Survey Report on Software Cost Estimation using Use Case Point Method

Preetam Pratap Jena
School of Computer Engineering
KIIT University
Bhubaneswar, Odisha, India
preetampratap@hotmail.com

Samaresh Mishra
School of Computer Engineering
KIIT University
Bhubaneswar, Odisha, India
samaresh2@gmail.com

Abstract - Software estimation is an important activity in software project management. Many software projects fail because of the inaccurate and untimely estimation of cost. Estimation in the early stages of the software life cycle became important though it helps to prevent or reduce project failures. Early estimation became so important while bidding on a project or signing the contract between the client and the developer. The early software estimation is carried out at a point when the details of the problem are not revealed completely. Now a days most of the software industries are focusing on Object-Oriented Software Development. Though the Use Case diagram is prepared after preparing a clear requirement specification, when all the artifacts about the system was clear enough, the UML Use Case Point method can be useful for early estimation. Many researchers and practitioners have done remarkable researches on Use Case Point Method and revealed several artifacts of use case in order to extend the well-known UCP method to support various range of software estimation. In our study, we have surveyed several papers on use case point method and highlighted the proposed metrics or factor by the researchers through this report.

Keywords : Early Software Estimation, Use Case Point, Cost Estimation.

I. INTRODUCTION

Software project estimation is crucial in software development. Effective software project estimation is one of the most important and challenging activities in software development. A proper project planning and control can't be possible without a proper and accurate estimation. Inaccurate estimation causes improper staff management, ineffective cash flow, poor schedule management which indirectly affect the quality of the product. If we give a project more resources than it really needs, then the project is likely to cost more, take longer time to deliver than necessary, and delay the use of resources on the next project.

Now a days most of the software industries are focusing on Object-Oriented Software Development. Though the Use Case diagram is prepared in the early stages of design, after preparing a clear requirement specification, when all the artifacts about the system was clear enough, so this can considered to be useful for early estimation. A use case diagram is the simplest representation of a user's interaction with the system. A use case diagram can portray the different types of users of a system and the different ways that they interact with the system. Use case diagrams are used at a very high level of design. Then this high level design is refined again and again to get a complete and practical picture of the system. A well structured use case also describes the pre condition, post condition, exceptions, direct and alternate scenarios etc. And these extra elements are used predict the size, effort and cost required for the software yet to be developed. In our research, we have surveyed several remarkable works by several researchers and highlighted their proposed metrics and the use case diagram artifacts considered by them.

II. LITERATURE REVIEW

Gustav Karner in 1993 [1], proposed a model which can predict the total amount of resources required for developing a software system with object-oriented principle, in the early phases of software development. In his research, he taken the UML Use-case diagram into consideration, because the Use-case diagram is the simplest representation of a system to be developed and it's prepared after getting a clear requirement specification. Karner named his model as 'Use Case Point Method'. During his research, he first surveyed the Function Point method used for early software estimation and identified some of the shortfalls of it. According to Karner, the greatest advantage

of Function Point is that it doesn't require a specific way to describe the system, for example a specific design method. And some disadvantage is that it can't be computed automatically, because we have to make many subjective decisions and it requires human interference [1].

The UCP method is the extension of Function Point method with the benefit of requirement analysis in the object-oriented process. It starts with measuring the functionality of the system based on the use case model in a count called *Unadjusted Use Case Point* (UUCP). The same technical factors are used as of Function Points. A new factor called *Environmental Factor* is proposed by the author. The author defined the Use Case Points (UCP) as the product of these three factors (i.e. UUCP, Technical Factors and Environmental Factors). The UCPs shows an estimation of the size of the system which can further mapped to man hours in order to calculate the effort required to develop the system.

In the early steps of obtaining use case point, the author classified the actors and use cases according to their complexity and assigned some weight factors[1].

- An actor is defined as “Simple”, if it interacts with the system with the help of a defined application programming interface (API). An actor is defined as “Average”, if it interacts with the help of an Interactive or Protocol-Driven Interface. The actor is defined as “Complex”, if it interacts through a Graphical User Interface. The assigned weight factors are 1,2 and 3 respectively.
- An use case is defined as “Simple”, if it's number of transaction is less than 3.An use case is defined as “Average”, if it's number of transaction is between 4 to 7. The use case is defined as “Complex”, if it's number of transaction is more than 7. The assigned weight factors are 5,10 and 15 respectively.

In the next step the total Actor and Use case weight is calculated :

$$UAW = \sum(\text{No. of actors} * \text{their respective weight factors})$$

$$UUCW = \sum(\text{No. of Use cases} * \text{their respective weight factors})$$

where,

UAW - Unadjusted Actor Weight

UUCW - Unadjusted Use Case Weight

After calculating the total weight of Actors and Use cases, the Unadjusted Use Case Point (UUCP) is calculated :

$$UUCP = UAW + UUCW$$

After obtaining the UUCP, the Use Case Point is calculated by adjusting the UUCP with two factors called Technical Factors (TCF) and Environmental Factors (EF) :

$$UCP = UUCP * TCF * EF$$

The author have considered the Use case point as a smallest unit of size[1]. Further the effort is estimated by mapping the UCP with man-hours.

$$\text{i.e. Effort} = UCP * PH_{per}\ UCP$$

Karner suggested that on an average, 20 man hour effort is required for each UCP[1].In this method, Karner only focused on the actor and use case complexity. The internal details are not taken into consideration. Karner counted the successful scenarios only, the alternative scenarios (exceptions) are not considered.

The Extended Use Case Point (e-UCP) was proposed by Kasi Periyasamy and Aditi Ghode in 2009[2], by extending the original Use Case Point by considering some additional information of use cases such as the relationships between actors, relationships between use cases etc. The authors extended the UCP model with a focus on the internal details of each use case i.e. the Use *case narratives* in order to calculate the Software Size and Effort[2].

In the e-UCP method, the authors used most of the possible aspects of use case model such as actors, use cases, relationships between actors, relationships between use cases, associations between actors and use cases and the detailed narrative of each use case. According to the authors, Use case narratives is essential because it reveals the hidden details of a use case diagram. The authors given a template for a use case narrative by considering the following factors[2] :

- Use case name
- Purpose
- Input parameters
- Output parameters
- Primary actor

- Secondary actor
- Precondition
- Post condition
- Successful scenario
- Exceptions
- Includes list
- Included by list
- Extends list
- Extended by list

The authors have defined a number of steps for calculating UCP, similar to the original UCP method by Karner. The difference is that in their research, the authors added the unadjusted use-case narrative weights while calculating the Unadjusted Use-case point (UUCP). The authors have re-classified both the Actor types and Use-case types and their respective weights.

TABLE I. ACTOR WEIGHT CLASSIFICATION [2]

Actor Type	Weight
Very Simple	0.5
Simple	1
Less Average	1.5
Average	2.0
Complex	2.5
Very Complex	3.0
Most Complex	3.5

TABLE II. USE CASE WEIGHT CLASSIFICATION [2]

Use Case Type	Weight
Simple	0.5
Average	1
Complex	2
Most Complex	3

After re-classifying the actors and use-cases, the authors have assigned weights to the different parameters of a use case narrative in the table given below :

TABLE III. USE CASE NARRATIVE WEIGHT CLASSIFICATION [2]

Use Case Narrative Parameters	Weight
Input parameter	0.1
Output parameter	0.1
A predict in Precondition	0.1
A predict in Post condition	0.1
An action in successful scenario	0.2
An exception	0.1

All the parameters given in the template are not being used in Table 2 because, according to the authors some of the parameters do not contribute towards the coding efforts, and some other factors like 'actors' are already taken into consideration[2].

Calculating e-UCP

1. *Calculate the total use case weight factor :*

   ∑ Simple use cases * WF + ∑ Average use cases * WF + ∑ Complex use cases * WF+
   ∑ Most Complex use cases * WF

2. *Calculate the total actor weight factor :*

   ∑ Very Simple actors*WF +∑ Simple actors * WF +∑ Less Average actors*WF+
   ∑ Average actor * WF + ∑ Complex actor * WF + ∑ Very Complex actor * WF+

∑ Most Complex actor * WF

3. *Calculate the total Use Case Narratives Weight :*

   ∑ Use Case Narratives * their respective Weight Factors

4. *Calculate the Unadjusted Use Case Points (UUCP):*

   UUCP = UseCase_Weight + Actor_Weight + Narratives_Weight

5. *Calculate the Extended Use-case Point (e-UCP) :*

$$e\text{-}UCP = UUCP * TCF * EF$$

where,

$TCF = 0.6 + (0.01 * Ti)$, for $1 \leq i \leq 13$ (Technical Factor)

$EF = 1.4 + (- 0.03 * Ei)$, for $1 \leq i \leq 8$ (Environmental Factor)

e-UCP can be considered as the smallest unit of size. And the required effort can be calculated by multiplying 20 man-hours with the e-UCP as suggested by Karner. This project was given to graduate students of University of Wisconsin-La Crosse as a course project and the estimated effort is nearly matched with the actual effort spent by the students.

Gustavo Bestetti Ibarra and Patricia Villain in 2010[3], proposed a method for estimating software size by considering the "use case size" as the evaluation factor. The size of each use case is derived by taking the following three factors into consideration[3]:

- Use Case Type Points (UCTP)
- Number of Business Rules (NBR)
- Number of User Interface Requirements (NUIR)

In the original UCP method, Karner classified the use cases into three categories according to their complexity; simple, average and complex[1]. According to the authors, this classification depends heavily on the way, the use cases are written and detailed. Hence, the Karner's method is only suitable if use cases are sufficiently detailed at the early stages of development, which is not possible in companies adopting agile approach in their software process [3].

A. Use Case Type Points (UCTP)

The types of use cases defined by the authors here are extensions of two special types of use cases defined by Cockburn : CRUD and Parameterized Use Cases[4].

TABLE IV. USE CASE TYPES [3]

Sl.	Use Case Type	UCTP
1	CRUD Use case	**2**
2	Basic Operation (CRUD)	**1**
3	CRUD Tabular	**2**
4	CRUD Master / Detail	**4**
5	CRUD Detail	**3**
6	Report	**1**
7	System Service	**3**
8	Non-standard Use case	**5**

B. Number of Business Rules (NBR)

These rules are defined by the company itself to make a significant difference with their competitors and the smooth function of their business. It differs from company to company. Some of the examples of business rules are listed below[3] :

1. Calculation rules
2. Information processing rules

The authors have suggested that, the business rules which are already adopted for one use case should not be reused further.

C. Number of User Interface Requirements (NUIR)

It is to specify the requirements of the graphical user interface (GUI) for a use case. According to the authors, it is essential to identify the expectations of users from the GUI in order to interact with the system early in the project. Some examples of interface requirements are:

1. Nesting and automatic loading of information
2. Data display formats

After obtaining the UCTP, NBR and NUIR variables, the authors given the following formula for obtaining the Use case Size (UCS)[3]:

$$\mathbf{UCS = UCTP + \alpha\ NBR + \beta\ NUIR}$$

where α e β are the weights representing the degree of complexity related to the implementation of business rules and graphical user requirements, respectively. These weights are adjusted by each company according to the tools, standards and frameworks utilized in their projects. The final project size (UCP) is calculated by the summation of all use cases sizes (UCSs) adjusted by the technical and environmental factors as defined in the following equation :

$$\mathbf{UCP} = \sum_{n=1}^{u} (\mathbf{UCS}) * \mathbf{TCF} * \mathbf{EF}$$

where u is the number of use cases.

This method was utilized in the Labor Court of Santa Catarina - Brazil to provide support to the negotiation process with third party companies that outsource software development[3].

Yaeghoob Yavari et.al in 2011[5], identified that the Use case point method proposed by Karner doesn't have enough accuracy, because of the inaccuracy in determining Use Case complexity metrics. The authors found that, the definition and the identification of transactions is the most difficult part of UCP. The definition which states that, "Transaction is a set of activities in use-case-scenarios (that is meaningful to the actor), which is either performed entirely, or not at all, and leaves the business of application in consistent state", is not clear enough[8]. Different researchers have defined use case transactions in different ways. Some of them are listed below :

According to Ivar Jacobson, inventor of Use Case; "A transaction of Use Cases is such as a mutual interconnection vector and route within a period of schedule from user to system and again it returns to user. A transaction terminates when system expects one stimulus".

According to Karner, the inventor of UCP;

- A transaction is not always a specific stage of Use Case flow.
- A transaction is not a stimulus. Also a transaction is not an operation of an activity for basis data.

According to M.Ochodek, J, and Nawrocki, transactions have 2 parts:

- Request of user.
- Response to system request [6].

According to the authors, a transaction consists of 4 steps:

- Primary actor sends the request and the data to the system
- The system validates the request and the data;
- The system alters its internal state;
- The system replies to the actor with the result[5].

According to the authors, the transaction doesn't stood as a good and efficient criteria for determining the Use case complexity and calculating Unadjusted Use Case Weights (UUCW) due to the following reasons[5] ;

- Transactions contain ambiguous criterion.
- Transactions contain different complexion levels.
- There is no specific guideline for writing user cases. It varies person to person.

So the authors introduces several new use case complexity metrics in order to reduce project risks and improve customer satisfaction. Those metrics are listed below :

- Type of the Use Case (UCT)
- Priority of the Use Case (UCP)
- Goal Levels of Use Case (UCGL)
- Number of Main and Alternative Scenarios (NMAS)
- Related Actor Types (TRA)

- Number of Related Database Entity (NRED)

According to the above mentioned factors introduced as Use Case metrics, the authors managed to put them all in a table given below in order to determine the complexity level of Use cases:

Use Case ID	Goal (Summary Description)	UCT*WF	UCP *WF	UUGL* WF	NMAS* WF	BR *WF	TRA *WF	NRED *WF	Final Weight
#01									
#02									
TOTAL UUCP =									

After calculating the final weight of each use case, the authors given a formula for calculating the Unadjusted Use Case Weight (UUCW) [5]:

$$UUCW = \sum (\text{Final Weight of each Use case})$$

In this method, authors calculated the UUCW for each individual use case and calculated the final weight by summing them all. Here, the authors didn't consider the Unadjusted Actor Weight (UAW), because in their study they found that, with and without UAW provides similar prediction accuracy[5].

III. SUMMARY

After reviewing all the above mentioned papers, the authors found that, in each paper, the researchers taken several new factors of UML Use Case into consideration for calculating the UCP. The authors summarized them and managed to put them all in the following table:

Sl No.	Paper	Model	Factors	Formula(s) (if any)	What is estimated	Validation
1	G. Karner [1]	Use Case Point (UCP)	• Complexity of Use cases • Complexity of Actors	UCP = UUCP * TF * EF Where, UUCP = UseCase_WeightFactor + Actor_WeightFactor TF = 0.6 (0.01 * TWF) EF = 1.4 + (-0.03 * EWF)	Software Size & Effort	YES Developed for Objectory (now Rational Software).
2	Kasi Periyasamy and Aditi Ghode [2]	Extended Use Case Point (e-UCP)	Use Case Narratives • Input parameters • Output parameters • Precondition • Post-condition • Successful scenario • Exceptions	e-UCP = UUCP * TCF * EF Where, UUCP = UseCase_WeightFactor + Actor_WeightFactor+ Narratives-_WeightFactors	Software Development Cost	NO This method use the same technical and environmental factors proposed by Karner. The authors have planned to revise those factors for better result.

3	Gustavo Bestetti Ibarra and Partricia Villain [3]	Use Case Size	• Use Case Type Points (UCTP) • No. of Business Rules (NBR) • Number of User Interface Requirements (NUIR)	UCS=UCTP+αNBR +βNUIR SIZE = $\sum_{N=1}^{U} UCS * TCF * EF$	Software Size	YES Applied to the project of the Administrative Lawsuit Management System (PROAD TRT/SC) developed by the TRT/SC
4	Yaeghoob Yavari, Mohsen Afsharchi and Mojtaba Karami [5]	New Use Case Complexity Metrices	• Type of Use Case(UCT) • Priority ofUse Case(UCP) • Goal Levels of Use Case(UCGL) • Number of Main and Alternative Scenario(NMAS) • Related Actor Types (TRA) • Number of Related Database Entity (NRED) • Business Rules (BR)	UUCW = (UCT *WF)+ (UCP*WF)+ (UCGL*WF)+ (NMLS *WF)+ (BR *WF)+ (TRA*WF)+ (NRED*WF)	Software Development Cost	NO Yet to be validated. Still in data capturing stage.

IV. CONCLUSION

This paper has highlighted the importance of software size and effort estimation in the early stages of software development. In this survey, we have reviewed several papers on Use Case Point method used for early software estimation. Several researchers have done remarkable work by extending the UCP method with Soft Computing methods such as neural networks, fuzzy logic etc. But the paper has mainly focused on the original UCP method artifacts and highlighted different factors or artifacts of use case dig. used by researchers. It has been found that the effectiveness of the UCP method mainly depends on the way the use case transactions are counted. So a need for further planning is required to simplify the way of counting transactions by redefining them as well as we have planned to introduce some new environmental factors in order to obtain more precise and accurate result.

V. REFERENCES

[1] G. Karner, "Metrics for Objectory", Diploma thesis,University of Linköping, Sweden. No. LiTHIDA- Ex- 9344:21.December 1993.
[2] K. Periyasamy and A. Ghode, "Cost Estimation Using Extended Use Case Point (e-UCP) Model", International Conference on Computational Intelligence and Software Engineering, 2009 .
[3] G. Ibarra and P. Villain. "Software Estimation Based on Use Case Size, Brazilian Symposium on Software Engineering, 2010.
[4] A. COCKBURN, “Writing Effective Use Cases”. Addison-Wesley, 2001.
[5] Y. Yavari. "Software complexity level determination using software effort estimation use case points metrics", Malaysian Conference in Software Engineering, 12/2011
[6] M. Ochodek and J. Nawrocki,” Automatic Transactions Identification in Use Cases In: Balancing Agility and Formalism in Software Engineering,” 2nd IFIP Central and East European Conference on Software Engineering Techniques, 2008.