Incremental programming**- This is close to traditional programming, but limited to changing a small part of a program, such as a method in a class. It is easier than programming from scratch.
- **Model-based development**- The user just provides a conceptual description of the intended activity to be supported and the system generates the corresponding interactive application [7].
- **Extended annotation or parameterization**- A new functionality is associated with the annotated data or in a cooperative environment users identify a new functionality by selecting from a set of modifications other people have carried out and stored in shared repositories.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

V. COST MODELING EUD

In 1995, Boehm et al. introduced COCOMO version 2.0, an extension and elaboration of the earlier COCOMO cost estimation framework, in order to address various shortcomings in the earlier version [4]. Many of these shortcomings had arisen due to changes in the way professional programmers performed their trade (such as the increasing focus on producing reusable modules). To accommodate these changes, COCOMO 2.0 incorporated a variety of new variables representing cost drivers ranging from "team cohesion" to "process maturity" in an attempt to better capture the details of the professional programming environment. Many variables in each version of COCOMO do not apply to end-user programmers. These workers, who range from marketing specialists to business administrators, generally build small software. They often develop their software on their own rather than in organized teams (making the "team cohesion" variable less meaningful), and they make little attempt to apply a disciplined software development process (making the "process maturity" variable less meaningful). In the words of the original COCOMO 2.0 technical report, "The User Programming sector does not need a COCOMO 2.0 model. Its applications are typically developed in hours to days, so a simple activity-based estimate will generally be sufficient" [4]. EUD essentially out-sources development effort to the end user. Hence one element of the cost is the additional design time expended. Another cost is learning. This is a critical cost in EUD because end users are busy people for whom programming is not their primary task. They only tolerate development activity as a means towards the end that they wish to achieve; for instance, creating a simulation, experimenting with a design, building a prototype. Learning to use an EUD environment is an up-front cost that has to be motivated with a perceived reward in improved efficiency or empowered work practice. Cost of errors is a significant penalty for EUD users both in operation and learning. Cost of EUD to the user can be assessed in terms of the time taken to learn to use the EUD product and possibly its language, the requirements or specification effort entailed in refining general ideas into specific instructions, the programming effort, followed by time for testing and correcting from errors. The trade-offs between effort and reward can be summarized as a set of motivating principles for EUD: The aim for all design is to achieve an optimal fit between the product and the requirements of the customer population, with minimal cost. Generally, the better the fit between users' needs and application functionality, the greater the users' satisfaction; however, product fit will be a function of the generality/specialization dimension of an application. This can be summarized in the principle of user motivation:

- The user motivation to accept an EUD technology will be inversely proportional to product complexity and variability in the user population.

The consequences of this law are that EUD will consume more effort with a heterogeneous user population, because getting the right fit for each sub-group of individuals becomes progressively more challenging and expensive. The second consequence is that larger scale and more complex applications will be more difficult to develop; people have a larger learning burden with complex products. The second follows on from the first principle, in that general technologies may not motivate us to expend development effort because the utility they deliver is less than a perceived reward from satisfying our specific requirements:

- User motivation to customize and learn EUD software will be proportional to the perceived utility of the software delivering usable and useful applications.[5]

As EUD is now very much in demand but in all the previously existing software cost models it is not included as a cost driver.

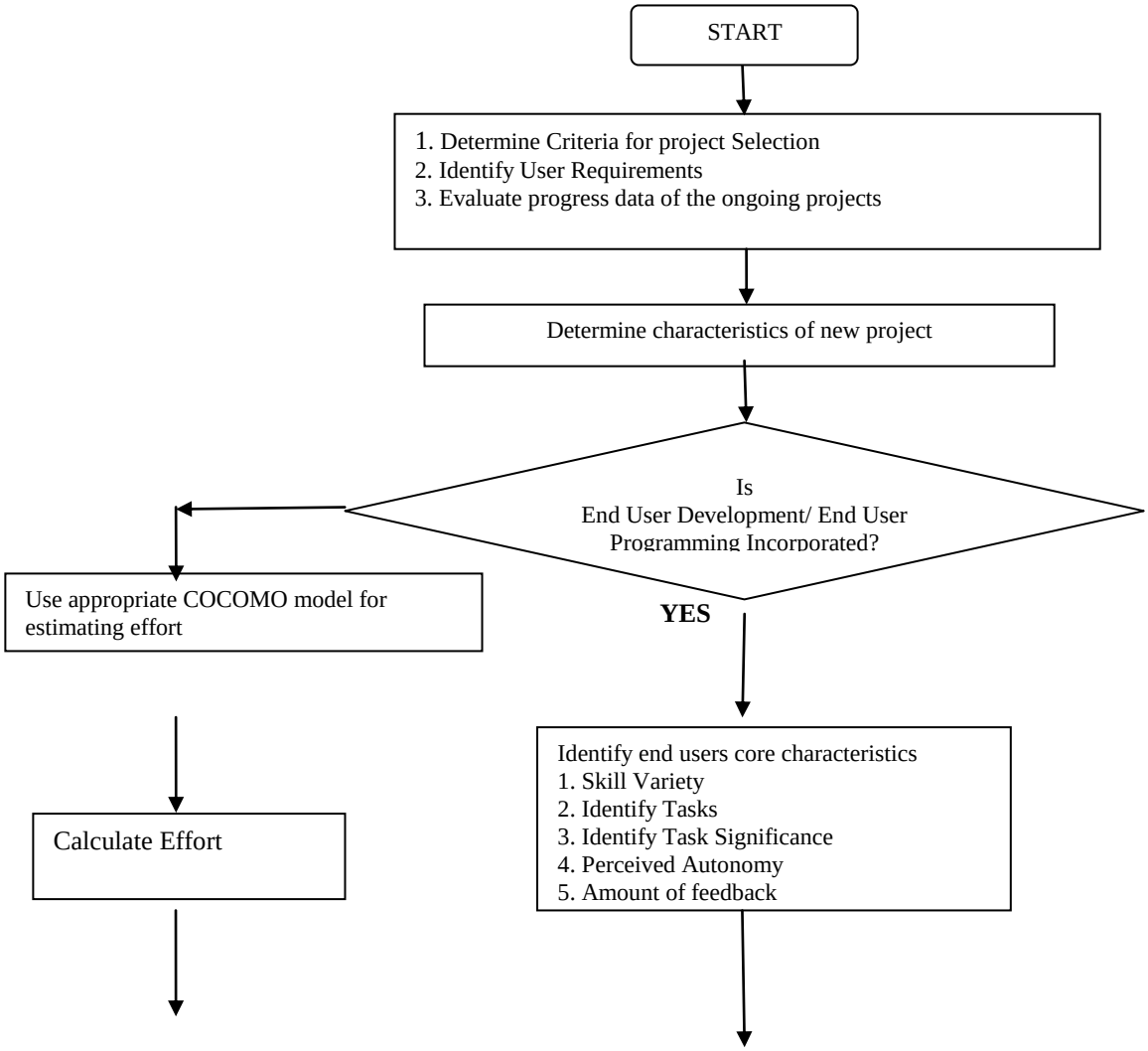
VI. IMPACT OF EUD ON SOFTWARE DEVELOPMENT PROCESS

Enabling end-users to substantially alter Software systems creates a number of issues concerning correctness and consistency, security and privacy. One approach to handle these issues is to let the system monitor and maintain a set of desired system properties during EUD, like integrity and consistency by, for example, allowing only safe operations. But as H. Lieberman (Massachusetts Institute of Technology) points out [8], user errors and incompleteness of information cannot be ruled out altogether, whereas users may often be able to supply missing information or correct errors if properly notified. For this reason, handling the issues above may often best be done by a cooperation of both user and system. Another issue of EUD is how to make users aware of existing EUD functions and how to make these functions easily accessible. For addressing these issues the system must have adequate tools and online help that will guide the user according to their requirements and also some testing tools has to be implemented to check the user developed programs. All these requirements will lead to following changes in the development models:

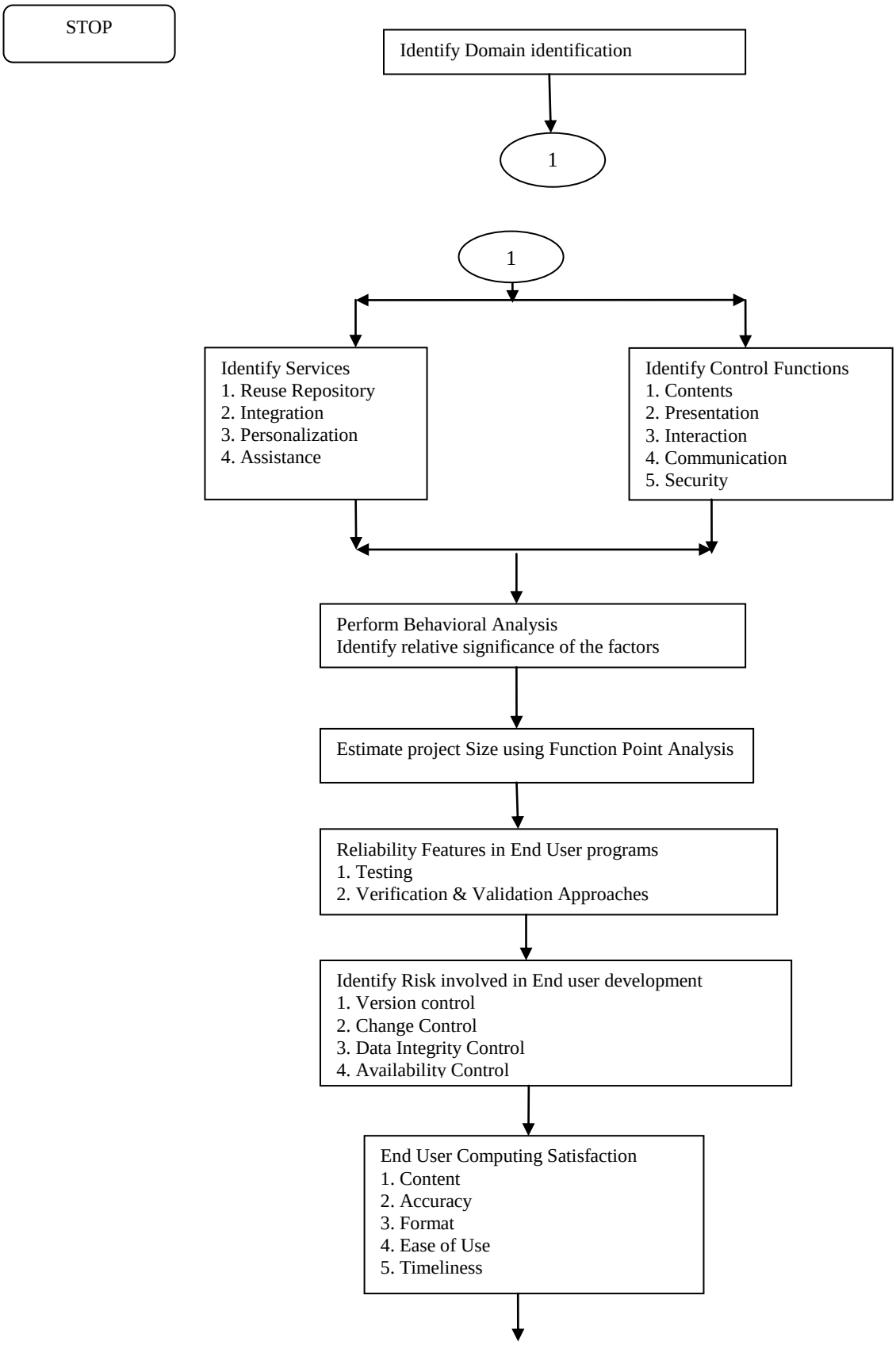
ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

Requirements for EUD software will be generated in a different manner. Because developers are users themselves, there may not be need for an elaborate requirement analysis phase as a major activity preceding the construction of the software system. Though rapid changes of requirements need not be avoided; they are desired because the computer in such contexts is used to explore the new possibilities and to find the implicit requirements of users. Designing will get enhanced. Because designers will have to develop additional tools to facilitate development practices to end user. Inbuilt testing tools have to be designed to test the user's development. These testing tools will provide correctness, authenticity and reliability to user's development. Lots of security features has to be implemented. Special tools for testing the user developed modules will be the main requirement. Software testing is conducted differently. Because domain expert developers themselves are the primary users, complete testing is not as important as in the case when the developers are not the users. Collaboration takes place along different dimensions. In self-conscious software development, a team of developers is often organized before the project starts — in unself-conscious software development, a predefined project team does not exist. Collaboration is spontaneous and opportunistic rather than planned. The path to the acquisition of knowledge and skill for software development is different. Due to the lack of professional training, domain experts are more likely to acquire software knowledge in a piecemeal fashion and demand-driven manner. Their knowledge is more fragmental than systematic. Hence special help and tools and tips will be required for increasing the usability feature. Software will evolve in a different style. The system is evolved gradually by a large number of people who make small contributions each time. Evolution is more spontaneous and situational due to the co-adaptivity of tools and their users. Every customer modification implies costs because it has to be maintained by the customer. Each time a support package is imported there is a risk that the customer modification may have to be adjusted or re-implemented.

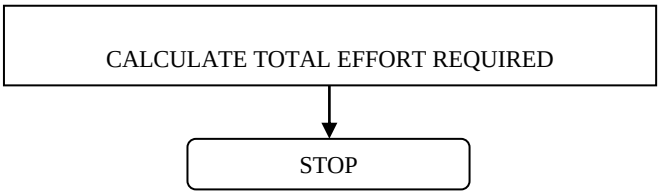
VII. SOFTWARE COST ESTIMATION FRAMEWORK FOR SOFTWARE PROJECTS INCORPORATING EUD/EUP FEATURES



ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

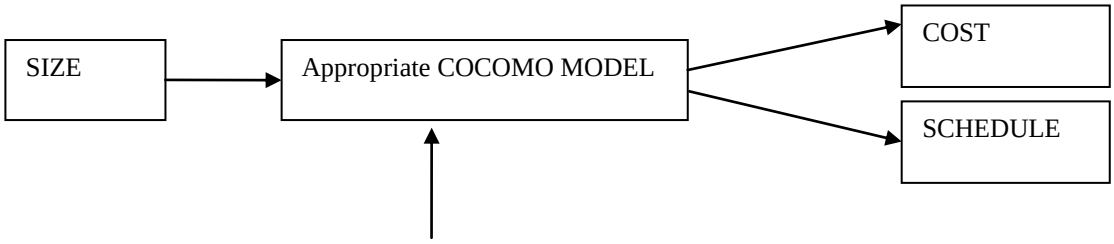


ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

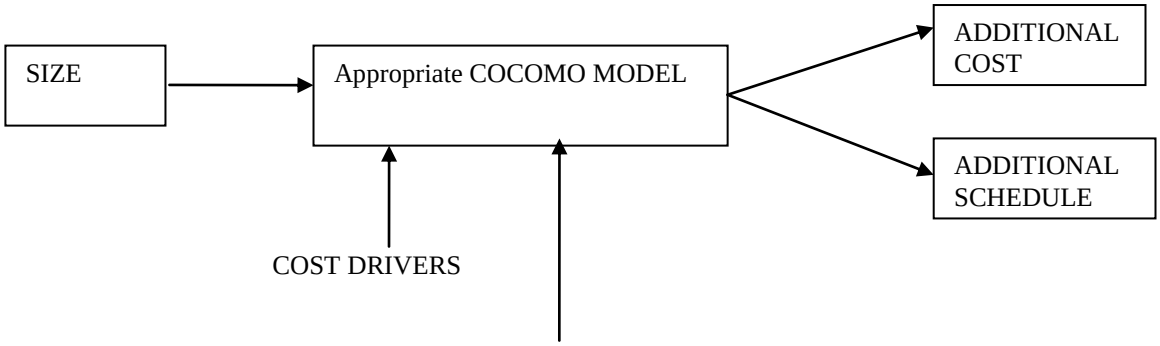


VIII. INTERPRETATION OF THE FRAMEWORK

In software without EUP/EUD features Effort/Cost can be calculated as:



Where, size can be estimated as KLOC or Function Points. In software incorporating EUP/EUD features, considering the factors from the above system total effort required to develop a software project incorporating the end user development features will be increased and can be depicted as:



Where, size can be estimated as KLOC or Function Points. Applicable factors according to the software domain related to EUD/EUP can be derived from the above framework. The increase in software cost enhances the end user satisfaction and quality in end user - developed applications. Quality is defined as the degree to which an application of "high grade" accomplishes or attains its goal from the perspective of the user. The major advantages of EUC are that the problems associated with eliciting information requirements are shifted to the insiders and at the same time ownership is immediately transferred to the users. Having users develop their own applications eliminates the problems associated with ineffective communications between the systems analyst and the end users [11].

IX. CONCLUSION

Hence EUD can be seen as an important contribution to create a user-friendly Information Society, where people will be able to easily access information specific to their current context and to their cognitive and physiological abilities or disabilities. People will have access to adapt IT-systems to their individual requirements, and if all actors will be involved the design of IT-systems will find a higher common acceptance. Also, end-users must be motivated to pay the cost of performing EUD operations. Software cost estimation is accounted as an important factor while making estimations in Software Engineering. There is no simple way to make an accurate estimate of the effort required to develop software systems incorporating EUD/EUP because of many reasons like unclear user requirements, lack of knowledge on new technology, changing technology requirements and not considering the

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

reusable component cost of End user development. Using the above framework identification of cost factors of end user development will be more accurate.

IX. FUTURE WORK

In future we will develop a mathematical model based on the given framework i.e. including features of COCOMO model as well EUP features. A survey will be conducted on various software development companies to exactly identify impact of the weights of additional EUP/EUD factors in quantitative terms. After thorough analysis actual and estimated cost of various software's, the model will be analyzed for accuracy in estimation. Measurement models can vary across samples and must be tested and retested before they are accepted as valid Software Effort Estimation Models. Hence future work will also involve hypothetical testing and retesting of the model.

REFERENCES

[1] Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwing, M., Scaffidi, C., Lawrance, J., Lieberman,., Myers, B., Rosson, M.B., Rothermel, G., Shaw, M. and Wiedenbeck, S. 2010. The state of the art in end-user software engineering. ACM Computing Surveys. (2010).

[2] Lieberman, H., Paterno, F., Klann, M. and Wulf, V. 2006. End-user development: An emerging paradigm. End User Development. (2006), 1–8.

[3] Myers, B. Making Programming Easier by making it More Natural. Presentation at first EUD-Net workshop in Pisa, Italy, 23./24. Sep. 2002.

[4] B. Boehm et al. Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. Annals of Software Engineering Special Volume on Software Process and Product Measurement (J. Arthur and S Henry, eds), J.C. Baltzer AG, Science Publishers, Amsterdam, The Netherlands, 1995.

[5] Evaluating the Costs and Benefits of End-User Development Alistair Sutcliffe, Centre for HCI Design, School of Informatics, University of Manchester, PO Box 88, Manchester M60 1QD, UK, a.g.sutcliffe@manchester.ac.uk

[6] Liebermann, H. (2001). Your Wish is My Command: Programming by Example. San Francisco, Morgan Kaufmann.

[7] Paternò, F. (2001). Model-based Design and Evaluation of Interactive Applications, Springer Verlag.

[8] Lieberman, H. Your Wish is My Command: Programming by Example for End-User Development. Presentation at first EUD-Net workshop in Pisa, Italy, 23./24. Sep.2002.

[9] Sutcliffe, Alistair (July 2005). "Evaluating the costs and benefits of end-user development" (PDF). ACM SIGSOFT Software Engineering Notes (ACM) 30 (4): 1–4. doi:10.1145/1082983.1083241. http://portal.acm.org/citation.cfm?id=1082983.1083241. Retrieved 2008-05-29.

[10] IBM Global Technology Services June 2011, "Taking end user services to the next level with IBM's Right-to-Left strategy".

[11] "Quality End User-Developed Applications: Some Essential Ingredients by Donald L. Amoroso, University of Colorado, Colorado Springs and Paul H. Cheney, University of South Florida.

AUTHOR BIOGRAPHY

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

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.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

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 20 years of teaching experience. Till date he has published over 15 research papers in National and International journals.