RESEARCH ARTICLE OPEN ACCESS

Size and Time Estimation in Goal Graph Using Use Case Points (UCP): A Survey

Amir Abbaszadeh Sori And Mohammad Amini
Department of Computer Engineering and Information Technology Amirkabir University of Technology Tehran, Iran

Abstract
In order to achieve ideal status and meet demands of stakeholders, each organization should follow their vision and long term plan. Goals and strategies are two fundamental basis in vision and mission. Goals identify framework of organization where processes, rules and resources are designed. Goals are modelled based on a graph structure by means of extraction, classification and determining requirements and their relations and in form of graph. Goal graph shows goals which should be satisfied in order to guarantee right route of organization. On the other hand, these goals can be called as predefined sub projects which business management unit should consider and analyse them. If we know approximate size and time of each part, we will design better management plans resulting in more prosperity and less fail. This paper studies how use case points method is used in calculating size and time in goal graph.
Keywords: Use Case, Size Estimation, Time Estimation, Goal Graph, Use Case Points

I. INTRODUCTION

Recent changes in organizations and competitive atmosphere caused major changes in infrastructure of organizations such as re-engineering of business processes, development of IT and IS. On the other hand, prosperity and applying these changes depend on realizing stakeholder objectives and strategies. To do so identifying and modeling goals is an important aspect of business management.

In majority of goal modeling techniques, a model is organized in form of a tree or graph. To achieve this structured form, three activity should be carried out including goal classification, goal identification and extraction and indicating their relations [1]. It is most likely many projects are taken or developing at the same time in organizations. Rough estimation and analysis of both time and size of projects influence future schedules including budget plans, managing resources methods, functionality and quality of projects [2]. One of methods to estimate size and time of project, or estimating amount of needed effort based on changes is use case points method. Use case point as a software Size Estimation approach, have been developed by Gustav Karner in 1993. This method is extension of FP analysis [3]. If we want to estimate a project with use case points, first, we need to write all of its use cases.

Use case modelling is an easy technique for capturing and describing functional requirements of a software system. These use cases describe the primary goals which actors interact with system, display how this goals may be delivered. Use cases and goals are used instead of each other. In small scale, each use case provide steps how to achieve specific goal, they are called scenarios. Each step in scenario is called sub goal. This hierarchical relationships between goals and sub-goals need precise requirement analysis tools [4]. In goal-oriented analysis, abstract goals of customers are classified into sub goals. This decomposition process is stored as a graph where nodes represent goals and edges represent dependency relation between the goals [5]. In order to identify, realize and elicit requirements, many tools have been designed and studied in various papers. In this paper we study goal graph that how use case point method can estimate whole size and time of project.

This paper is prepared in 6 section. Second and third section study size and time estimation methods. Section 4, talk about using use case points method in goal graph. Section 5 investigates use case points method and it's characteristic. Conclusion is section 6.

II. SIZE ESTIMATION

Traditional method of Size Estimation includes the number of Source Lines of Code (SLOC), Function Points (FP) and Object Points (OP) [2] another method which we study here is the use case point method that briefly we call use case point. Number of use case points of each project is a function of the following factors: number and complexity of system use case and system actors, a set of non-functional requirements and environmental factors of project [6]. These non-functional requirements are ones which are not written in use case form such as portability, efficiency, maintainability [6] each use case consists of set of scenarios described with a non-official language in business field. These sets indicate transactions among

system and actors [6]. In each use case we face two concept, main success scenario and extensions. The former one refers to main steps of use case should be passed. The latter word refers alternative steps, most are employed for handling errors in use cases [7]. In the following you will see the steps of this method, each step has a formula and final size of each use case is calculated by combined formula which you can see all of them in table (3).

At first step, based on Kerner's definition, each Use case includes a set of points based on number of transactions. Each step in use case is called as transaction. Another word, it is set of activities that all should be done [8]. Then counting steps in use case means we have already counted transaction numbers. It should be noted that because extensions are not main steps, they are not counted in these calculations although in logical and exact estimation they should be considered because of investing time and energy on them [7]. In one of the estimation mechanism, based on transaction numbers we assign a degree of complexity to each use case this way: if number of transaction would be 3 or less, then we classify it as simple level and assign point weight 5. If transaction number be between 4 to 7, it is called average level with weight number 10, in other cases, we consider as complex level with weight point 15 [3, 7]. In another approach, we take into account classes which implement use cases. Based on this theory, we can count number of classes in each use case, therefore, we assign a degree of complexity to each of them. Another approach that we can take is considering type and complexity of user interface which is involved with it [8]. Developers categorised it to 3 level and assigned a weight factor each level. If it involves with one entity or simple user interface, weight point 5 is dedicated. If it involves with equal or more than 2 entity or average user interface weight factor 10 is assigned, in other cases including more complex user interface or higher number of entities it is given 15 as weigth [8]. Total weights of use cases are known as UUCW stands for Unadjusted Use Case Weight.

In second phase, actors within a use case are another aspect that contributes with complexity also. Actor can be a named as a person, any program or hardware device. Here, similar to previous methods, concerning level of complexity, a predefined weight is assigned to each actor. Three level are defined including simple, average and complex. For clarify, a simple actor is another system interacts with the system via API. At this level, weight point 1 is given to it. At another level when a person interacts with system via text-based user interface or another system using HTTP protocol, TCP/IP, SOAP are samples of average actor level.

The number which is given is 2. Also, for third category we can name a human who uses graphical user interface or web page to interact with system. Weight point 3 is considered for this level [7, 8]. Total weights of system actors are known as UAW stands for Unadjusted Actor Weight. UUCP is abbreviation form of Unadjusted Use Case Points. This quantity calculate the unadjusted size of the project or system by following equation.

UUCP=UUCW+UAW (1)

In third step, during projects, there are a set of factors related difficulty of building and completing project. For instance, security, complex processing and so on [3]. Elicited the most 13 important factors and assigned an importance factor. Factors were selected based on based on their importance degree. In the below table you see these requirements. Therefore, these values are summed up as tfactor and Technical Complexity Factor of project is obtained this way:

TCF = 0.6 + (0.01* tfactor) (2)

Table 1. Factors influencing technical complexity of project with their importance degree

Factor	Description	Weight
T1	Distributed system	2
T2	Performance objectives	2
T3	End-user efficiency	1
T4	Complex processing	1
T5	Reusable code	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent use	1
T11	Security	1
T12	Access for third parties	1
T13	Training needs	1

In forth step, environmental factors are used for measuring level of experience of personnel, stability of project, final efficiency and experience level of project members. These factors are rendered as follows in table with their weights. Karner gained these weights by asking and investigating into well-experienced persons. Consequently, formula is based on some estimation results [3]. Sum of these values is equal to Environment Factor, EF, which is calculated by following equation. Negative points in table indicate extra effort is needed to train team group as well as improving stability problems of project [8].

EF = 1.4 + (-0.03 * efactor) (3)

Table 2. Factors influencing completing project [3]

Factor	Description	Weight
E1	Familiar with the development process	1.5
E2	Application experience	0.5
E3	Object-oriented experience	1
E4	Lead analyst capability	0.5
E5	Motivation	1
E6	Stable requirements	2
E7	Part-time staff	-1
E8	Difficult programming language	-1

III. TIME ESTIMATION

In order to calculate time estimation for each leaf we can use time estimation tool for each use case since each node represent a goal as use case. Kerner suggested cost of 20 hour work per use case. Based on these definition and following equation, time is calculated for each singular leaf and for whole graph either [3].

Time = UCP * 20 (4)

But Ribu indicated these efforts range from 15 to 30 hours per use case [9] another approach designed by Schneider and Winter. In this method, they suggest to plus weights associated within factors e1-e6 which are more than 3 with weights associated within factors e7-e8 which are less than 3. If the sum becomes 2 or less than it, then 20 work hour per use case is considered. In cases the sum is equal to 3 or 4, as time estimation, 24 hour per work is considered. In cases sum exceeds, The project should be stopped until improving weak environmental factors or applying major changes in project [7, 8, 10] In another method 36 PH (personal hour) per UCP is proposed [8].

IV. USE CASE POINT IN GOAL GRAPH

There are many different strategies regarding eliciting and identifying goals. When all goals are identified and shown in graph form, based on their relation between goals and sub goals, we have different relationships. As you can see in figure 1, left picture (a), for achieving goal A, three sub-goal should be fired. Figure in middle (b), because of OR relations, shows if one of them is fired, root is fired as consequence. Also, firing some goals influence other goals either. Right graph (c) shows such an example. Firing goal B may affect firing goal A but not necessarily. We have another relations of goals. For clarify, firing some goals surely affect firing other goals, Firing some goals certainly negate and contradict achieving other goals. In another case, firing some goals may negatively affect achieving other goals but not necessarily. Here just for the first three type of dependency graphs are drawn [1].

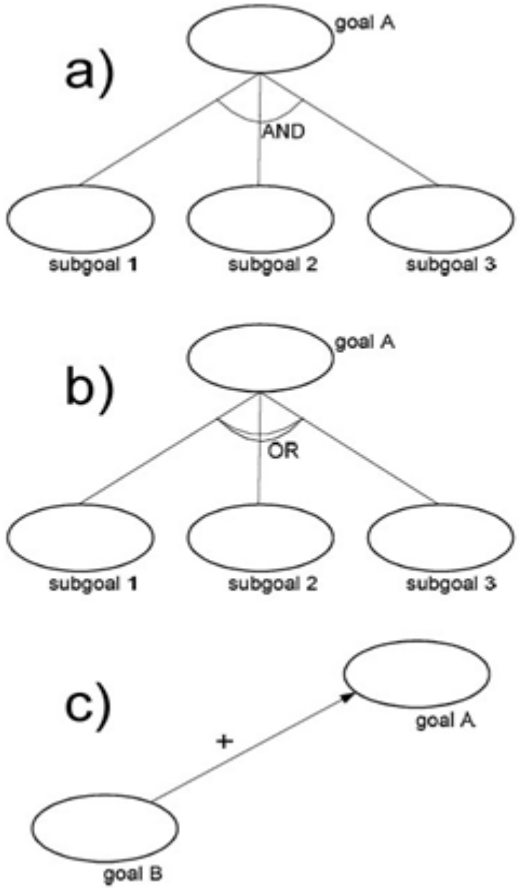

Figure 1. View of tree type of dependency [1]

In order to get a rough estimation of time and size of whole project, we need to calculate these values for root of goal graph since it is as an indicator of whole project. Based on previous sections we showed how use case point method works and calculate these values for use case. Relations between goals and use cases are showed. You can see brief steps of this method in table 3. We can consider graph as a max-min tree with slight difference. In max-min tree, max-min rows are ordered one by one but in this type of supposition tree, max-min nodes are scattered in tree. We take a bottom-to-up approach to traverse tree after calculating pairs of time and size for each node of graph. During calculation, it is worth mentioning type of dependency (And- OR, etc) between nodes and calculating contribution degree for each node. By applying use case point method on each node, size is estimated also time is calculated based on considering time estimation method. By doing this for all nodes, root values are gained at the end of calculations. It is notable that an epsilon amount is considered due to relative error. Right now, size and time values of root determine whole size and needed time. Figure 2 is sample of goal graph, pair of calculated size and time for each node.

Table 3. Steps of UCP method for each use case indicates a goal [3]

Step	Formula
Categorize use cases	Unadjusted Use Case Weights (UUCW) = Σ(#Use Cases *WF)
Categorize actors:	Unadjusted Actor Weights (UAW) = Σ(#Actors *WF)
Gain the Unadjusted Use Case Point (UUCP).	UUCP = UAW + UUCW
Tfactor is gained based on technical factors	(TCF) = 0.6 + (0.01 * tfactor)
Efactor is gained based on environmental factors	(EF) = 1.4 + (-0.03 * efactor)
Gain adjusted Use Case Points (UCP).	UCP = UUCP * TCF * EF
Estimate effort (E) in person per hour	E = UCP * (person per hour) per use case point

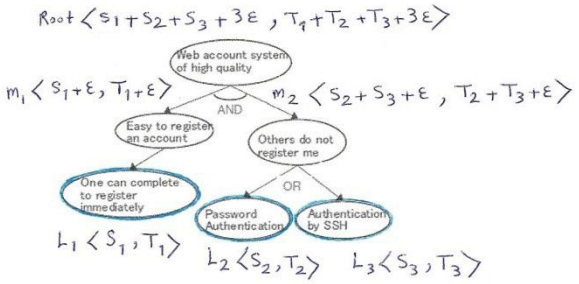

Figure 2. Sample of goal graph

V. ESTIMATION FEATURES WITH UCP

Use case point is an easy way to reach estimation because in usual mode it does not need many additional tool, any special programming languages or model. By and large, it is easy to learn, use, and understand. Different findings show it is an efficient method for large projects, specific context, incremental Development and also when development is outsourced [11]. Time and size Estimation process can be designed automatic through automating some part of use case management tools [7]. In UCP method, size and time estimation procedures are done two distinct steps [7]. On the other side, this method has many challenges. Use cases can be described at different levels of granularity and there is no guarantee of consistency among two different use cases. Then a standard during writing phase is needed. This feature becomes a problem when final written use cases are not comparable and provide poor basis for estimation [6]. Estimation process cannot be done until after writing and reflecting major changes in all use cases [7]. Writing use case points at the same level of detail and accuracy is one of the basis should be considered [4]. For instance, when software is incrementally developed, set of use case should have a level of details to estimate needed effort [11]. UCP approach contributes and estimates use cases, actors, etc, plus various requirements. Combination of these entities and attributes, causes nature and type of final use case uncertain and unknown. Another words, precise of this method depends on definition of many factors [6]. Assigning weight numbers to requirements needs precise knowledge and experience that is why two persons may hold different opinion about same requirement at the same time [6]. Sometimes, all requirement based on changes are not reflected in use cases, then using use case point method does not provide precise estimation, in these situation we may use additional methods [6].

VI. CONCLUSION

For any problem, there is a pool of solution but issue is that further research should be done to discover and bring them into reality. Use case points is just one of many methods of estimation. But many surveys and examinations showed this method has a potential to estimate what we want and bring practical advantages, however, it is more likely to develop and resolve challenges later. Moreover, using different technics for eliciting and drawing goal graph may cause slight changes in results and estimations. This happen because there are more than one technique to identify and elicit goals. All have their own characteristic and they differ from each other.

REFERENCES

[1] E. Kavakli, "*Modeling organizational goals: analysis of current methods*," presented at the Proceedings of the ACM symposium on Applied computing, 2004.

[2] B. Boehm, et al., "*Cost models for future software life cycle processes: COCOMO 2.0.* ," 1995.

[3] G. Karner, " *Metrics for Objectory ,", University of Linköping, Sweden, 1993.*

[4] *Alistair Cockburn,* " Writing Effective Use Cases," 2001.

[5] Daisuke Tanabe , et al., "*Supporting Requirements Change Management in Goal Oriented Analysis*," in 6th IEEE International Requirements Engineering Conference 2008.

[6] Joost Ouwerkerk and A. Alain, "*An Evaluation of the Design of Use Case Points (UCP).*"

[7] M. Cohn, "*Estimating With Use Case Points*," 2005.

[8] G. Banerjee, "*Use Case Points – An Estimation Approach* " 2001.
[9] R. Kirsten, "*estimating Object - Oriented sotware Proj ect s with Use Case* ," Master, university of Oslo, 2001.
[10] Jason P . Winter Ger i and Jason P . Wi nt er s. 1998. *Applying Use Cases: A P ractcal Guide . Addison Wesl ey.* , 2 ed., 2001.
[11] Parastoo Mohagheghi, et al., "*Effort Estimation of Use Cases for Incremental Large-Scale Software Development*," presented at the ICSE'05, 2005.

Amir Abbaszadeh Sori was born in Mazandaran, Iran, in 1990. He received the B.sc degree in Software Technology engineering from the Technical and Vocational University, Tehran, Iran, in 2012 and M.sc degree in Computer Networks from the Amirkabir University of Technology (Tehran Polytechnic). His current research interests include Computer Networks, Multimedia Software and Future Tech.

Mohammad Amini received the Master of Engineering in Information Technology majoring in Computer Networks at Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, 2013. His current research interest is in the field of Computer Networks, Future Internet and ICT.