A Light-Weight Incremental Effort Estimation Model for Use Case Driven Projects

Kan Qi
University of Southern California
USA
kqi@usc.edu

Barry W. Boehm
University of Southern California
USA
boehm@usc.edu

***Abstract*—Use case analysis has been widely adopted in modern software engineering due to its strength in capturing the functional requirements of a system. It is often done with a UML use case model that formalizes the interactions between actors and a system in the requirements elicitation iteration, and with architectural alternatives explored and user interface details specified in the following analysis and design iteration. On the other hand, to better support decision making in software management, effort estimation models are required to provide estimates about the required project effort at the very early stage of a project, which, however, provides little information for accurately evaluating system complexity. To solve this dilemma, an incremental approach of integrating information available throughout the early iterations to provide multiple effort estimations is preferred in keeping the balance between utility and accuracy. In this paper, we proposed an effort estimation model that incorporates two sub-models to provide two points of effort estimation during the early iterations of a use case driven project. Our proposed model is lightweight due to the fact that its size metrics are defined to be countable directly from the artifacts of the early iterations. To better calibrate the model, especially in considering the situation of having limited data points available, we also introduced a normalization framework in our model calibration process to reduce noise from the effort data. By calibrating the proposed sub-models with the data points collected from 4 historical projects, we demonstrated that the sub-models fit the data set well, and the later-phase model is superior to the early-phase model for it fits the data set better and shows less uncertainty in the calibrated parameters.**

***Index Terms*—effort estimation, use case analysis, use case driven process, model calibration, data normalization, software size metrics, model-based analysis, incremental estimation model, use case points, function points, unified modeling language, object-oriented modeling, project management, automated analysis**

I. Introduction

Effort estimation models have been playing an important role in making software management decisions. For example, with project effort estimated, one can further estimate the schedule or the cost for a project to avoid the risks of being over schedule or budget [1]. To ensure its utility in decision making, it is necessary for an effort estimation model to provide estimates at the early stage of a project, which, however, provides little information for accurately evaluating system complexity. As is often the case, system designs are gradually made more detailed as a project proceeds and the more detailed designs provide more information to size a project. In the context of use case driven development, system behavior is incrementally elaborated by analysis and design activities [2]–[4]. With the information revealed in these activities, we are able to define more fine grained size metrics to better estimate the system complexity, which can further be used as predictor variables to estimate project effort. The well-known example of incrementally integrating information for effort estimation is the COCOMO II's early design model and post architecture model [1], [5], which provides two points of estimation throughout a project and promises better accuracy in the latter. In comparison with the COCOMO II's approach that uses delivered source instructions (DSI) as the size metric, we defined the size metrics based on the artifacts of a use case driven project.

Requirements elicitation has been a longstanding topic for software engineering, and use case analysis has got widely adopted due to its strength in capturing the functional requirements. To be specific, use cases can be formally written as use case narratives to describe the interactions between actors and a system. A use case narrative includes the name or id of the use case, a brief description, pre-conditions, post-conditions, basic flow, and alternative flows. The basic flow of events models the interactions between an actor and a system for the sunny day scenario, and the alternative flows are created to model the rainy day scenarios. The logical interconnection of the basic flow and the alternative flows can be represented by structured scenarios. Each event of the basic flow is indexed by a step number, and each alternative flow is separated from the basic flow at certain step to represent a rainy day scenario or exceptional scenario, and then rejoins the basic flow after taking actions on the alternative conditions. An example of structured scenario is given in TABLE I. In the context of Model Based Engineering (MBE), use cases are captured with a UML use case model [4], [6], [7], and can further be described in detail by use case diagrams, robustness diagrams, sequence diagrams, activity diagrams, and class diagrams. Also, the structured scenarios can be automatically converted into activity diagrams by CASE[1]tools, for example, Enterprise Architect [6].

In our incremental approach of effort estimation, we followed the typical deliverable model of the use case driven

[1]CASE: Computer-aided software engineering

TABLE I
AN EXAMPLE OF STRUCTURED SCENARIO

Step	Action of Basic Path	
1	System displays welcome screen	
2	User inserts a debit card	
3	System prompts for PIN	
4	User enters correct PIN	
5	System authenticates user and logs access info	
6	System prompts options	
7	User selects balance option	
8	System displays current account balance	
9	System displays welcome screen after 3 seconds	
Step	Alternative Path	Join
2a	Invalid Card	1
4a	Incorrect PIN	1

process, which is depicted in Fig. 1 [8]. The proposed size metrics rely on the deliverables of the requirements elicitation iteration (iteration II) and the analysis and design iteration (iteration III). For iteration II, we defined a size metric called Early Use Case Points (EUCP) that weights each use case with the number of scenarios identified from the structured scenarios, and uses the sum of the weights to calculate the total Unadjusted Early Use Case Weight (UEUCW). With the calculated UEUCW and the ratings for Unadjusted Actor Weight (UAW), Technical Complexity Factor (TCF), and Environmental Factor (EF), which are defined in the original Use Case Points (UCP) definition, EUCP is calculated and used to calibrate a linear model with project effort data to provide a first-order estimation of project effort. We will denote the sub-model for this iteration as $Model_{EUCP}$ or $Model_I$ in the rest of the paper. In iteration III, architectural solutions are introduced to implement the required functions, and also user interface specifications are largely captured by UI prototypes or storyboards [2], [4], [8]. We specifically take into consideration the UI complexity when defining the size metric at this iteration, due to the fact that UI design and refinement have become a major source of project effort for application and information system development. More specifically we defined a size metric called Extended Use Case Points (EXUCP) for this iteration, which uses the number of domain elements and UI elements that each identified transaction interacts with to reflect its internal complexity. Using EXUCP as the size metric, we calibrated the second linear model ($Model_{EXUCP}$) to provide effort estimation for better accuracy ($Model_{EXUCP}$ is also denoted as $Model_{II}$ for simplicity). To summarize, the major contributions we made in this paper are as follows:

1) Proposed an effective size metric called Early Use Case Points (EUCP) as the early stage size metric.
2) Proposed a more precise size metric called Extended Use Case Points (EXUCP) as the later stage size metric, which considers the structural complexity and the UI complexity of the identified transactions.

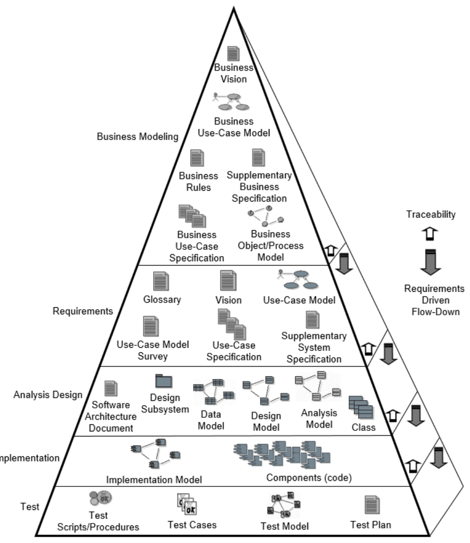

Fig. 1. Use Case Driven Flow-Down Chart

3) Proposed a normalization framework to reduce the unexplainable portion of variation within effort data.
4) Conducted an empirical study on 4 historical projects to demonstrate that the linear relationships exist between the proposed size metrics and project effort, and also $Model_{II}$ is superior to $Model_I$.

The rest of the paper is structured into 7 sections. In the following section, we introduced several related studies that are fundamental to our study, and also discussed the reasons why some related works are inadequate to provide the values our approach promises. In section III, we first summarized the principles that we followed in order to deliver the promised values, and then introduced the size metrics and their counting methods. In section IV, a normalization framework based on COCOMO II is introduced to eliminate the portion of variation from the effort data, which is attributable to the unmodeled factors. In section V, the model calibration process is introduced through an empirical study of four student projects, and the proposed effort estimation model is calibrated through ordinary least squares linear regression. In section VI, the quality of the model is assessed with respect to goodness of fit and significance of the estimated parameters. In section VII, we summarized our approaches in dealing with the challenges, applicability of our approach and the calibrated models, and also the future directions of this study.

II. RELATED WORKS

To achieve the goal of estimating project effort at the early stage of a project, parametric effort estimation models use software size and other relevant aspects of a project to model the sources of project effort, and the effects of the factors are calibrated through empirical data, a well-known example of which is COCOMO II [1]. An accurate effort estimation by COCOMO II requires an accurate estimation of software size in terms of Delivered Source Instructions (DSI). However, as the common practice of a use case driven approach suggests, it's more focused on modeling the system behavior at an abstract level at the early stage of a project [3], [4], [8], which makes it difficult for use case driven projects to gather information about software size in terms of DSI.

There also exist several size metrics focused on measuring software size by analyzing system behavior. For example, Use Case Points (UCP), first conceived by Gustav Karner, measures software size through use cases [9], and Function Point (FP) analysis, developed by Allan Albrecht, computes functional size measurement (FSM) of software [10], [11]. However, their proposed size metrics are either at high level that doesn't reflect system complexity with the required precision to guarantee effort estimation accuracy, or at low level that requires substantial effort to gather information about data functions and transaction functions in the counting process. To deal with this dilemma, an incremental approach of integrating information that is gradually made available at different phases provides a good balance between utility and accuracy. This follows our idea of extending the existing UCP technique to accommodate the information that is available at different phases of use case driven projects.

Several studies of extending the existing UCP counting method have been proposed to improve use case classification accuracy. For example, Periyasamy and Ghode proposed an extended version of UCP called e-UCP [12] that redefined use case and actor classifications and their associated weights by taking into account the information about input parameters, output parameters, pre-condition, post-condition, and successful and exceptional scenarios within use case narratives when measuring software size. Another revision of UCP is done by Mudasir Manzoor Kirmani and Abdul Wahid called Re-UCP [13], in which one extra rating level - "critical"- is added in both the use case and actor weighting schemes, and "scalability" and "project methodology" are added as the 14th technical complexity factor and 9th environmental factor. These methods of customizing the original UCP technique are majorly for the purpose of improving use case classification accuracy. However, in addition to accuracy, we also aimed to improve efficiency of the counting process to avoid investing too much effort in gathering information for size estimation.

A practical difficulty prevents one from creating accurate effort estimation models is the noise within effort data, which is because the effort data is usually collected from high dimensional uncontrolled environments. Noise may take the leading effect on the calibrated parameters especially when available data points are limited. To deal with the situation, measures of identifying the sources of noise and removing the unexplainable variation from effort data are required [14], [15]. In this paper, we proposed a normalization framework to eliminate the influences from the un-modeled factors whose effects on the effort, however, have been modeled by COCOMO II. By calibrating the models using the normalized effort data, we are able to reveal more plausible relationships between our proposed metrics and project effort.

III. MODEL DEFINITION

A. Design Principles

To ensure the values our proposed effort estimation models promise, we followed the following design principles:

1) Taking size measurements for effort estimation at the early stage of the process ensures utility of the estimates: As described in Fig. 1, through iteration II and iteration III, functional requirements, UI specifications, and architectural solutions are well understood. These two iterations usually happen at the early stage of a project, and largely define the scope of the entire project. For this reason, we focused on taking size measurements from these two iterations of a use case driven project.

2) Incrementally integrating information available throughout the iterations increases the precision in measuring system complexity: In iteration II, EUCP measures use case complexity by the number of scenarios identified from structured scenarios. While, in iteration III, EXUCP takes into consideration domain elements and UI components to reflect the internal complexity of each identified transaction.

3) Having the size metrics defined to be countable directly from the artifacts improves efficiency of the counting process: To avoid the situation of investing too much effort in buying information for effort estimation, EUCP and EXUCP are directly countable from the artifacts of iteration II and iteration III. This makes it possible to develop automated counting techniques to measure software size. Semi-automated approaches are suggested in the section of introducing the counting methods. Complete automation is also possible if requirements analysis and modeling activities are all done with a CASE tool.

B. Size Metrics and Counting Methods

1) Early Use Case Points: Early Use Case Points reuses the weighting schemes of UCP for unadjusted actor weight (UAW), technical complexity factor (TCF) and environmental factor (EF), however, it adopts a different approach of measuring use case complexity. Specifically, the number of transactions of a use case is counted as the number of scenarios, which can also be automatically computed as the cyclomatic complexity of the activity diagram converted from the structured scenarios. Based on the number of scenarios, a weight is assigned to the use case to reflect its complexity. The steps of counting EUCP is introduced below:

1) Extract structured scenarios from use case narratives.
2) Convert structured scenarios to activity diagrams. Step 1 and step 2 can be automated, as introduced in [6] with Enterprise Architect.
3) Count the cyclomatic complexity (CC) for the converted activity diagrams. This step can either be manually conducted or automated, for example, using a script to parse the activity diagrams with Enterprise Architect.
4) Determine the use case complexity for each identified use case according to TABLE II and calculate the Unadjusted Early Use Case Weight (UEUCW) by (1),

TABLE II
UNADJUSTED EARLY USE CASE WEIGHT

Use Case Complexity	CC	Weight
Simple	1-3	5
Average	4-7	10
Complex	≥8	15

TABLE III
UNADJUSTED ACTOR WEIGHT

Classification	Type of Actor	UAW
Simple	External system that must interact with the system using a well-defined API	1
Average	External system that must interact with the system using standard communication protocols (e.g. TCP/IP, FTP, HTTP, database)	2
Complex	Human actor using a GUI application interface	3

where C_{simple}, $C_{average}$, and $C_{complex}$ represent the sets of use cases that are determined as simple, average, and complex respectively.

$$UEUCW = 5 * C_{simple} + 10 * C_{average} + 15 * C_{complex} \quad (1)$$

5) Calculate Unadjusted Actor Weight (UAW) by (2) based on the weights assigned to the identified actors. The weight for each identified actor is determined by its type according to TABLE III. Here we reused the actor classification weights from the original Use Case Points definition [16].

$$UAW = 1 * A_{simple} + 2 * A_{average} + 3 * A_{complex} \quad (2)$$

6) Calculate the Technical Complexity Factor (TCF) based on the sum of the impact (denoted as TFactor) of the 13 technical factors given TABLE IV by (3). Here we reused the technical factor weighting scheme and the impact calculation method from the original Use Case Points definition [16].

$$TCF = 0.6 + (TFactor/100) \quad (3)$$

7) Evaluate the Environmental Factor (EF) based on the sum of the impact (denoted as EFactor) of the 8 environment factors given TABLE V by (4). Here we reused the environmental factor weighting scheme and the impact calculation method from the original Use Case Points definition [16].

$$EF = 1.4 + (-0.03 * EFactor) \quad (4)$$

8) Calculate the adjusted Early Use Case Points (EUCP) by (5).

$$EUCP = (UEUCW + UAW) * TCF * EF \quad (5)$$

TABLE IV
TECHNICAL COMPLEXITY FACTOR

Factor	Description	Weight
T1	Distributed system	2
T2	Response time/performance objectives	1
T3	End-user efficiency	1
T4	Internal processing complexity	1
T5	Code reusability	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portability to other platforms	2
T9	System maintenance	1
T10	Concurrent/parallel processing	1
T11	Security features	1
T12	Access for third parties	1
T13	Special user training facilities	1

TABLE V
ENVIRONMENTAL FACTOR

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

2) Extended Use Case Points: In iteration III, as architectural alternatives are explored and UI specifications are captured using UI prototypes or storyboards, we are able to identify the domain elements (DE) and UI elements (UIE) involved in the transactions of a use case, and use the number of DE and the number of UIE to reflect a transaction's internal complexity. To be specific, domain elements are the real-world concepts needed to be modeled in the system, which incorporate both behavior and data to realize the system transactions, and the number of DE a transaction involves measures the number of system components a developer needs to deal with when implementing a transaction. Also the number of UIE a transaction involves is used to reflect its independent effect on the effort for implementing the transaction. These numbers can be counted from sequence diagrams, or from a more concise presentation - robustness diagrams. Usually one use case is modeled by one sequence diagram or one robustness diagram [4], [6]. In the case of using robustness diagrams to model use cases, the method to count EXUCP is introduced below:

1) Identify the transactions from the robustness diagram that models a use case. This process can be automated with a script to count the independent paths from the XML files exported from CASE tools, for example, Enterprise Architect. Each identified independent path represents a transaction.

TABLE VI
TRANSACTIONAL COMPLEXITY LEVELS BY EXUCP

UIE\DE	1-3	4-7	≥ 8
0-1	Low	Low	Medium
2-5	Low	Medium	High
≥ 6	Medium	High	High

TABLE VII
UNADJUSTED TRANSACTION WEIGHT

Complexity Level	UEXTW
Low	1
Medium	2
High	5

2) Count the number of UI elements (UIE) involved within each identified transaction. In the case of using robustness diagrams to model use cases, it is the number of identifiable user interface components within the "boundary" elements of a path representing a transaction.
3) Count the number of domain elements (DE) involved within each identified transaction. In the case of using robustness diagrams to model use cases, it is the number of the nodes on a path that represents a transaction.
4) Evaluate the complexity level for each transaction by UIE and DE according to TABLE VI.
5) Assign Unadjusted Transaction Weight (UTW) for each transaction according to the complexity level evaluated at step 4 and TABLE VII.
6) Calculate Unadjusted Extended Use Case Weight (UEXUCW) by (6), where C denotes the set of identified use cases and T_c represents the set of identified transactions for a use case $c \in C$.

$$UEXUCW = \sum_{c \in C} \sum_{t \in T_c} UTW(t) \tag{6}$$

7) Reuse the evaluations for technical complexity factor (TCF) and environmental factor (EF) from $Model_I$ to calculate EXUCP by (7).

$$EXUCP = (UEXUCW + UAW) * TCF * EF \tag{7}$$

TABLE VIII
FACTOR COVERAGE BY UCP OVER COCOMO II

COCOMO II Cost Drivers	UCP Complexity Factors	Explained by $Model_I$ or $Model_{II}$?
PREC[2]		No
RESL[3]		No
PMAT[4]	E1	Yes
TEAM[5]	E5	Yes
FLEX[6]		No
RELY[7]		No
DATA[8]		No
CPLX[9]	T4	Yes
RUSE[10]	T5	Yes
DOCU[11]		No
TIME[12]	T2	Yes
STOR[13]		No
PVOL[14]		No
ACAP[15]	E4	Yes
PCAP[16]	E3	Yes
PCON[17]	E7	Yes
APEX[18]	E2	Yes
PLEX[19]		No
LTEX[20]	E8	Yes
TOOL[21]		No
SITE[22]		No
SCED[23]		No

IV. EFFORT DATA NORMALIZATION

Due to the fact the effort data of software projects are usually collected under high dimensional uncontrolled environments, a portion of the variation within effort data may come from certain environmental factors that are not considered by the models. A portion of the variation may be eliminated with appropriate modeling. This aspect of the effort data becomes more important to consider especially when one doesn't have sufficient data points to rely on to eliminate the outliers. With that being said, the essential goal for this section is to eliminate the portion of the variation in effort data, which is caused by the factors that are not explicitly modeled.

A. *Identify Un-modeled Factors*

To identify the sources of influence on project effort, we used COCOMO II's cost drivers, for the reason that the 5 scale factors and the 17 effort multipliers COCOMO II incorporates

[2]PREC: Precedentedness
[3]RESL: Architecture/Risk Resolution
[4]PMAT: Process Maturity
[5]TEAM: Team Cohesion
[6]FLEX: Development Flexibility
[7]RELY: Required Software Reliability
[8]DATA: Database Size
[9]CPLX: Product Complexity
[10]RUSE: Developed for Reusability
[11]DOCU: Documentation Match to Life-Cycle Needs
[12]TIME: Execution Time Constraint
[13]PLEX: Platform Experience
[14]STOR: Main Storage Constraint
[15]PVOL: Platform Volatility
[16]ACAP: Analyst Capability
[17]PCAP: Programmer Capability
[18]PCON: Personnel Continuity
[19]APEX: Applications Experience
[20]LTEX: Language and Tool Experience
[21]TOOL: Use of Software Tools
[22]SITE: Multisite Development
[23]SCED: Required Development Schedule

comprehensively evaluate the product, platform, personnel, project aspects of a software project [1] and the ratings have been practically proven to be the effective estimates of the magnitudes of their multiplicative or exponential effects on project effort. On the other hand, Use Case Points also evaluates projects based on a set of project characteristics, and the corresponding ratings of those project characteristics reflect how much the characteristics can influence project effort. To be specific, Use Case Points provides 13 technical factors and 8 environmental factors, as shown in TABLE IV and TABLE V. However, after examining the project evaluation factors defined in UCP against COCOMO II's cost drivers, we found that certain aspects of a project that are considered in COCOMO II are not modeled by Use Case Points, and those aspects are summarized in TABLE VIII.

To formally describe this observation, the model coverage $C(M)$ of an effort estimation model M is defined as the subset of the project characteristics defined in model M, such that, for $\forall c \in C(M)$, c is identical or correlated with a cost driver in COCOMO II: $c_{driver} \in C_{cocomo}$, where $C_{cocomo} = \{SF_1, \dots, SF_5\} \cup \{EM_1, \dots, EM_{17}\}$. $\{SF_1, \dots, SF_5\}$ are the 5 scale factors and $\{EM_1, \dots, EM_{17}\}$ are the 17 effort multipliers defined in COCOMO II. By this definition, since EUCP and EXUCP use the same set of project characteristics as UCP, $C(M_{UCP}) = C(M_{EUCP}) = C(M_{EXUCP})$, which means $C(M_{UCP})$ is the set of COCOMO II cost drivers that are covered by both M_{EUCP} and M_{EXUCP}, and $T = C_{cocomo} - C(M_{UCP})$ represents the subset of COCOMO II cost drivers that are not covered by any of the two models. In other words, for $\forall c_{driver} \in T$, c_{driver} is a source of variation in the effort data, which is explainable by COCOMO II but unexplainable by the proposed size metrics, and we use this set of cost drivers $T = \widetilde{SF} \cup \widetilde{EM}$ in normalization formula (8) to normalize actual effort data.

B. Normalization Formula

The normalization function is formulated by (8), which is derived from COCOMO II's post architecture model [3]. Let T be the set of cost drivers covered by none of the proposed estimation models, $\widetilde{EM}$ be the set of effort multipliers, and $\widetilde{SF}$ be the set of scale factors in T. Then, the normalized effort $Effort_{norm}$ is defined by (8).

$$\begin{aligned} Effort_{norm} &= \frac{Effort_{actual}}{\prod_i^{|\widetilde{EM}|} EM_i * KSLOC^{0.01*(\sum_i^{|\widetilde{SF}|} \Delta SF_i)}} \\ &= s_{norm} * Effort_{actual} \end{aligned} \quad (8)$$

, where $\Delta SF_i = SF_i - \overline{SF_i}$, $\overline{SF_i}$ represents the nominal value for SF_i, and s_{norm} represents the calculated scaling factor

[20]UEUCW: Unadjusted Early Use Case Weight
[21]UEXUCW: Unadjusted Extended Use Case Weight
[22]UAW: Unadjusted Actor Weight
[23]TCF: Technical Complexity Factor
[24]EF: Environmental Factor

TABLE IX
COUNTING RESULTS FOR THE 4 SAMPLE PROJECTS

Project	UEUCW	UEXUCW	UAW	TCF	EF
BDR	125	62	14	1.14	1.21
PCS	200	132	8	1.06	1.03
LBA	370	270	12	1.12	1.32
TIKI	95	34	3	1.18	1.24

TABLE X
RATINGS FOR THE UN-MODELED FACTORS

CDR	LBA	PCS	BDR	TIKI
PREC	L	N	N	N
RESL	H	H	H	N
FLEX	N	N	N	N
RELY	H	N	H	N
DATA	N	H	H	N
DOCU	H	H	H	H
STOR	N	N	N	N
PVOL	N	N	N	N
PLEX	N	N	H	N
TOOL	N	N	H	N
SITE	N	N	N	N
SCED	N	N	N	N
KSLOC	21.34	6.81	4.71	3.12
$E_{actual}(PH)$	3680	2016	1392	737
s_{norm}	0.82	0.85	0.91	0.90
$E_{norm}(PH)$	3029.65	1711.80	1270.21	663.96

V. MODEL CALIBRATION

A. Empirical Data

Four projects from USC's software engineering courses CSCI 577 and CSCI 590 - Location-based Advertising Platform (LBA, 2014-2015), Picshare (PCS, 2016), Bad Driver Report Platform (BDR, 2016), and Tiki Man Go (TIKI, 2017) - are selected to apply EUCP and EXUCP counting methods. They are either about developing web applications or mobile applications ranging from 3 - 21 KSLOC. The empirical study was done by examining the artifacts from these projects. The counting results are presented in TABLE IX.

B. Effort Data Normalization and Its Results

By applying the normalization equation (8) on the actual effort E_{actual} based on the ratings provided in TABLE X for the un-modeled cost drivers in T, we derived the normalized effort for the four sample projects. The interpretation of the normalized effort is, if the project is done under the nominal conditions of the un-modeled cost drivers in T, the expected effort would be PM_{norm}.

C. Calibrated Results with Normalized Data

Based on the counting results of EUCP and EXUCP (presented in TABLE XI) and the normalized effort data, we calibrated the two proposed effort estimation models respectively by ordinary least squares linear regression. The calibrated parameters are presented in TABLE XII and Fig. 2.

TABLE XI
EUCP, EXUCP, ACTUAL EFFORT, AND NORMALIZED EFFORT FOR THE 4 SAMPLE PROJECTS

Project	EUCP	EXUCP	PH_{actual}	PH_{norm}
BDR	191.74	104.83	1392	1270.21
PCS	227.09	152.85	2016	1711.80
LBA	564.75	416.91	3680	3029.65
TIKI	143.39	54.14	737	663.96

TABLE XII
CALIBRATED PARAMETERS WITH NORMALIZED EFFORT

Model	β_0	β_1
I	244.68	5.06
II	558.01	6.10

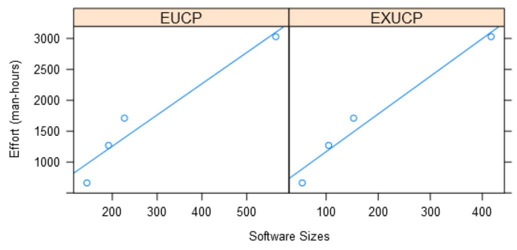

Fig. 2. Calibrated Linear Models with Normalized Effort Data

D. *Estimation Result Interpretation and Applicability*

Since the models are calibrated by normalized effort data, the effort estimated by the calibrated models can be interpreted as the expected effort for a project done under the nominal conditions of the un-modeled cost drivers in T. One can further estimate the multiplicative and exponential effects of the un-modeled cost drivers in T on project effort by rating the cost drivers according to the actual situation of a project, and then applying the estimates on top of the model-estimated effort to yield more accurate estimation. For example, if an effort estimate given by $Model_{EUCP}$ is t person-hours, we understand that this result is estimated under the nominal condition of platform volatility (PVOL), while if the actual rating of PVOL for the project is high, we can further adjust the estimated result t by multiplying the value for the high rating of PVOL according to COCOMO II. As a result, integrating the knowledge about the effect of PVOL on project effort from COCOMO II helps achieve more accurate effort estimation. As for the applicability of the calibrated models presented in TABLE XII, since the projects used in calibration as data points are majorly domain-specific as web or mobile applications, and the data points are minimal for the purpose of demonstrating calibration process and show certain degree of linear relationships between the metrics and the effort, local calibration is recommended if one wants to get accurate estimation for a specific setting of software development.

TABLE XIII
MMRE, PRED (25%), AND R-SQUARED FOR $Model_I$ AND $Model_{II}$

Model	MMRE	PRED (25%)	R^2
I	17.9%	75%	0.933
II	13.7%	75%	0.964

TABLE XIV
SE, T-VALUES, P-VALUES FOR INTERCEPTS AND SLOPES

Model	β_0			β_1		
	SE[25]	t-v[26]	p-v[27]	SE	t-v	p-v
I	313.92	0.779	0.517	0.960	5.266	0.034
II	192.25	2.903	0.101	0.837	7.286	0.018

VI. MODEL EVALUATION

We first evaluated the goodness of fit with the statistics: R-squared (R^2), mean magnitude of relative error (MMRE), and percentage relative error deviation of 25% (PRED (.25)), and then tested significance of the linear relationships between the two proposed size metrics and project effort with hypothesis tests on the slopes (β_1) of the linear models.

A. *Evaluation of Goodness of Fit*

As shown in TABLE XIII, the linear models explain the variance of effort data very well with respect to the high values of R-squared (R^2). Originally, MMRE and PRED are calculated using a testing data set to prove a model's estimation accuracy. However, due to the limited number of data points available, MMRE and PRED are calculated using the training data set to evaluate goodness of fit instead of estimation accuracy. To evaluate the goodness of fit of the calibrated models, we adopted the criteria commonly used in evaluating effort estimation models - MMRE being less than 25% and PRED (.25) being greater than 75% [17]. As the results presented in TABLE XIII, the MMRE values of 17.9% and 13.7% for $Model_I$ and $Model_{II}$ suggest the proposed models fit the data well, for the reason that, on average, the absolute errors are with 25% of the actual values. The values of PRED (.25) lead to the same conclusion for the absolute errors for both the models are within 25% of the actual values for 75% of the time.

B. *Hypothesis Tests for the Slopes*

As provided in TABLE XIV, the p-values for the slopes of the linear models - $Model_I$ and $Model_{II}$ - are 0.034 and 0.018 respectively, for which we reject the null hypothesis h_0 : $\beta_1 = 0$ for both the models, regarding the commonly adopted significance level of 5% in statistical hypothesis testing, which means linear relationships between the proposed size metrics and normalized effort are significant.

[25]SE: Standard Error
[26]t-v: t-value
[27]p-v: p-value

C. Model Comparison

In comparison, $Model_{II}$ is superior to $Model_{I}$, in terms of its lower MMRE value and higher R-squared value, both of which indicate $Model_{II}$ fits the data set better. Also the estimated standard errors for the slope and the intercept of $Model_{II}$ are both lower than the parameters of $Model_{I}$, which means $Model_{II}$ has less uncertainty in the estimated parameters. However, to draw a conclusion on the superiority of $Model_{II}$ over $Model_{I}$ in terms of estimation accuracy with certain degree of confidence requires more data points to be collected.

VII. Conclusions and Future Directions

In this paper, we proposed two size metrics based on the information available at the early stage of a use case driven project - the requirements elicitation iteration and the analysis and design iteration. We further introduced a framework to eliminate unexplainable variation in the effort data by normalizing the influences from the un-modeled factors, while the magnitudes of the influences are understood from COCOMO II. Also, to demonstrate the model calibration process and the effectiveness of the proposed metrics in estimating project effort, we applied the proposed counting methods and the normalization framework on four student projects to derive the data points for OLS linear regression. The preliminary calibration results have shown that the calibrated models fit the data set well. We further argued that $Model_{II}$ is superior to $Model_{I}$ in terms of the goodness of fit and the uncertainty in the estimated parameters. To further evaluate the estimation accuracy, the models need to be tested with an independent testing data set. To make a conclusion about the superiority of $Model_{II}$ over $Model_{I}$ in terms of estimation accuracy with certain degree of confidence also requires more data points. For those reasons, future work will be focused on collecting more data points. Also the whole framework of taking measurements from the relevant artifacts, normalizing effort data, and training and testing the models need to be streamlined by a suite of software tools. This will improve the efficiency of the model calibration process.

References

[1] B. W. Boehm, *Software cost estimation with Cocomo II*. Upper Saddle River, NJ: Prentice Hall, 2000.

[2] B. W. Boehm, J. A. Lane, S. Koolmanojwong, and . A. . Turner, Richard, *The incremental commitment spiral model: principles and practices for successful systems and software*. Upper Saddle River, NJ: Addison-Wesley, 2014.

[3] I. Jacobson, *Object-oriented software engineering: a use case driven approach*. [New York] :Wokingham, Eng. ;Reading, Mass: ACM Press ;Addison-Wesley Pub, 1992.

[4] D. Rosenberg, M. Stephens, and M. Collins-Cope, *Agile development with ICONIX process: people, process, and pragmatism*. Berkeley, CA: Apress, 2005.

[5] B. W. Boehm, *Software engineering economics*. Englewood Cliffs, N.J: Prentice-Hall, 1981.

[6] M. Stephens, D. Rosenberg, and I. Books24x7, *Design driven testing: test smarter, not harder*, 1st ed. New York: Apress, 2010;2011;.

[7] A. Cockburn, *Writing Effective Use Cases*, ser. Agile Software Development Series. Addison-Wesley, 2001. [Online]. Available: https://books.google.com/books?id=VKJQAAAAMAAJ

[8] I. S. Group, "Getting a good start with better requirement management use case driven development," IBM Rational Software, 2004.

[9] G. Karner, "Metrics for objectory. diploma thesis," Ph.D. dissertation, University of Linkoping, 1993.

[10] A. J. Albrecht, "Measuring application development productivity," in *Proc. of the Joint SHARE/GUIDE/IBM Applicaiton Development Symposium*, 1979, pp. 83–92.

[11] A. Albrecht, "Function point analysis," in *Encyclopedia of Software Engineering*. Wiley, 1994, vol. 1, pp. 518–524.

[12] K. Periyasamy and A. Ghode, "Cost estimation using extended use case point (e-ucp) model," 2009, pp. 1–5.

[13] M. M. Kirmani and A. Wahid, "Revised use case point (re-ucp) model for software effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 3, pp. 65–71, 2015.

[14] B. Clark, S. Devnani-Chulani, and B. Boehm, "Calibrating the cocomo ii post-architecture model," in *Software Engineering, 1998. Proceedings of the 1998 International Conference on*. IEEE, 1998, pp. 477–480.

[15] B. K. Clark, "The effects of software process maturity on software development effort," PhD dissertation, Queensland University of Technology, 1997.

[16] R. Clem, *Project Estimation with Use Case Points*, Diversified Technical Services, Inc, 2007.

[17] D. Nguyen and T. M. WVU, "Studies of confidence in software cost estimation research based on the criterions mmre and pred," 2009.

[18] S. Chulani, B. Boehm, and B. Steece, "Bayesian analysis of empirical software engineering cost models," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 573–583, 1999.

[19] H. Astudillo, G. Gnova, M. miaek, J. Llorens, P. Metz, and R. Prieto-Daz, *Use Cases in Model-Driven Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 3844, pp. 272–279.

[20] F. Wang, X. Yang, X. Zhu, and L. Chen, "Extended use case points method for software cost estimation," 2009, pp. 1–5.

[21] W. G. Cochran, "Errors of measurement in statistics," *Technometrics*, vol. 10, no. 4, pp. 637–666, 1968.

[22] Z. John Lu, "The elements of statistical learning: data mining, inference, and prediction," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 173, no. 3, pp. 693–694, 2010.

[23] M. Arnold and P. Pedross, "Software size measurement and productivity rating in a large-scale software development department," in *Software Engineering, 1998. Proceedings of the 1998 International Conference on*. IEEE, 1998, pp. 490–493.

[24] B. Anda, H. Dreiem, D. I. Sjøberg, and M. Jørgensen, "Estimating software development effort based on use casesexperiences from industry," in *International Conference on the Unified Modeling Language*. Springer, 2001, pp. 487–502.

[25] J. Smith, "The estimation of effort based on use cases," *Rational Software white paper*, 1999.

[26] T. Uemura, S. Kusumoto, and K. Inoue, "Function point measurement tool for uml design specification," in *Software Metrics Symposium, 1999. Proceedings. Sixth International*. IEEE, 1999, pp. 62–69.

[27] *Function Point Counting Practices Manual*, 4th ed., International Function Point Users Group (IFPUG), 2002.

[28] C. R. Symons, "Function point analysis: difficulties and improvements," *IEEE transactions on software engineering*, vol. 14, no. 1, pp. 2–11, 1988.

[29] F. Heemstra and R. Kusters, "Function point analysis: Evaluation of a software cost estimation model," *European Journal of Information Systems*, vol. 1, no. 4, pp. 229–237, 1991.

[30] D. Rosenberg, B. Boehm, K. Qi, and B. Wang, "Rapid, evolutionary, reliable, scalable system and software development: The resilient agile process," in *Proc. of the ICSSP17*, 2017.

[31] B. Boehm, J. A. Lane, S. Koolmanojwong, and R. Turner, "Architected agile solutions for software-reliant systems," in *Agile Software Development*. Springer, 2010, pp. 165–184.

[32] J. E. Matson, B. E. Barrett, and J. M. Mellichamp, "Software development cost estimation using function points," *IEEE Transactions on Software Engineering*, vol. 20, no. 4, pp. 275–287, 1994.

[33] M. Azzeh, "Software cost estimation based on use case points for global software development," in *Computer Science and Information Technology (CSIT), 2013 5th International Conference on*. IEEE, 2013, pp. 214–218.