A Survey on Effort Estimation of Object-oriented Programming Systems from use case diagrams

Sai Sruthi

Abstract— In object-oriented analysis, use case models depict the utilitarian prerequisites of a future programming framework. Estimating the framework could be carried out by measuring the size or intricacy of the use cases in the use case model. The size can then serve as info to an expense estimation technique or model, so as to process an early gauge of expense and exertion. Evaluating programming with use cases is still in the early stages. This paper portrays a product estimating and expense estimation strategy dependent upon use cases, called the 'Use Case Points Method'. The strategy was made a few years back, however is not well known. One of the reasons may be that the strategy is best used with elegantly composed use cases at a suitable level of practical subtle element.

Index Terms— Conversion Factor, EFactor, LOC, TFacor, Use Case Point (UCP), UML, UAW, UUCW

.

1 INTRODUCTION

While working at Ericsson in the late 1960s, Ivar Jacobson formulated what later got known as use cases. Ericsson at the time displayed the entire framework as a set of interconnected pieces, which later got to be "subsystems" in UML. The squares Ire found by working through formerly specified 'movement cases', later known as use cases. Jacobsen left Ericsson in 1987 and made Objectory AB in Stock- holm, where he and his partners created a procedure item called 'Objectory', a contraction of 'Article Factory'. A graphing method was produced for the idea of the use case. In 1992, Jacobson concocted the product philosophy OOSE (Object Oriented Software Engineering), a use case driven technique, one in which use cases are included at all phases of improvement. These incorporate dissection, plan, acceptance and testing. In 1993, Gustav Karner created the Use case Points strategy for evaluating article arranged programming. In 1994, Alistair Cockburn built the 'On-screen characters and Goals theoretical model' while composing use case guides for the IBM Consulting Group. It gave direction as how to structure and compose use cases.

Use case modelling is a prevalent and broadly used method for catching and depicting the useful prerequisites of a product framework. The architects' of UML suggest that engineers take after a use case driven improvement process where the use case model is used as info to outline, and as a premise for confirmation, acceptance and different manifestations of testing.

A use case model characterizes the practical extent of the framework to be created. The utilitarian extension therefore serves as a premise for top-down appraisals. A strategy for utilizing use case models as a premise for evaluating programming advancement exertion was presented by Karner. This technique is affected by the capacity focuses strategy and is focused around closely resembling use case focuses. The use of an adjusted form of the use case focuses technique was discovered that characteristics of a use case model are dependable pointers of the measure of the ensuing usefulness. Use case models have additionally been discovered appropriate as a premise for the estimation and arranging of undertakings in a product change venture. In any case, I have been not able to discover contemplates that portray the use case focuses estimation transform in subtle elements. The point of this paper is to give a nitty gritty depiction of the technique used and encounters from applying it.

Estimation of software testing effort is a standout amongst the most essential parts of the whole testing life cycle transform as it is specifically relative to the cost of the venture. Estimation has an effect on all the 3 most essential parts of a client need – Time, Cost and Quality. A right estimation helps in conveying the items in correct time. If the estimation is not correct it might lead to delay in deliverables, increased cost and inappropriate results. An imperative essential for applying a use case based estimation technique is that the use cases of the framework under development have been distinguished at a suitable level of subtle element. The use case model may be organized with a shifting number of performing artists and use cases. These numbers will influence the estimates. The division of the utilitarian prerequisites into use cases is, nonetheless, outside the extent of this paper.

- *Author Sai Sruthi is currently pursuing masters degree program in information technology in SRM University, India. E-mail: saisruthi9.prasanthi@gmail.com*

2 VARIOUS ESTIMATION METHOD

John Smith of Rational Software depicts a technique showing a system for estimation dependent upon use cases made as lines of code. There does not appear to be any more research finished on this strategy, in spite of the fact that the device 'Gauge Professional', which is supplied by the Software Productivity Center Inc, and the apparatus "Costxpert" from

Marotz Inc. produce assessments of exertion for every use case figured from the amount of lines of code. There are heaps of routines accessible at this moment for testing estimation like Test Case Points, Function Point and so forth. Use Case technique is likewise picking up prevalence for evaluating programming exertion. It can used extremely useful in case of offering ventures as the use cases are one of the first or frequently the main data accessible to start with of a product venture.

Elective routines for estimation dependent upon use cases are on the groundwork for including capacity focuses, which turn may be utilized to acquire an evaluation of exertion and an alternate is to gauge the amount of lines of code (LOC) in the completed framework. This number of LOC is accordingly utilized as the support for an assessment. These two systems show up more perplexing than the one I have utilized as they individually make presumptions on the relationship between use cases and capacity focuses, and between use cases and the amount of LOC in the completed framework. These suppositions have not been tried. The focal point of these strategies, nonetheless, is that they may misuse the far reaching knowledge with estimation utilizing capacity focuses or lines of code.

Upgrade measures the measure of the issue including and ordering extension components a task. The set of use cases in the undertaking's use case model is one sort of degree component. Different potential outcomes are, for instance, the task's classes, parts and pages. Qualifiers are connected to every degree component. The many-sided quality qualifier characterizes every degree component as straightforward or complex. The apparatus gives a set of default measurements, extrapolated for a fact on more than 100 ventures. The client can likewise redo metric information to transform evaluations balanced for an association. Advance composes the degree components and metric information to register an appraisal of exertion and expense. I plan to assess this apparatus all the more completely. My impression is that the device obliges adjustment to the specific association to give a sensible evaluation. In addition, the expense of procurement and preparing makes it less open than the technique with copartnered spreadsheet that I have utilized.

3 USE CASE METHOD

The use instances of the framework under development must be composed at a suit- capable level of point of interest. It must be conceivable to include the transactions the use case depictions to characterize use case unpredictability. The level of point of interest in the use case depictions and the structure of the use case has an effect on the accuracy of assessments dependent upon use cases. The use case model might likewise hold a shifting number of performing artists and use cases, and these numbers will again influence the appraisals. This method is very Ill suited for bidding projects as most of the time use case is the only information available at the beginning of a project. The Use Case point method considers the technical and environmental factors which can be refined further to achieve more accurate estimates. This can be used to illustrate productivity benchmarks across an organization since it is independent of test cases.

The principal playing point to evaluating with use case focuses is that the methodology might be robotized. Some use case administration apparatuses will naturally tally the amount of use case focuses in a framework. This can spare the group an incredible arrangement of evaluating time. Obviously, there's the counter contention that an appraisal is just comparable to the exertion put into it. This scientific method gives more accurate and precise results over any traditional method available for effort estimation.

A focal point is that it ought to be conceivable to secure an authoritative normal usage time for every use case point. This might be extremely of service in determining future timetables. Lamentably, this depends intensely on the supposition that all use cases are reliably composed with the same level of point of interest. This may be a false supposition, particularly when there are different use case creators. Here no any detail requirements are required for estimation.

An interest to utilize case focuses is that they are an exceptionally immaculate measure of size. Great estimation methodologies permit us to independent evaluating of size from determining length of time. Use case focuses qualify in this respect in light of the fact that the measure of a provision will be autonomous of the size, ability, and knowledge of the group that actualizes it.

4 VARIOUS FACTORS IN USE CASE POINT METHOD

This area gives a short review of the steps in the use case point method. This estimation technique obliges that it ought to be conceivable to include the amount of transactions each one use case. A transaction is an occasion happening between a performing artist and the framework, the occasion being performed totally or not in the slightest degree. The First step for every type of estimation is to calculate the size of activity to be performed. Second is to calculate Effort estimation.

1. Size Estimation

The four steps of the use case point technique are as takes after:

1.1. The performing artists in the use case model are classified as basic, normal or complex. A basic on-screen character speaks to an alternate framework with a characterized API; a normal performing artist is an alternate framework associating through a convention, for example, TCP/IP; and a complex performer may be an individual communicating through a graphical user interface or a site page. A weighting variable is relegated to every performer class:

- Simple: weighting factor 1
- Average: weighting factor 2
- Complex: weighting factor 3

The total unadjusted actor weight (UAW) is calculated counting the number of actors in each category, multiplying each total by its specified weighting factor, and then adding the products.

1.2. The use cases are additionally ordered as straightforward, normal or unpredictable, contingent upon the amount of transactions, incorporating the transactions in elective streams. Included or developing use cases are not recognized. A straightforward use case has 3 or fewer transactions; a normal use case has 4 to 7 transactions; and a complex use case has more than 7 transactions. A weighting component is alloted to each one use case classification:

- Simple: weighting factor 5
- Average: weighting factor 10
- Complex: weighting factor 15

The unadjusted use case weights (UUCW) is ascertained checking the amount of use cases in every class, increasing every classification of use case with its weight and including the items. The UAW is added to the UUCW to get the unadjusted use case focuses (UUPC).

UUCP = UAW + UUCW

1.3. The use case points are adjusted based on the values assigned to a number of technical factors (Table 1) and environmental factors (Table 2).

TABLE I. TECHNICAL COMPLEXITY FACTORS

Technical Factor	*Description*	*weight*
T1	Distributed System	2
T2	Performance	1
T3	End User Efficiency	1
T4	Complex Internal Processing	1
T5	Reusability	1
T6	Installability	0.5
T7	Usability	0.5
T8	Portability	2

TABLE II. ENVIRONMENTAL FACTORS

Environment Factor	*Description*	*Weigh t*
F1	Familiarity with Life-Cycle model used	1.5
F2	Application domain experience	0.5
F3	Experience with development methodologies used	1
F4	Analyst capability	0.5
F5	Team motivation	1
F6	Stability of requirements	2
F7	Use of part-time team members	-1
F8	Use of difficult programming language	-1

Each one variable is doled out a quality between 0 and 5 relying upon its accepted impact on the venture. A rating of 0 means the element is immaterial for the undertaking; 5 would not joke about this is vital. The Technical Factor (TCF) is computed reproducing the quality of each one variable (T1 – T13) in Table 1 by its weight and after that adding all these numbers to get the whole called the Tfactor. At long last, the accompanying recipe is connected

TCF = 0.6 + (.01*TFactor)

The Environmental Factor (EF) is calculated accordingly by multiplying the value of each factor (F1 – F8) in Table 2 by its weight and adding all the products to get the sum called the Efactor. The formula below is applied:

EF = 1.4+(-0.03*EFactor)

The adjusted use case points (UCP) are calculated as follows:

UCP = UUCP*TCF*EF

1.4. Karner proposed a variable of 20 staff hours for every use case point for a venture assessment, while Sparks states that field experience has demonstrated that exertion can run from 15 to 30 hours for every use case point.

2. Effort Estimation

2.1. Conversion Factor (CF):

When the span of a venture has been ascertained as far as Adjusted Use Case Points, the aggregate size needs to be changed over to exertion by duplicating it with a conversion factor. The Conversion factor is characterized as the aggregate testing time needed to test one Use Case Point. The Conversion factor might be inferred by figuring out strategy i.e. by putting the recorded task information in the estimation layout for different technologies. It is 20(hrs) for Java based applications.

Final Effort = UCP * Conversion factor

5 CASE STUDY

1. Actor weight

S.N	*Actor Name*	*Weight*	*Factor*
1.	Actor 1	Complex	3
2.	Actor 2	Medium	2
3.	Actor 3	Simple	2
4.	Actor 4	Simple	1
		Total =	8

2. Use case weight

S.N	*Use case description*	*Weight*	*Factor*
1.	Use case 1	Simple	5
2.	Use case 2	Complex	15
3.	Use case 3	Medium	10
4.	Use case 4	Medium	10
		Total =	40

3. Unadjusted Use Case point (UUCP)

UUCP = Actor weight + Use case weight
= 8+40
= 48

4. Technical complexity Factor calculation

S.N	*Description*	*Weigh t*	*Perceived complexity*	*Calculated Factor*
1.	Distributed system	2	1	2
2.	Performance	1	2	2
3.	End user efficiency	1	3	3
4.	Complex internal processing	1	4	4
5.	Re usability	1	5	5
6.	Install-ability	0.5	3	1.5
7.	Usability	0.5	2	1
8.	Portability	2	1	2

Total TF = 20.5

TCF= 0.6 + (0.01 * TF)
= 0.6 + (0.01 * 20.5)
= 0.6 + 0.205
= 0.805

5. Environmental complexity Factor

S.N	*Description*	*Weight*	*Perceived complexity*	*Calculated Factor*
1.	Life cycle model used	1.5	2	3
2.	Application domain	0.5	1	0.5
3.	Development methodologies	1	3	3
4.	Analyst capability	0.5	5	2.5
5.	Team motivation	1	2	2
6.	Stability of requirements	2	5	10
7.	Part-time team members	-1	2	-2
8.	Different programming language	-1	1	-1

Total = 18

ECF = 1.4 + (-0.03 * EF)
= 1.4 + (-0.03 * 18)
= 1.4 – 0.54
= 0.86

6. Calculate final use case points (UCP)

UCP = UUCP * TCF * ECF
= 48 * 0.805 * 0.86
= 33.23

7. Calculating Final Effort

Final Effort (Hrs) = UCP * conversion factor
= 33.23 * 20
= 664.6

6 CONCLUSION

I led a study on applying a strategy for assessing programming advancement exertion dependent upon use cases, the use case focuses system. The effects show that this technique could be utilized effectively since the use case evaluations Ire near the master gauges. In one case it was likewise near the real exertion. It is in this manner my feeling that the system may help master information.

Additionally, my experience is that applying the use case point strategy in practice is not direct. For instance, the decision of structure for the use case model has an effect on the evaluations. There is hence a need for further studies on the exactness of the assessments when utilizing the use case focuses technique in distinctive sorts of ventures. I additionally accept that it might be suitable to explore how the use case focuses strategy, which gives top-down assessments dependent upon a measure of size, could be joined together with different routines that give bottom up evaluations. The motivation behind utilizing the estimation strategy examined as a part of this paper is to give a complete evaluation to all the exercises. In any case, I accept that a portion of the exercises in an advancement task don't rely on upon size or use case focuses. Along these lines, such exercises ought to be assessed in elective ways and afterward be added to the use case assessment to give a last gauge.

ACKNOWLEDGMENT

This research paper is made possible through the help and support from everyone, including: parents, teachers, family, friends, and in essence, all sentient beings. First and foremost, I would like to thank Professor Dr. Subburaj Ramasamy for his most support and encouragement. He kindly read my paper and offered invaluable detailed advices on grammar, organization, and the theme of the paper. Finally, I sincerely thank to my parents, families, and friends, who provide the advice and support. The product of this research paper would not be possible without all of them.

REFERENCES

[1] L.M Alves, “An empirical study on the estimation of software development effort with use case points”, IEEE 2013.

[2] N.A Ahmed and A.H Ahmed, “Enabling complexity use case function points on service-oriented architecture”, IEEE 2013.

[3] A.B Nassif, L.F Capretz and M Azzeh, “A Treeboost Model for Software Effort Estimation Based on Use Case points,” IEEE 2012.

[4] A.B Nassif, “Software size and effort estimation from use case diagrams using regression and soft computing models”, IEEE 2012.

[5] Qiudong Yu and Chungui Liu, “Application of estimation based on use cases in software industry”, IEEE 2011.

[6] Ribu and Kirsten, “Estimating Object-Oriented Software Projects with Use Cases. Master of Science Thesis,” University of Oslo, Department of Informatics 2001.

[7] J. Smith, “The Estimation of Effort Based on Use Cases”, Rational Software, White paper. 1999

[8] G. Karner, “Use case points: Resource estimation for objector projects,” September 1993.

[9] Cockburn and Alistair, “Writing Effective Use Cases”, Addison-Wesley.

[10] G. Schneider and J. Winters,”Applying Use Cases-A Practical Guide”,Addison-Wesley.