

Fuzzy Logic as a Basic for Use Case Point Estimation

Jonathan Lee

Department of Computer Science and
Information Engineering
National Central University
Jhongli, Taiwan
Email: yjlee@selab.csie.ncu.edu.tw

Wen-Tin Lee

Department of Software Engineering
National Kaohsiung Normal University
Kaohsiung, Taiwan
Email: wtlee@nknknu.edu.tw

Jong-Yih Kuo

Department of Computer Science and
Information Engineering
National Taipei University of Technology
Taipei, Taiwan
Email: jykuo@ntut.edu.tw

Abstract—Project estimation based on function point or use case point (UCP) methods provide only fixed complexity grades which can not deal with the uncertain and imprecise conditions. This study, therefore, provides a fuzzy size estimation procedure for goal-driven use case model based on UCP using fuzzy theory. We propose a metric to calculate the unadjusted use case points of goal-driven use cases based on the relations between each use cases and goals with the fuzzy membership functions and the fuzzy rules. Furthermore, the technical and environmental factors are considered to calculate the use case points which can be used to estimate the implementation time and effort of the system under development. The proposed approach is illustrated by a benchmark problem domain of a meeting scheduler systems.

Index Terms—function point analysis, use case point, goal-driven use cases, fuzzy theory.

I. INTRODUCTION

Software estimation is an important and difficult activity in project planning and management process. Several estimation models have been proposed to estimate the software size [3], [5], [14] based on function point (FP) [1]. In object oriented development process, use case analysis is a crucial activity to develop software system. Therefore, the use case points method, a software sizing and estimation method based on use case counts called use case points (UCP) [6], [12], [13] is provided to estimate functional size of software in early stage of a project. The use case point method can be divided into five parts: Weighting actors by indicating the three actor types, simple, average and complex to generate unadjusted actor weights (UAW); Weighting use cases by indicating the three use case types based on transactions or classes to generate unadjusted use case weights (UUCW); Weighting technical complexity factors (TCF) to indicate the expectations of the users for the delivered software; Weighting environment factor (EF) to indicate the experience level of people on project; Finally, calculate use case points (UCP) by the formula $UCP = (UAW + UUCW) \times TCF \times EF$.

Nevertheless, FP and UCP methods provides only fixed complexity grades which can not deal with the uncertain and imprecise conditions. Therefore, some studies have been proposed to estimate the software size using fuzzy set theory. Xia et al [15] provide a neuro-fuzzy FP calibration model that integrate fuzzy logic, statistical regression and neuro

network techniques to improve the accuracy of software effort estimation. Braz and Vergilio [2] present two metrics, use case size points (USP) which considers the internal structure of the use case and fuzzy use case size point (FUSP) which uses fuzzy set theory for gradual change of complexity. Rodrigo et al [11] propose a task fuzzy estimator based on the UCP techniques with three inputs: complexity to modify the modules, experience of developers and the size of the alteration. The inference result is the effort in days for software development.

However, those studies do not consider the relationships between use cases and goals that would affect the complexity and implementation effort of each use cases. In this work, we propose a fuzzy software sizing and estimation method that considers the relationships between use cases and goals based on our previous work on goal-driven use case (GDUC) [7]–[10], in which use cases are derived based on the analysis of goals interactions. The meeting scheduler problem [4], a widely used benchmark problem domain in requirements engineering community, is adopted throughout this work as an example to illustrate the proposed approach.

The paper is organized as follows. Section 2 describes the concept of goal-driven use cases. Section 3 introduces the proposed method. Section 4 shows the results of the application of the introduced method. Section 5 concludes the paper.

II. GOAL-DRIVEN USE CASES (GDUC)

A brief summary of our goal-driven use case model [8] to construct a use case model with goals is outlined below for the readers' reference. To identify goals from domain descriptions and system requirements, we propose a faceted classification scheme so that each goal can be classified with three facets: *competence*, *view*, and *content*. The *competence* describes whether a requirement is satisfied completely or only to a degree. A *rigid* type of goal describes a minimum requirement that a target system must satisfy utterly. A *soft* goal describes properties or concerns that stakeholder care about for a target system, and can be satisfied to a degree. The *view* facet concerns whether a goal is *actor-specific* or *system-specific*. Actor-specific goals are actor's objectives in using a system; system-specific goals are requirements on services that the system provides. A goal can be further distinguished based

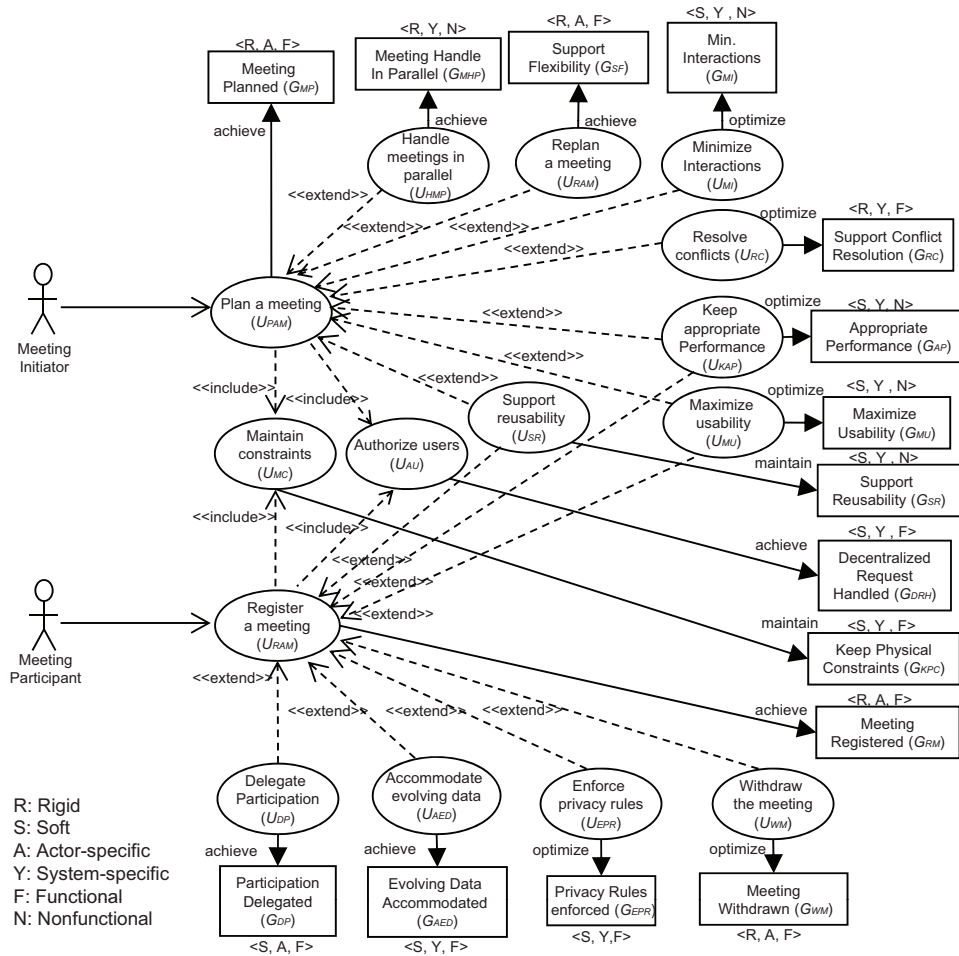


Fig. 1. Use Case Model for Meeting Scheduler System

on its *content* and can be either related to a system's *functional* aspects or associated with the system's *nonfunctional* aspect.

A goal can be achieved, optimized or maintained by its associated use case. The use cases of the meeting scheduler system are established for actors *meeting initiator* and *meeting participant* (see Figure 1). For the actor *meeting initiator*, the use case *plan a meeting* covers the scenario for an initiator to achieve an original goal *MeetingPlanned*, which is a goal of rigid, actor-specific, and functional. Meanwhile, in the case of the actor *meeting participant*, the use case *register a meeting* covers the case for a participant to achieve an original goal *MeetingRegistered*, which is a goal that is rigid, actor-specific, and functional.

To achieve a system-specific goal, an extension use case may be created. Referring to our example, the original use case *plan a meeting* describes the process to create a meeting from the view of the actor *initiator*. The extension use cases *Handle meetings in parallel* and *Resolve conflicts* extends it to take all initiators into account, that is, to achieve the system-specific goals *MeetingHandleInParallel* and *SupportConflictResolution*. The extension use cases *Accommodate evolving data* and *Enforce privacy rules* extends the original

use case *Register a meeting* to achieve the soft, system-specific and functional goals *EvolvingDataAccommodated* and *PrivacyRulesEnforced*, respectively.

To achieve a nonfunctional goal, an extension use case serves as a constraint to qualify its original use case. In our example, several constraints (may be rigid or soft) on a meeting are considered as extension use cases to extend or constrain the behavior of the original use case *Plan a meeting* which is a direct course to create a meeting, including *Minimize Interactions*, *Keep Appropriate Performance*, *Support Reusability* and *Maximize Usability*. To enhance reusability, the use case models are further elaborated by extracting the common fragments among various use cases into an included use case by using "include" relationships. For example, the use cases *Maintain constraints* and *Authorize users*, are included by the use case *Plan a meeting* and use case *Register a meeting*.

A. Evaluate Relation from Use Case to Goal

To analyze the satisfaction values from use cases to goals, we examine the relationships among goals and use cases in a pairwise manner, which results in the satisfaction values in Use Case to Goal U2G matrix (see U2G Matrix in Figure 2).

	U_{PAM}	U_{RAM}	U_{RC}	U_{MI}	U_{HMP}	U_{AU}	U_{MC}	U_{KAP}	U_{RM}	U_{DP}	U_{AED}	U_{WM}	U_{EPR}	U_{SR}	U_{MU}
G_{MP}	5	3	1	1	1	1	1	0	0	0	0	0	0	0	0
G_{SF}	3	5	2	-1	0	0	0	0	0	0	0	0	0	0	0
G_{RC}	2	2	5	-2	0	0	0	0	0	0	0	0	0	0	0
G_{MI}	2	-1	-2	5	0	0	0	0	0	0	0	0	0	0	0
G_{HMP}	2	0	0	0	5	0	0	0	0	0	0	0	0	0	0
G_{DRH}	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
G_{KPC}	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0
G_{AP}	1	0	0	0	0	0	0	5	1	0	0	0	0	-2	-1
G_{RM}	0	0	0	0	0	1	1	0	5	1	1	1	1	0	0
G_{DP}	0	0	0	0	0	0	0	0	2	5	1	0	0	0	0
G_{AED}	0	0	0	0	0	0	0	0	2	1	5	0	0	0	0
G_{WM}	0	0	0	0	0	0	0	0	2	0	0	5	0	0	0
G_{EPR}	0	0	0	0	0	0	0	0	2	0	0	0	5	0	0
G_{SR}	1	0	0	0	0	2	2	-2	1	0	0	0	0	5	-1
G_{MU}	1	0	0	0	0	0	0	-1	1	0	0	0	0	-1	5

Fig. 2. Satisfaction Values of Use Case to Goal (U2G) Matrix

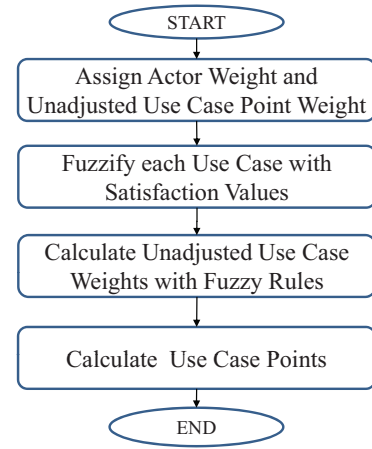


Fig. 3. Fuzzy Size Estimation Method

The gauge of satisfaction values is from 5 to -5. 5/-5 means that the goal is fully satisfied/fully denied after the use case is performed. 1/-1 means that the goal is partially satisfied/partially denied after the use case is performed. 0 means that the goal is not affected after the use cases is performed. Others represent the degrees between scores listed above. Detail explanation of the rating of satisfaction degree can be found in Table I.

TABLE I
RATINGS DEFINITION OF SATISFACTION VALUES

Score	Explanation
5	The goal is fully satisfied after the use case is performed
4	Represent the degrees between scores 5 and 3
3	The goal is largely satisfied after the use case is performed
2	Represent the degrees between scores 3 and 1
1	The goal is partially satisfied after the use case is performed
0	The goal is not affected after the use cases is performed
-1	The goal is partially denied after the use case is performed
-2	Represent the degrees between scores -1 and -3
-3	The goal is largely denied after the use case is performed
-4	Represent the degrees between scores -3 and -5
-5	The goal is fully denied after the use case is performed

For example, the effect of performing use case U_{PAM} is evaluated with respect to all goals in the system. G_{MP} is rated as 5 since performing U_{PAM} can fully satisfy the goal. G_{SF} is rated as 3 to indicate that performing U_{PAM} can largely satisfy the goal. The effect of performing use case U_{RC} with respect to goal G_{MI} is rated as -2 to show that performing U_{RC} can partially to largely deny the goal.

III. FUZZY SIZE ESTIMATION METHOD

In order to estimate the size and analyze the complexity of the goal-driven use case model, there are four steps involved in our fuzzy size estimation method (see Figure 3 for an overview). The method extends UCP method by considering the relations between use cases and goals with fuzzy theory. The details are explained below:

A. Assign Actor Weight and Unadjusted Use Case Point Weight

The unadjusted actor weights (UAW) and unadjusted use case weights (UUCW) are assigned base on UCP method with table II, III and IV [13]. Table II indicates the three actor types in UCP method. Table III and IV indicate the three use case types which are used to weight use cases based on transaction or class, respectively. The unadjusted use case weights (UUCW) is calculated by summing the weights of each use case.

TABLE II
ACTOR WEIGHTING FACTORS [13]

Actor Type	Description	Factor
Simple	Program interface	1
Average	Interactive, or protocol-driven, interface	2
Complex	Graphical interface	3

TABLE III
TRANSACTION-BASED WEIGHTING FACTORS [13]

Use Case Type	Description	Factor
Simple	3 or fewer transactions	5
Average	4 to 7 transactions	10
Complex	More than 7 transactions	15

TABLE IV
ANALYSIS CLASS-BASED WEIGHTING FACTORS [13]

Use Case Type	Description	Factor
Simple	Fewer than 5 analysis classes	5
Average	5 to 10 analysis classes	10
Complex	More than 10 analysis classes	15

B. Fuzzify each use case with satisfaction values

Since each use case satisfies each goals in different degree, $rValue$, represents the relevance from a use case to the goals of the system, can be calculated using equation 1 according to the satisfaction values in U2G matrix (see figure I).

U_iG_j , the satisfaction values in column i of $U2G$ matrix, represents the relationships between use case U_i and the goals of the system. Equation 1 generates $rValue$ of use case U_i by finding the maximum value of U_iG_j except the satisfaction value of use case U_i to its goal G_i and adding the average of the rest satisfaction values of U_iG_j where $U_iG_j \neq 0$. That is, except goal G_i of use case U_i , the satisfaction value of most complex goal of use case U_i should achieve is regarded and the average of the sum of the rest satisfaction values which are not zero is added to indicate the degree that the use case should satisfy the goals. The result generated in equation 1 denotes the relevance between use case U_i and the goals of the system.

$$rValue_i = \max(|U_iG_j|) + \text{avg}(\sum |U_iG_j| - \max(|U_iG_j|))$$

where $n = \#$ of goals, $i = 1 \dots n$, $j = 1 \dots n$, $i \neq j$, $U_iG_j \neq 0$ (1)

To allow a gradual change from one complexity category to another, we define the fuzzy input membership function $F(x)$ that classify $rValue$ which is between $0 \sim 10$ into five linguistic terms. (see figure 4).

$$F(x) = \begin{cases} \frac{2.5 - x}{2.5} & 0 \leq x < 2.5 & \text{Simple} \\ \frac{x}{2.5} & 0 \leq x < 2.5 & \text{Average} \\ \frac{5 - x}{2.5} & 2.5 \leq x < 5 & \text{Average} \\ \frac{x - 2.5}{2.5} & 2.5 \leq x < 5 & \text{Complex} \\ \frac{7.5 - x}{2.5} & 5 \leq x < 7.5 & \text{Complex} \\ \frac{x - 5}{2.5} & 5 \leq x < 7.5 & \text{Very Complex} \\ \frac{10 - x}{2.5} & 7.5 \leq x < 10 & \text{Very Complex} \\ \frac{x - 7.5}{2.5} & 7.5 \leq x < 10 & \text{Most Complex} \end{cases}$$

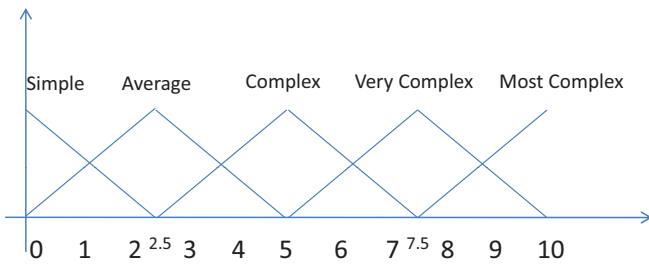


Fig. 4. The Membership Function $F(x)$ of $rValue$

C. Calculate Unadjusted Use Case Weights with Fuzzy Rules

Table V defines the fuzzy rules with the use case types and the membership function $F(x)$. For each use case, we use min-max fuzzy inference method to match the rules with its use case types and the result of $F(rValue)$. The center of gravity defuzzification is made through the application of the rules matched with the output membership function. (see figure 5).

After this, the fuzzy unadjusted use case weight (FUUCW) of each use case can be calculated.

TABLE V
FUZZY RULES

use case type / $F(x)$	Simple	Average	Complex	Very Complex	Most Complex
Simple	Simplest	Very Simple	Simple	Average	Complex
Average	Very Simple	Simple	Average	Complex	Very Complex
Complex	Simple	Average	Complex	Very Complex	Most Complex

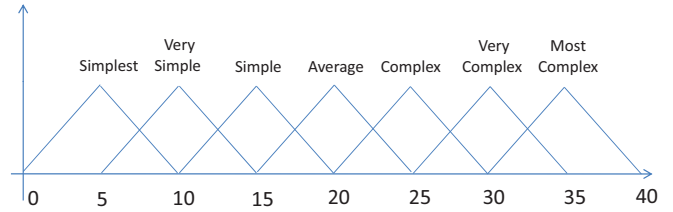


Fig. 5. The Membership Function of Fuzzy Rules

D. Calculate Use Case Points

We adopt the technical complexity factors (TCF) and environment factors (EF) [13] to take the characteristics of the system-to-be into count to generate use case points. Table VI shows the technical complexity factors which indicate the characteristics of the delivered software. The TFactor is calculated by adding all the numbers resulting from multiplying the value of each technical factors (T1-T13) by its weight. $TFactor = \sum_i (Ti \times \text{Weight})$, where $i = 1, \dots, 13$. The following formula is then applied to get the TCF value: $TCF = 0.6 + (0.01 \times TFactor)$

TABLE VI
WEIGHTING TECHNICAL COMPLEXITY FACTORS (TCF) [13]

Factor number	Factor Description	Weight
T1	Distributed system	2
T2	Response or throughput	1
T3	End-user efficient	1
T4	Complex internal processing	1
T5	Code must be reusable	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent	1
T11	Includes special security feature	1
T12	Access for third parties	1
T13	Special user training facilities required	1

Table VII [13] shows the 8 environment factors which indicate the experience level of people on project. The EFactor is calculated by adding all the numbers resulting from multiplying the value of each environment factors (F1-F8) by its weight.

$$EFactor = \sum_i (Fi \times \text{Weight}) , \text{ where } i = 1, \dots, 8$$

The following formula is then applied to get the EF value:
 $EF = 1.4 + (-0.03 \times \text{EFactor})$.

TABLE VII
WEIGHTING ENVIRONMENT FACTOR (EF) [13]

Factor number	Factor Description	Weight
F1	Familiar with RUP	1.5
F2	Application experience	0.5
F3	OO experience	1
F4	Lead analyst capability	0.5
F5	Motivation	1
F6	Stable requirements	2
F7	Part-time workers	-1
F8	Difficult programming language	-1

The value of unadjusted use case points (UUCP) is calculated by adding unadjusted actor weights (UAW) and unadjusted use case weights (UUCW) in UCP method. In this study, we use fuzzy unadjusted use case weights (FUUCW) to replace UUCW (see equation 2).

$$UUCP = UAW + FUUCW \quad (2)$$

The use case points of the system-to-be is generated by considering the technical complexity factors and environment factors using equation 3.

$$\text{Use Case Points} = UUCP \times TCF \times EF \quad (3)$$

IV. CASE STUDY

The meeting scheduler system is used to explain how our fuzzy size estimation method be applied. The detail steps are illustrated below.

A. Step 1: Assign Actor Weight and Unadjusted Use Case Point Weight

TABLE VIII
ACTOR WEIGHTS OF MEETING SCHEDULER SYSTEM

Actor	Actor Type	UAW
Meeting Initiator	complex	3
Meeting Participant	complex	3

Table VIII represents the unadjusted actor weights (UAW) of meeting scheduler system. The total UAW is calculated as:
Total UAW = 2 complex = $2 \times 3 = 6$

Table IX represents the unadjusted use case weights (UUCW) of meeting scheduler system. The total UUCW is calculated as:

$$\text{Total UUCW} = 6 \text{ simple} + 6 \text{ average} + 3 \text{ complex} = 6 \times 5 + 6 \times 10 + 3 \times 15 = 30 + 60 + 45 = 135$$

B. Step 2: Fuzzify each Use Case with satisfaction value

Taking U_{PAM} as an example, we use the satisfaction values in the first column of table 2 to calculate $rValue$ of U_{PAM} below.

$$rValue = \max(|U_{PAM}G_j|) + \text{avg}(\sum |U_{PAM}G_j| - \max(|U_{PAM}G_j|)) \text{ where } n = \# \text{ of goals, } j = 1 \dots n$$

$$= 3 + 1.5 = 4.5$$

The fuzzy input membership function then takes 4.5 as input to classify the relevance of U_{PAM} to the goals of meeting scheduler system into 0.2 average and 0.8 complex (see figure 6)

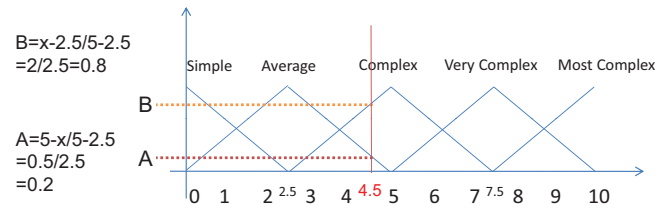


Fig. 6. U_{PAM} Example of Fuzzy Input Membership Function

C. Step 3: Calculate Unadjusted Use Case Weights with Fuzzy Rules

The fuzzy unadjusted use case weight (FUUCW) can be calculated by using center of gravity defuzzification with the output membership function of fuzzy rules. Since the complexity of U_{PAM} is complex and the results of the input membership function of $rValue$ are 0.2 average and 0.8 complex, the rules matched is 0.2 complex and 0.8 very complex. We use the output membership function of fuzzy rules and center of gravity defuzzification to calculate the FUUCW which is 23.79 (see figure 7).

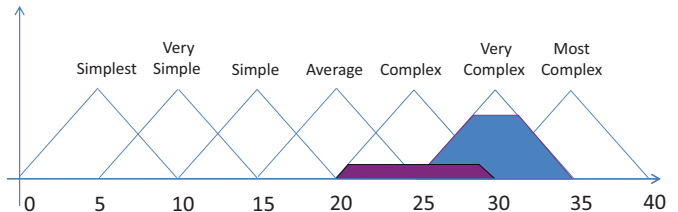


Fig. 7. U_{PAM} Example of Fuzzy Output Membership Function

Table IX represents the unadjusted use case points of meeting scheduler system.

TABLE IX
UNADJUSTED USE CASE POINTS OF MEETING SCHEDULER SYSTEM

Use Case	Use Case Type	UUCW	$rValue$	FUUCW
U_{PAM}	complex	15	4.5	23.79
U_{RAM}	simple	5	4.5	13.79
U_{RC}	complex	15	3.5	22.10
U_{MI}	average	10	3.0	16.21
U_{HMP}	simple	5	1.0	7.10
U_{AU}	average	10	3.0	16.21
U_{MC}	average	10	3.0	16.21
U_{KAP}	simple	10	3.0	16.21
U_{RM}	complex	15	3.5	22.10
U_{DP}	average	10	2.0	13.79
U_{AED}	average	10	2.0	13.79
U_{WM}	simple	5	1.0	7.10
U_{EPR}	simple	5	1.0	7.10
U_{SR}	simple	5	3.0	11.21
U_{MU}	simple	5	2.0	8.79
Total		135	40.0	215.50

D. Step 4: Calculate Use Case Points

To calculate the use case points of meeting scheduler system, we should consider TCF and EF of the system-to-be to adjust the overall complexity. Table X represents the technical complexity weights of meeting scheduler system. The TFactor is calculated as:

$$\text{TFactor} = 0+3+3+3+3+0.5+2.5+0+3+3+1+0+3 = 25$$

The TCF value is calculated as:

$$\text{TCF} = 0.6 + (0.01 \times \text{TFactor}) = 0.6 + (0.01 \times 25) = 0.85$$

TABLE X

TECHNICAL COMPLEXITY FACTORS OF MEETING SCHEDULER SYSTEM

Factor number	Weight	Assigned Value	Extended Value	Reason
T1	2	0	0	Not at first release
T2	1	3	3	Time response is one of the important criteria
T3	1	3	3	Needs to be efficient
T4	1	3	3	Needs to resolve conflicts
T5	1	3	3	Some code should be reused
T6	0.5	1	0.5	End users can use it through browser
T7	0.5	5	2.5	Needs to be easy for non-technical people
T8	2	0	0	Not at this time
T9	1	3	3	Yes
T10	1	3	3	Handle meeting in parallel
T11	1	1	1	Simple security
T12	1	0	0	No
T13	1	3	3	Require some training

Table XI represents the environment complexity weights of meeting scheduler system. The EFactor is calculated as:

$$\text{EFactor} = 3+1.5+3+2.5+5+6+0+(-1) = 20$$

The EF value is calculated as:

$$\text{EF} = 1.4 + (-0.03 \times \text{EFactor}) = 1.4 + (-0.03 \times 20) = 0.8$$

TABLE XI

ENVIRONMENT FACTORS OF MEETING SCHEDULER SYSTEM

Factor number	Weight	Assigned Value	Extended Value	Reason
F1	1.5	2	3	Some of team familiar
F2	0.5	3	1.5	Some of team familiar
F3	1	3	3	Some of team familiar
F4	0.5	5	2.5	Team is really good
F5	1	5	5	Team is really eager
F6	2	3	6	Changes may occur
F7	-1	0	0	No part-timers
F8	-1	1	-1	We are looking at Java

Calculated by the equation 3, the total use case points of meeting scheduler system = $(6+215.5) \times 0.85 \times 0.8 = 150.62$

V. CONCLUSION

This article proposes a fuzzy size estimation method of goal-driven use cases using fuzzy theory by extending the use case point analysis method with the relations between use cases and goals. We evaluate and analyze the relations from use cases to goals to generate the relevance value which can be used as inputs of the fuzzy input membership function. The fuzzy unadjusted use case weight (FUUCW) can be calculated using center of gravity defuzzification with the matched fuzzy rules

and output membership function. FUUCW not only considers the internal functionality of a use case but also its relations with other use cases and goals, which indicates the complexity of each use case. The total use case points shows the overall system size by considering the technical complexity factors and environment factors of the system under consideration.

Our future research plan will focus on applying our fuzzy size estimation method in more case studies and collecting historical data in the historical database for regression analysis.

REFERENCES

- [1] A. Albrecht. Measuring Application Development Productivity. In *IBM Application Development Symp.*, pages 83–92, October 1979.
- [2] M. R. Braz and S. R. Vergilio. Software effort estimation based on use cases. In *Proceedings of the 30th Annual International Computer Software and Applications Conference*, pages 221–228, 2006.
- [3] COSMIC. *COSMIC-FPP measurement manual - The COSMIC implementation guide for ISO/IEC 19761:2003, version 2.2*. Common Software Measurement International Consortium (COSMIC), 2003.
- [4] M. S. Feather, S. Fickas, A. Finkelstein, and A. van Lamsweerde. Requirements and specifications exemplars. *Automated Software Engineering*, 4(4):419–438, October 1997.
- [5] IFPUG. *Function point counting practices manual, Release 4.2*. International Function Point Users Group (IFPUG), Princeton Junction, USA, 2004.
- [6] G. Karner. Resource Estimation for Objectory Projects. *Objectory Systems*, 1993.
- [7] J. Lee and K.-H. Hsu. Modeling software architectures with goals in virtual university environment. *Information and Software Technology*, 44(6):361–380, April 2002.
- [8] J. Lee and N.-L. Xue. Analyzing user requirements by use cases: A goal-driven approach. *IEEE Software*, 16(4):92–101, July/August 1999.
- [9] J. Lee, N.-L. Xue, and J.-Y. Kuo. Structuring requirements specifications with goals. *Information and Software Technology*, 43(2):121–135, February 2001.
- [10] W.-T. Lee, W.-Y. Deng, J. Lee, and S.-J. Lee. Change impact analysis with a goal-driven traceability-based approach. *International Journal of Intelligent Systems*, 25(8):878–908, August 2010.
- [11] R. P. Pantoni, E. A. Mossin, and D. Brandao. Task effort fuzzy estimator for software development. *INFOCOMP Journal of Computer Science*, 7(2):84–89, 2008.
- [12] K. Ribu. Estimating object-oriented software projects with use cases. Master's thesis, November 2001.
- [13] G. Schneider and J. P. Winters. *Applying Use Cases: A Practical Guide (2nd Edition)*. Addison-Wesley, 2001.
- [14] UKSMA. *Mk II Function Point Analysis, Counting Practices Manual, version 1.31*. United Kingdom Software Metrics Association (UKSMA), 1998.
- [15] W. Xia, L. F. Capretz, D. Ho, and F. Ahmed. A new calibration for function point complexity weights. *Information and Software Technology*, 50:670–683, June 2008.