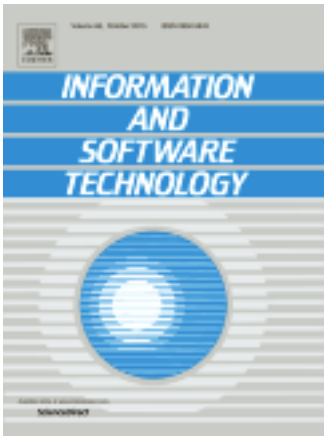
Simplifying effort estimation based on Use Case Points

by M. Ochodek, J. Nawrocki, K. Kwarciak



pre-print submitted to:
Information and Software Technology

Please cite as:

Ochodek, M., Nawrocki, J., & Kwarciak, K. (2011). Simplifying effort estimation based on Use Case Points. Information and Software Technology, 53(3), 200-213,
doi: http://dx.doi.org/10.1016/j.infsof.2010.10.005.

Bibtex entry:

```
@article{Ochodek2011200,
title = "Simplifying effort estimation based on Use Case Points",
journal = "Information and Software Technology",
volume = "53",
number = "3",
pages = "200--213",
year = "2011",
issn = "0950-5849",
doi = "http://dx.doi.org/10.1016/j.infsof.2010.10.005",
author = "M. Ochodek and J. Nawrocki and K. Kwarciak"
}
```

Simplifying Effort Estimation Based on Use Case Points☆

M. Ochodek[a,*], J. Nawrocki[a], K. Kwarciak[a]

[a]*Poznan University of Technology, Institute of Computing Science*
ul. Piotrowo 2, 60-965 Poznań, Poland

Abstract

Context: The Use Case Points (UCP) method can be used to estimate software development effort based on a use-case model and two sets of adjustment factors relating to the environmental and technical complexity of a project. The question arises whether all of these components are important from the effort estimation point of view.

Objective: This paper investigates the construction of UCP in order to find possible ways of simplifying it.

Method: The cross-validation procedure was used to compare the accuracy of the different variants of UCP (with and without the investigated simplifications). The analysis was based on data derived from a set of 14 projects for which effort ranged from 277 to 3593 man-hours. In addition, the factor analysis was performed to investigate the possibility of reducing the number of adjustment factors.

Results: The two variants of UCP – with and without unadjusted actor weights (UAW) provided similar prediction accuracy. In addition, a minor influence of the adjustment factors on the accuracy of UCP was observed. The results of the factor analysis indicated that the number of adjustment factors could be reduced from 21 to 6 (2 environmental factors and 4 technical complexity factors). Another observation was made that the variants of UCP calculated based on steps were slightly more accurate than the variants calculated based on transactions. Finally, a recently proposed use-case-based size metric TTPoints provided better accuracy than any of the investigated variants of UCP.

Conclusion: The observation in this study was that the UCP method could be simplified by rejecting UAW; calculating UCP based on steps instead of transactions; or just counting the total number of steps in use cases. Moreover, two recently proposed use-case-based size metrics Transactions and TTPoints could be used as an alternative to UCP to estimate effort at the early stages of software development.

Keywords: Use Case Points, software cost estimation, use cases, use-case transactions, TTPoints

1. Introduction

Software effort estimation is one of the key aspects of successful project management. If an unrealistic assumption about the development cost is made, the project is in danger. Both underestimated and overestimated effort is harmful. Underestimation leads to a situation where a project's commitments cannot be fulfilled because of a shortage of time and/or funds. Overestimation can result in the rejection of a project proposal, which otherwise would be accepted and would create new opportunities for the organization.

Unfortunately, effort estimation at the early stages of software development is a challenge. Firstly, very little is known about the project. Secondly, there is a threat that the project will not be accepted for further development, so limited resources can be spent on effort estimation. Thus, there is a trade-off between the level of estimation error and the resources assigned to the estimation activities (typically, the smaller the estimation error the bigger the estimation cost associated with acquiring knowledge about the project at hand).

In this context two kinds of research could be useful:

- simplifying effort estimation methods without compromising their accuracy;
- making effort estimation more accurate without increasing the time and money spent on effort estimation.

Typical inputs available at early stages of software

☆This research project operated within the Foundation for Polish Science Ventures Programme co-financed by the EU European Regional Development Fund.

*Corresponding author
Email addresses: mochodek@cs.put.poznan.pl (M. Ochodek), jnawrocki@cs.put.poznan.pl (J. Nawrocki), kkwarciak@cs.put.poznan.pl (K. Kwarciak)

development are functional requirements, which describe what a system is expected to do. These kinds of requirements can be used to measure the size of a system, and estimate its development effort.

The idea of functional size measurement (FSM) was introduced by Allan Albrecht [1], who proposed a method called Function Point Analysis (FPA). Since the introduction of the method, its construction has been broadly discussed and frequently questioned (see, e.g., [2, 3, 4, 5, 6]). Nevertheless, it still remains one of the most popular FSM methods, and since 1986, it has been maintained by a non-profit organization called the International Function Point User Group (IFPUG).

Albrecht's FPA method stimulated evolvement of other FSM methods, e.g., Mark II Function Points proposed by Symons [7], COSMIC [8], or Use Case Points[1] (UCP) introduced by Karner [9].

The latter method is especially valuable in the context of early size measurement and effort estimation, because it employs *use cases* as an input. Use cases, proposed by Jacobson [10, 11], are a popular form of representing functional requirements (according to the survey conducted by Neill and Laplante in 2003 [12], 50% of projects have their functional requirements presented as scenarios or use cases). They are also available in the early stages of software development.

The mechanism of the Use Case Points method was inspired by both Albrecht's FPA [1] and MK II Function Points [7], and since its introduction many variants of the method have been proposed, e.g., Use Case Size Points (USP) and Fuzzy Use Case Size Points (FUSP) [13]; UCPm [14]; and the adapted version for incremental large-scale projects [15].

As use cases are gaining popularity also the UCP method (and its derivatives) are getting more popular. However, some people pointed out problems concerning the construction of the method (differences in use case models [16, 17], assessment of the use-case-model complexity [13, 17], assessment of adjustment factors [14, 17, 18], and involvement of calculations that are based on algebraically inadmissible scale-type transformations [18, 19]). Therefore, the question arises whether the method is well designed. Maybe it could be simplified without loosing much of its accuracy.

[1]It is not clear whether UCP is a size measure or a software estimation method. Some sub-components of UCP (presented in Section 2.2) such as UUCW, UAW, and UUCP could be clearly treated as functional size measures. However, when UUCP is multiplied by TCF and EF, it is no longer clear whether it represents size of the system or its predicted development effort. (The environmental factors represent commonly used cost drivers.)

This question is even more important in the context of recently proposed use-case-based size metrics, i.e., Transactions [20], and TTPoints [21]. These metrics seem simpler than UCP.

Therefore, the goal of this study is to analyze the construction of the UCP method, investigate the influence of its components on the accuracy of the method, and propose possible simplifications.

The paper is organized as follows. The next section provides a brief introduction to use cases and the UCP method. Section 3 presents a set of projects used in this study as a historical database. Section 4 describes the research method that was used to evaluate the estimation accuracy of the different variants of UCP and other use-case-based size metrics. In the following sections components of UCP are analyzed: actors complexity – in Section 5; adjustment factors – in Section 6; use-case complexity – in Sections 7 and 8. The role of transactions in use-case-based effort estimation is investigated in Section 9. The threats to validity of this study are discussed in Section 10. The summary and the list of the most important findings can be found in Section 11.

2. Use Cases and the Use Case Points Method

2.1. Use Cases

The main aim of use cases is to present interaction between end-user (called actor) and the described system in terms of user-valued transactions – using natural language. Such use cases are called system-level use cases. (There are also business-level use cases: they describe interaction between people who cooperate to obtain a business goal.)

According to the guidelines for writing use cases [22, 23] the most important parts of a use case are as follows: *name/title* which describes the goal, *actors* participating in the use case (people or cooperating systems), *main scenario* which is the most common sequence of steps leading to the goal, and *extensions* to the main scenario describing alternative steps associated with the occurrence of some events. An example of a use case is presented in Figure 1.

2.2. The Use Case Points Method

In order to obtain UCP for the system one has to start with the assessment of the complexity of actors and use cases; and then adjust it with two kinds of factors characterizing the development environment and the technical complexity of the system under development.

UC1: Submit a paper
Level: **User**
Main actor: **Author**

Main Scenario:
1. **Author** chooses the option to submit a *paper*.
2. **System** presents the submission form.
3. **Author** provides necessary information about the *paper*.
4. **System** informs Author that the *paper* was submitted.

Alternatives, Extensions, Exceptions:
3.A. Not all required data was provided.
3.A.1. **System** displays error message.
3.A.2. Go to step 2.

Figure 1: An example of a use case presented as a structured text

2.2.1. Actors Complexity

The first step of the UCP method is to assign each actor to one of three complexity classes:

- *simple*: an actor representing a system which communicates with other actors using API;
- *average*: a system actor which communicates through a protocol (e.g. HTTP, FTP), or a person who interacts with a system through a terminal console;
- *complex*: a person who uses graphical user interface (GUI) in order to communicate with a system.

Each actor-complexity class, c, is characterized by two numbers:

- $aWeight(c) = 1$ for *simple*, 2 for *average*, and 3 for *complex*;
- $aCardinality(c)$ is the number of actors assigned to class c (depends on a described system).

For a given system, the *unadjusted actor weights (UAW)* are computed as a sum of products – the weight of complexity class and the number of actors assigned to that class (see Equation 1).

$$UAW = \sum_{c \in C} aWeight(c) \times aCardinality(c), \qquad (1)$$
$$C = \{simple, average, complex\}$$

2.2.2. Use-Cases Complexity

The second step of the UCP method is the assessment of use-case complexity. This complexity depends on the number of transactions identified in each use case. (Transaction is a set of activities in use-case-scenarios, which is either performed entirely, or not at all.)

Let $\#trans(u)$ denote the number of transactions in a use case u. Each use case, u, is assigned to complexity class, $cmplx$, in the following way:

$$cmplx(u) = \begin{cases} simple & \text{if } \#trans(u) < 4, \\ average & \text{if } 4 \leq \#trans(u) \leq 7, \\ complex & \text{if } \#trans(u) > 7. \end{cases}$$

Each use-case-complexity class, c, is characterized by two numbers:

- $uWeight(c) = 5$ for *simple*, 10 for *average*, and 15 for *complex*;
- $uCardinality(c)$ is the number of use-cases assigned to class c (depends on a described system).

For a given system, the *unadjusted use case weights (UUCW)* are calculated according to Equation 2.

$$UUCW = \sum_{i \in C} uWeight(c) \times uCardinality(c), \qquad (2)$$
$$C = \{simple, average, complex\}$$

2.2.3. Technical and Environmental Factors

The UCP method includes 21 adjustment factors, which concern the technical complexity of the developed system (13 *technical complexity factors*), and the environment in which it is developed (8 *environmental factors*). All the factors are presented in Table 1.

The influence of technical complexity factors (TCF) are assessed by assigning a value from 0 to 5 to each of them (the bigger the number is, the greater the extent a given factor appears with). This value is multiplied by a weight of a factor and totaled (see Equation 3).

$$TCF = 0.6 + (0.01 \times \sum_{i=1}^{13} TF\_weight_i \times value_i) \qquad (3)$$

where

- TF_weight_i is the weight of the i-th technical complexity factor (see Table 1);
- $value_i$ is the predicted degree of influence of the i-th technical complexity factor on the project (value between 0 and 5).

The influence of environmental factors (EF) is assessed in a similar way as in the case of technical complexity factors (see Equation 4).

$$EF = 1.4 + (-0.03 \times \sum_{i=1}^{8} EF\_weight_i \times value_i) \qquad (4)$$

Table 1: Technical Complexity Factors and Environmental Factors

Technical Complexity Factors		
Factor	Description	Weight
T1	Distributed system	2
T2	Performance	1
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	1
T11	Security features	1
T12	Access for third parties	1
T13	Special training required	1
Environmental Factors		
Factor	Description	Weight
F1	Familiarity with the standard process	1.5
F2	Application experience	0.5
F3	Object-oriented 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

where

- EF_weight_i is the weight of the i-th environmental factor (see Table 1);
- $value_i$ is the predicted degree of influence of the i-th environmental factor on the project (value between 0 and 5).

2.2.4. Calculating Use Case Points

By adding UAW to UUCW, according to Equation 5, one obtains *unadjusted use case points (UUCP)*.

$$UUCP = UAW + UUCW \tag{5}$$

To obtain *use case points (UCP)* one has to multiply UUCP by TCF and EF (see Equation 6).

$$UCP = UUCP \times TCF \times EF \tag{6}$$

2.2.5. Productivity Factor and Effort Estimation

To obtain effort estimation in man-hours one has to multiply UCP by the *productivity factor (PF)*[2].

The default value for PF proposed by Karner is 20 hours per UCP. Schneider and Winters [26] proposed a method for determining the initial value of PF. Based on their experience, they suggested to count the number of environmental factors F1-F6 which influence is predicted to be less than 3 and factors F7-F8 which influence is predicted to be greater than 3. If the counted total is equal to 2 or less, the default value of 20 h/UCP should be used. If the total is between 3-4, they suggested using PF equal to 28 h/UCP. If the calculated number is greater than 4, the value of 36 h/UCP should be used. (However, in this case the project is regarded as an extremely risky one.)

2.2.6. Calibrating UCP with Historical Data

Using default values for PF is a necessity if an organization does not have historical data concerning productivity. However, if historical data is available, it is reasonable to use such data to determine PF for the project being estimated. After the completion of a single project, a post-productivity factor (PostPF) might be calculated as presented in Equation 7.

$$PostPF = \frac{ActualEffort}{UCP} \tag{7}$$

3. Characteristics of the Projects

This study is based on an analysis of 14 projects, which actual effort ranged from 277 to 3593 man-hours. The brief description of the projects is presented in Table 2. The detailed characteristics related to the effort estimation with UCP are presented in Table 3.

The projects labeled from A to G were developed by industrial organizations. The projects H to N were developed at the Poznan University of Technology (PUT). (All of the projects which origin was marked as 'U' were also budgeted; therefore, they remained similar to the industrial projects in this aspect.)

Because the main data set was heterogeneous we decided to perform the analysis based on the whole data set and the number of its subsets that seemed homogeneous in some aspects.

[2]The name "Productivity Factor" can be misleading, because a quotient between effort and size is usually referred as Project Delivery Rate (PDR) [24]. (Productivity, however, is more often defined as a quotient between size and effort [25].)

The first criterion for choosing candidates for subsets was the origin of a project (subsets *Ind* and *Uni*). The next criteria were the type of application and the main programming language (subsets *Web* and *Java*). We also defined a small, but highly homogeneous subset *ADM*. It contained three projects that were conducted at PUT to develop the students' admission system in years 2006-2010. Finally, we decided to perform a *hierarchical clustering analysis*[3] to find the most similar projects in respect to their TCFs and EFs. As a result three small subsets were defined *EF1*, *EF2*, and *TCF1*.

The summary of the data sets considered in this study is presented in Table 4.

Table 4: The subsets of the main data set considered in this study

ID	Description	Projects
All	All projects	A-N (14)
Ind	Industrial projects	A-G (7)
Uni	Projects developed at the university	H-N (7)
Web	Web applications	A, C, D, F-N (12)
Java	Projects developed in Java	G-N (8)
ADM	Students' admission system at PUT	H, I, K (3)
EF1	Projects with similar EFs	D, F, N (3)
EF2	Projects with similar EFs	H, I, K, M (4)
TCF1	Projects with similar TCFs	H, I, M (3)

4. Framework for Analysis of Estimation Accuracy

The accuracy of effort estimation based on each of the considered size metrics was analyzed according to the procedure presented in this section.

4.1. Calculation of Use Case Points

The process of calculating UCP (and other size metrics) for the projects, consisted of the following steps:

1. *Reviewing use cases.* All business-level use cases were rejected (they create a context for user-level use cases); included and extended use cases were rejected (as suggested in UCP [9]), but only if they described the same transaction as the invoking use case, but at lower level of abstraction. Use cases that were not implemented or duplicated were also rejected.
2. *Counting transactions and steps.* Use-case transactions for UCP were identified by a single person, with the use of the method proposed by Robiolo and Orosco [20] (which seems to be compliant with the Karner's definition of use-case transaction [9]). In addition, semantic transaction types were identified [29] for the purpose of the TTPoints method discussed in Section 9. We also asked an experienced person, not co-authoring this paper, to review the transaction counts for the sample of 50 use cases (20%). Then, the results were discussed. Although the final transaction-counts and UUCW were not the same for both persons, they seemed to be convergent[4].
3. *Obtaining TCF and EF.* The values of the adjustment factors were obtained by surveying participants of the projects. The final values of TCF and EF for the project were calculated as (Optimistic + 4 × Average + Pessimistic) / 6.
4. *Calculating UCP.* It was done according to the procedure presented in Section 2.2.

4.2. Evaluation of Prediction Accuracy

Each metric should be evaluated in respect to its purpose. In this study we investigated the usefulness of use-case-based size metrics in the context of software cost estimation. Therefore, the main criterion for comparing the size metrics was the accuracy of effort estimation calculated based on them.

In order to determine the accuracy of effort estimation based on each size metric we followed a *cross-validation* procedure [32]. It is a statistical method for validating a predictive model. The considered set of observations is divided into two subsets. One of them, called a training set, is used to construct a prediction model. The remaining one is used to validate the model. In this study we used a form of cross-validation called "leave-one-out" which leaves out a single observation for the validation purpose at a time.

In the study two following effort predictions models[5]

[3]The goal of hierarchical clustering is to organize a set of observations into a hierarchy of subsets (hierarchy of clusters) so that observations in the same cluster are similar to each other. In this study we used Ward's algorithm [27] and Euclidean distance. (They can be used to cluster ordinal data [28].)

[4]The normalized root mean squared deviation (*NRMSD*), calculated as $\sqrt{\frac{1}{n}[\sum_{i=1}^{n}(x_{1,i}-x_{2,i})^2]}/(x_{max}-x_{min})$, between the number of transactions identified by two raters ranged from 0.10 to 0.15. The inter-rater agreement in assignment of use cases to complexity classes, measured as weighted Cohen's kappa coefficient (κ) [30, 31], could be interpreted as "almost perfect" [31] (κ ranged from 0.83 to 0.85).

[5]We did not use the default value of PF equal to 20 h/UCP, because it can be different for each variant of UCP (it has to be obtained by calculating the PostPFs for a set of projects). In addition, it can differ between organizations – at the end Karner proposed the value based on the projects developed at Objectory. (The observed PostPF for the projects considered in this study ranged from 3 to 35 – see Table 3.)

Table 2: Application domain and basic description of the projects under study. *Origin: I – project developed by a software development company; U – projects developed by university staff and students for the internal usage at the university; S2B – project developed by students for external organizations. Type: N – application was developed from scratch; C – application was based on existing solution and was tailored for the customer; E – major enhancement, i.e., strongly simplified version was available (e.g. a prototype).*

ID	Effort [h]	Origin	Type	Team	Technologies	Description
Project A	3037	I	N	5	ASP.Net, C#, MS SQL DBMS, custom web-framework	Web-based e-commerce solution.
Project B	1917	I	N	4	Delphi, Firebird DBMS	Integration of two sub-systems within the ERP scale system.
Project C	1173	I	N	8	Python, Django, PostgreSQL	Web application for collecting and tracking projects' metrics.
Project D	742	I	C	6	Python, Plone, Zope	Web application developed based on existing CMS solution.
Project E	614	I	N	4	Delphi, Firebird DBMS	Bank system integrating payments into virtual accounts in one real account.
Project F	492	I	C	3	Python, Plone, Zope	Content Management System (CMS).
Project G	277	I	N	1	Java, PHP, MySQL DBMS, Eclipse Rich Client Platform (RCP)	Web-based invoices repository with additional standalone client application.
Project H	3593	U	N	8	Java, Oracle DBMS, Hibernate, GWT-based custom framework, JSON, OSGi	Backend application for the university admission system (Rich Internet Application).
Project I	1681	U	E	8	Java, Oracle DBMS, Apache Struts 1.2, Java Swing	Web-based frontend for the university admission system. Some parts were re-used from the previous prototype version.
Project J	1344	S2B	N	6	ASP.Net, C#, MS SQL DBMS	Web-based Customer Relationship Management (CRM) system.
Project K	1220	U	N	6	Java, Oracle DBMS, Hibernate, GWT-based custom framework, JSON, OSGi	University admission system for foreign students.
Project L	720	S2B	N	6	Java, Java Servlets, JSP, Java Swing, SOAP, MySQL DBMS	Web & standalone application for managing members of the organization.
Project M	524	U	N	6	Java, Hibernate, Axis, PHP, C++	Bibliometric Information System. A system for collecting information regarding publications and citations.
Project N	397	U	N	3	Java, Apache Struts 2, Hibernate	Web-based system supporting the assignment of B.Sc and M.Sc. theses projects.

were used to estimate effort: single linear regression and multiple regression.

4.2.1. Ordinary Least Squares Regression (OLS)

Ordinary least squares regression (OLS) is used to linearly approximate the relationship between a single dependent variable (e.g., development effort) and one (or more) independent variables (e.g., size of the system). It minimizes the sum of squared distances between the observed values of the dependent variable, and the values predicted by the linear approximation. (For further details refer to [33].)

A single linear regression model for predicting effort based on the value of the size metric which was used in this study has a form presented in Equation 8. (The linear regression equation was constructed in each step of the cross-validation procedure based on the projects constituting a training set.)

$$Effort_{p_i} = \beta_{Size} \times Size(p_i) + \beta_0 \quad (8)$$

where

- p_i is the project for which effort is estimated;
- β_{Size} is the slope for *Size*;
- β_0 is the constant or intercept (set to 0);
- $Size(p_i)$ is the value of the size metric calculated for the project p_i;

4.2.2. Multiple Regression

The purpose of multiple regression is to learn about the relationship between several independent (or predictor) variables and dependent variable.

In the already presented linear regression model we considered only one independent variable – the size metric. The question is whether any additional variables could be added to the model to improve its effort prediction capabilities. Therefore, we decided to use the *best subset regression procedure* [34] to look for other candidate-variables that could be incorporated to the model (among all environment factors, technical complexity factors, and a factor called team size). This procedure was performed for each size metric and data set. The coefficients of the regression model were, then, calibrated in each step of the cross-validation procedure based on the projects belonging to a training set (the fac-

Table 3: Projects characteristics *(T – transactions, identified using stimuli-verb approach proposed by Robiolo and Orosco [20]; S – steps without reference to other use cases – include and extend relations; number of use cases before the review of the specifications are placed in brackets)*

	Project A	Project B	Project C	Project D	Project E	Project F	Project G	Project H	Project I	Project J	Project K	Project L	Project M	Project N
Actual Effort [h]	3037	1917	1173	742	614	492	277	3593	1681	1344	1220	720	514	397
UCP-T	148	55	76	105	63	44	22	304	80	74	89	50	31	95
UCP-S	276	58	113	161	112	90	36	505	162	157	160	90	59	144
UUCP-T	207	90	81	139	97	59	56	319	97	142	89	82	34	115
UUCP-S	387	95	121	214	172	119	91	529	197	302	159	147	64	175
UUCW-T	195	80	75	130	85	50	50	305	85	130	80	70	30	100
UUCW-S	375	85	115	205	160	110	85	515	185	290	150	135	60	160
UCP-T no UAW	139	49	70	98	55	38	20	291	70	67	80	43	27	82
Steps	224	45	69	123	101	67	36	324	109	152	96	69	38	84
Transactions [20]	86	30	33	62	25	17	12	135	34	49	33	27	10	33
TTPoints [29]	137	70	36	52	35	22	15	156	74	56	39	38	28	35
No. of use cases	31 (35)	12* (16)	11 (12)	19 (20)	15 (21)	10 (12)	10 (12)	42 (47)	17 (19)	26 (37)	13 (20)	14 (17)	6 (8)	18 (20)
Simple - T	23	6	7	13	13	10	10	23	17	26	10	14	6	16
Average - T	8	5	4	5	2	0	0	19	0	0	3	0	0	2
Complex - T	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Simple - S	0	9	3	3	4	3	3	4	0	0	2	1	2	6
Average - S	18	1	4	10	5	2	7	15	14	20	5	13	2	10
Complex - S	13	2	4	6	6	5	0	23	3	6	6	0	2	2
No. of actors	4	4	2	3	4	3	2	5	4	4	3	4	2	5
Simple	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Average	0	2	0	0	0	0	0	1	0	0	0	0	2	0
Complex	4	2	2	3	4	3	2	4	4	4	3	4	0	5
TCF**	0.92	0.75	0.90	0.85	0.82	0.85	0.78	0.94	1.03	0.71	1.05	0.78	0.96	0.90
EF**	0.78	0.81	1.05	0.89	0.79	0.88	0.51	1.02	0.80	0.73	0.95	0.79	0.96	0.91
Post PF (transactions)	21	35	15	7	10	11	12	12	21	18	14	14	17	4
Post PF (steps)	11	33	10	5	5	5	8	7	10	9	8	8	9	3
Default PF [26]	20	20	20	20	20	20	20	20	20	20	20	20	20	20

* *One of use cases did not have stimuli in the sense of [20].*
** *Results of the surveys conducted within each development team, and aggregated as (Optimistic + 4 × Average + Pessimistic) / 6.*

tors chosen for each subset of projects are presented in Table 5).

There are some issues regarding applying multiple regression that are important for this study. First of all, one has to be careful to not "overfit" the model to the data. (It is observed if the model has to many degrees of freedom, in relation to the amount of data available.) Therefore, we decided to include only one additional independent variable at a time, and apply the multiple regression model only to the sets that contained more than 4 observations. We also did not apply the multiple regression to the original UCP measure as it already included technical and environmental factors.

The multiple regression model considered in this study is presented in Equation 9.

$$\begin{aligned} Effort_{p_i} &= \beta_{Size} \times Size(p_i) \\ &+ \beta_{Factor} \times Factor(p_i) + \beta_0 \end{aligned} \tag{9}$$

where

- p_i is the project for which effort is estimated;
- $Factor(p_i)$ is the value of the additional factor included in the regression model (chosen among the EFs, TCFs, and team size) for the project p_i;
- β_{Size} is the slope for *Size*;
- β_{Factor} is the slope for *Factor*;
- β_0 is the constant or intercept;
- $Size(p_i)$ is the value of the size metric calculated for the project p_i;

4.2.3. The Evaluation Criteria

The next step of the analysis is the evaluation of the prediction accuracy, which is performed based on the following criteria [35, 36, 37]:

- *MRE* – which stands for the magnitude of relative error. The MRE is calculated for each project in the data set according to Equation 10. It indicates the difference between the estimated and the actual effort in respect to the actual effort. The *mean* MRE (MMRE) is used to aggregate the multiple observations of MRE in the whole data set.

$$MRE = \frac{|ActEffort - EstEffort|}{ActEffort} \tag{10}$$

- *Mean RE* – which stands for the mean relative error. It is calculated for a set of projects according to Equation 11.

$$Mean\ RE = \frac{1}{n} \sum_{i} \frac{(ActEffort_i - EstEffort_i)}{ActEffort_i} \tag{11}$$

- $Pred(e) = k/n$ – prediction quality is calculated on a set of n projects, where k is the number of projects for which estimation error (MRE) is less than or equal to e. In this study e was set to 0.25. Conte et al. [37] suggested that for acceptable estimation model, the value of Pred(0.25) should exceed 0.75.

The interpretation of MRE and Pred criteria is that the accuracy of an estimation technique is proportional to the Pred and inversely proportional to the MRE.

The Mean RE helps to investigate the bias of the estimates. If the estimation model is unbiased the value of the Mean RE should be close to zero (equal number of over-estimates and under-estimates cancel each other).

For testing the statistical significance of differences in accuracy between paired samples we used the two-tailed Wilcoxon signed-rank test with the significance level α set to 0.05. (However, the results of these tests should be taken with caution due to their low statistical power – the issue is discussed later on in Section 10.)

5. Actors Complexity in Use Case Points

Karner included information concerning actors in the UCP method (UAW, see Section 2.2.1), however the impact of the UAW on the accuracy of the method has not been investigated empirically.

Moreover, there are at least two problems related to the calculation of UAW:

- *Generalization of actors.* A generalization / specialization relation can be used to show that one or more actors are a special case of another actor (e.g., Senior Analyst can be a specialization of the Analyst actor). The number of actors in the use-case model impacts the calculated UCP. Therefore, generalized actors should be counted once only [17].

- *Additional actors.* Analyst can introduce additional actors, which is often dictated by the properties of a particular project [14]. For instance, analyst can introduce a goal-oriented actor, if the same goal is meaningful for many actors. (E.g., in the system supporting reviews of scientific papers, actors like Editor, Reviewer, and Author can be interested in obtaining the goal "read a paper". To enable this, a new actor called Reader can be introduced and associated with the "Read a paper" use case.)

If the UAW was rejected without decreasing accuracy of UCP it would simplify the method and would eliminate the problems presented above, related to the analysis of actors' complexity.

Thesis: UAW is negligible from the point of view of the effort estimation with UCP.

Investigation: In order to investigate the impact of UAW on the accuracy of effort estimation based on UCP, we performed the analysis described in Section 4 on the base variant of UCP (UCP-T) and the variant of UCP with UAW omitted (UCP-T no UAW). The values of both metrics are presented in Table 3. The results of the cross-validation analysis are presented in Table 5.

The observed values of MMRE were nearly the same for both variants of UCP and all of the data sets. The average MMRE was lower by 0.02 for UCP-T, however, the lower variability was observed for the UCP with UAW omitted. None of the null hypotheses about the equality of median values of MRE could be rejected.

The similar observation was made for the prediction quality. The average Pred(0.25) was higher by 0.05 in the case of UCP-T, but lower variability was observed for the variant of the UCP without UAW.

A minor tendency for overestimation was observed for both variants of UCP.

Summary: Based on the performed analysis, we observed that UAW had only minor impact on the accuracy of the effort estimation based on UCP.

Table 5: The accuracy of the effort estimation based on the different variants of UCP, Steps, Transactions, and TTPoints

		OLS regression			*Multiple regression*					*OLS regression*		
Size metric	Set	MMRE	Mean RE	Pred(0.25)	MMRE	Mean RE	Pred(0.25)	Factor	Set	MMRE	Mean RE	Pred(0.25)
UCP-S	All	0.40	-0.15	0.50	-	-	-	-	ADM	0.20	0.02	0.33
UCP-T		0.45	-0.15	0.43	-	-	-	-		0.32	0.03	0.33
UCP-T no UAW		0.44	-0.10	0.36	-	-	-	-		0.39	0.03	0.33
UUCP-S		0.58	-0.30	0.29	0.54	-0.31	0.57	T11		0.16	0.03	1.00
UUCP-T		0.59	-0.33	0.29	0.54	-0.23	0.50	T5		0.29	0.03	0.33
UUCW-S		0.56	-0.27	0.29	0.45	-0.20	0.43	T4		0.19	0.03	1.00
UUCW-T		0.56	-0.28	0.43	0.47	-0.22	0.43	T4		0.36	0.03	0.33
Steps		0.46	-0.20	0.36	0.50	-0.30	0.57	T11		0.22	0.03	0.33
Transactions		0.41	-0.11	0.64	0.53	-0.21	0.57	T5		0.44	0.02	0.33
TTPoints		0.25	-0.13	0.64	0.23	-0.06	0.64	T1		0.12	0.05	0.67
UCP-S	Ind	0.60	-0.28	0.29	-	-	-	-	EF1	0.43	-0.07	0.33
UCP-T		0.61	-0.34	0.14	-	-	-	-		0.52	-0.04	0.33
UCP-T no UAW		0.58	-0.31	0.14	-	-	-	-		0.46	-0.01	0.33
UUCP-S		0.78	-0.41	0.00	0.79	0.16	0.14	T5		0.35	-0.05	0.33
UUCP-T		0.78	-0.48	0.14	0.76	0.12	0.14	T5		0.43	-0.02	0.33
UUCW-S		0.75	-0.38	0.00	0.79	0.16	0.14	T5		0.32	-0.04	0.33
UUCW-T		0.74	-0.45	0.14	0.75	0.11	0.14	T5		0.39	0.00	0.33
Steps		0.68	-0.30	0.00	0.83	-0.26	0.14	T3		0.21	-0.02	0.67
Transactions		0.53	-0.23	0.43	0.86	0.07	0.14	T5		0.33	0.06	0.33
TTPoints		0.25	-0.10	0.57	0.30	0.00	0.57	T1		0.26	0.01	0.33
UCP-S	Uni	0.37	-0.15	0.71	-	-	-	-	EF2	0.19	0.05	0.50
UCP-T		0.50	-0.13	0.43	-	-	-	-		0.30	0.08	0.50
UCP-T no UAW		0.51	-0.07	0.43	-	-	-	-		0.37	0.10	0.25
UUCP-S		0.51	-0.24	0.43	0.35	-0.11	0.71	F3		0.15	0.05	1.00
UUCP-T		0.54	-0.24	0.57	0.28	-0.09	0.71	Team		0.27	0.08	0.50
UUCW-S		0.48	-0.20	0.57	0.24	0.00	0.57	Team		0.18	0.06	1.00
UUCW-T		0.49	-0.18	0.43	0.22	-0.09	0.71	Team		0.34	0.09	0.25
Steps		0.34	-0.14	0.57	0.14	-0.02	1.00	Team		0.20	0.06	0.50
Transactions		0.39	-0.05	0.43	0.14	-0.07	0.86	Team		0.44	0.12	0.25
TTPoints		0.27	-0.16	0.71	0.18	-0.04	0.71	F5		0.15	-0.02	0.50
UCP-S	Web	0.37	-0.18	0.50	-	-	-	-	TCF1	0.30	0.01	0.33
UCP-T		0.44	-0.17	0.58	-	-	-	-		0.47	-0.01	0.00
UCP-T no UAW		0.43	-0.12	0.42	-	-	-	-		0.57	-0.03	0.00
UUCP-S		0.55	-0.32	0.33	0.37	-0.07	0.50	T1		0.19	0.03	1.00
UUCP-T		0.58	-0.34	0.33	0.50	-0.08	0.42	Team		0.36	0.02	0.33
UUCW-S		0.52	-0.29	0.42	0.35	-0.07	0.50	T1		0.23	0.03	0.67
UUCW-T		0.54	-0.29	0.50	0.46	-0.08	0.58	Team		0.45	0.01	0.00
Steps		0.41	-0.21	0.42	0.35	-0.07	0.50	T1		0.27	0.02	0.33
Transactions		0.41	-0.12	0.50	0.59	-0.12	0.58	T5		0.59	0.02	0.00
TTPoints		0.25	-0.12	0.67	0.22	-0.04	0.58	T1		0.10	-0.07	0.67
UCP-S	Java	0.33	-0.12	0.75	-	-	-	-				
UCP-T		0.44	-0.11	0.50	-	-	-	-				
UCP-T no UAW		0.46	-0.05	0.50	-	-	-	-				
UUCP-S		0.57	-0.32	0.38	0.28	-0.09	0.88	T13				
UUCP-T		0.61	-0.35	0.50	0.60	0.15	0.50	Team				
UUCW-S		0.54	-0.29	0.38	0.26	-0.10	0.75	T13				
UUCW-T		0.56	-0.29	0.38	0.55	0.14	0.62	Team				
Steps		0.35	-0.17	0.50	0.20	-0.03	0.75	T1				
Transactions		0.37	-0.07	0.50	0.52	0.14	0.62	Team				
TTPoints		0.26	-0.17	0.75	0.18	-0.09	0.62	F5				

6. Adjustment Factors in Use Case Points

As presented in Section 2.2.3, there are 13 technical complexity factors and 8 environmental factors in the UCP method. Their role is to adjust UUCP to UCP.

There are at least two problems concerning assessment of these factors:

- *Lack of standardized (agreed) scale.* In the original UCP method the influence of each adjustment factor is assessed with the use of 0–5 ordinal scale (see Section 2.2.3). Although some rules for interpreting this scale have been proposed [18, 38], none of them have become an agreed standard (i.e., was reported to be used or validated in other studies).

- *Not verified weights of factors.* Each factor has its weight that reflects its general impact on the project's complexity. Values of the weights in the original method were proposed by Karner and his colleagues based on their experience. Little has been done so far to empirically evaluate their correctness.

It has already been reported that different people can provide different assessments of TCF and EF when they use the 0–5 ordinal scale [18, 38]. The same problem was observed for the projects presented in Section 3 (inner-team variability in assessment of TCF and EF for the projects is presented in Figure 2). Subjectivity in assessment of the adjustment factors can have a negative impact on the *consistency* of a historical projects database.

Taking into account problems with standardization of TCF and EF, a more general question arises – what is the impact of the adjustment factors on the accuracy of effort estimation with UCP?

Thesis: TCF and EF are negligible from the point of view of the effort estimation with UCP.

Investigation: To investigate the impact of adjustment factors on the accuracy of UCP, we compared the accuracy of the effort estimation based on the original UCP (UCP-T); unadjusted UCP (UUCP-T); and unadjusted use case weights (UUCW-T). The values of all metrics are presented in Table 3. The results of the cross-validation analysis are presented in Table 5.

The differences observed between the average values of MMRE for the original UCP and two variants that do not take into account the adjustment factors were minor. (They were in both cases equal to 0.04 in favour of UCP-T.) Lower variability was also observed for the original variant of UCP. However, we were not able to reject the null hypotheses about the equality of median values of MRE.

The prediction quality was also similar for all the compared variants of UCP. The highest average value of Pred(0.25) was observed for UUCP-T (the value for UCP-T was lower by 0.01 and for UUCW-T by 0.06).

The UCP-T performed slightly better for more heterogeneous subsets of projects. However, the observed improvement in the accuracy of effort estimation was still not impressive. Therefore, the question is whether all 21 factors are really necessary?

The previously discussed observations did not include the results obtained when the multiple regression model was used to estimate the effort, which could be treated as an extremely simplified version of the adjustment factors. When it was applied to UUCW-T and UUCP-T the prediction quality visibly increased. The average values of Pred(0.25) were higher for UUCW-T and UUCP-T than the value for UCP-T (by 0.03 and 0.08). Therefore in most cases, a single additional predictor added to the model was sufficient to obtain a similar accuracy of effort estimation as in the case of UCP incorporating 21 adjustment factors.

To further investigate the possibility of reducing the number of adjustment factors, we performed the *factor analysis*[6]. The goal of the analysis was to find groups of potentially overlapping factors within TCFs and EFs that could be substituted for more general factors.

The outcome of the analysis was that the initial adjustment-factors model including 21 factors could be substituted by a simpler model incorporating 4 technical complexity factors and 2 environmental factors.

In the case of technical complexity 4 factors that should be retained in the model accounted for 84% of the total variation in TCFs. The following "new" technical factors could be defined (we used the terminology defined in ISO 9126 [40]):

- *Efficiency* – this factor relates to the capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions. The main associated TCFs are "performance" (T2), "end-user efficiency" (T3), "complex processing" (T4), and "easy to use" (T7).

- *Operability* – this factor defines how easy it is to operate and control the system. The associated

[6]Factor analysis is a technique used to reduce the number of dimensions in the data. It assumes that the observed dependant variables are linear combinations of some underlying (hypothetical or unobservable) independent variables – called "factors" [39].

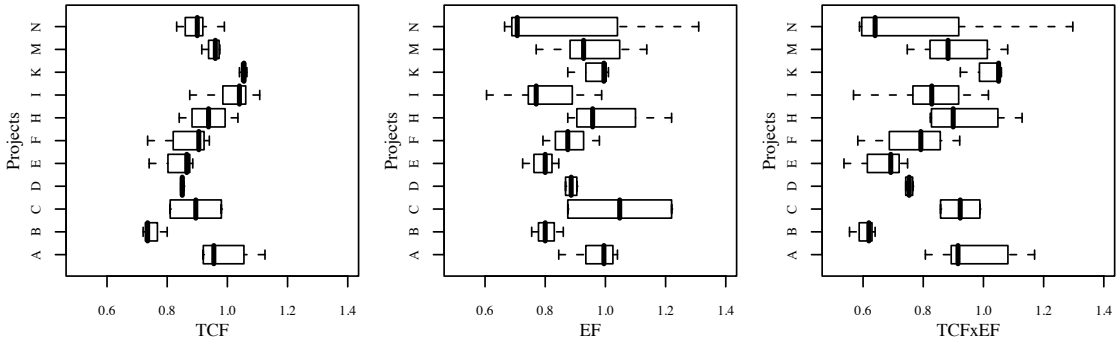


Figure 2: Box plots presenting differences in assessments of TCF and EF (and their cumulative value TCF×EF) based on the conducted surveys within the project teams (projects with only one response were not included to the plot)

factors are "easy to install" (T6), "portable" (T8), "security features" (T11), and "special training required" (T13). The difficulties in the installation and configuration of the system are reflected by factors T6 and T8 (requirements related to portability could imply that the system will have to be installed in different environments). The second aspect relates to the difficulties in operating and controlling the system (i.e., cumbersome security procedures or special training required).

- *Maintainability* – it is the capability of the software product to be modified. The main associated factors are "reusable code" (T5), and "easy to change" (T9).

- *Interoperability* – it is defined as the capability of the software product to interact with one or more specified systems. The associated factors are "concurrent" (T10), and "access to third parties" (T12).

A similar analysis was performed by Lokan [3] for *general system characteristics* (GSC) in FPA. The initial GSC model containing 14 factors was reduced to 5 factors in that study. Although the technical adjustment factors are slightly different than GSC (TCFs were adapted from MK II FPA), three of the identified factors were common for both studies (efficiency, operability, and interoperability).

The second set of adjustment factors is specific to UCP. Based on the results of the factor analysis we observed that the initial model including 8 factors could be substituted by a simpler model containing only 2 more general factors (these factors accounted for 52% of the total variation in the EFs):

- *Team experience* – it relates to the knowledge and skills of the development team. The associated factors are "familiarity with the standard process" (F1), "application experience" (F2), "object-oriented experience" (F3), "lead analyst capability" (F4)", and "difficult programming language" (F8). A minor correlation was also observed for the environmental factor "motivation" (F5).

- *Team cohesion* – it accounts for the level of collaboration between the project's stakeholders. The associated factors are "motivation" (F5), "stable requirements" (F6), and "part-time workers" (F7). The last factor F7 is negatively correlated – the more part-time workers there are, the less team cohesion is observed (this factor has also a negative weight in the original EFs model). The "team cohesion" is also a scale factor in the COCOMO II effort estimation method [41].

The high correlation between the factors F1-F4 is unsurprising. For instance, if the team is experienced in software development (F3) it seems also highly probable that its members are experienced designers (F4). (The factor F4 is, however, irrelevant if the team does not use object-oriented programming languages.)

Summary: Adjustment factors have been broadly discussed and criticized in the context of FPA [3, 4, 6, 42]. (The main reported problems related to the inadmissible operations on scales and the minor impact of the adjustment factors on the accuracy of the effort estimation based on FPA.)

In this study we observed a similar phenomenon related to adjustment factors in UCP. We observed the minor influence of these factors on the accuracy of the effort estimation based on UCP. The second observation was that some of the adjustment factors are clearly

overlapping. (The total number of 21 adjustment factors considered in UCP could be limited to 4 technical complexity factors and 2 environmental factors.)

7. Use Case Points – Steps vs. Transactions

As a use-case transaction is a set of activities in use-case-scenarios, which is either performed entirely, or not at all [9], a single transaction might be a part of a use-case scenario, a step, or even a single phrase.

Some authors suggested that counting transactions is equivalent to counting *steps* [18, 43], which in our experience is not always true. However, if steps could be counted instead of transactions, it would simplify the UCP method (as identification of transactions can be a difficult task [21][7]).

Thesis: The *value* of UCP calculated based on steps is the same as if calculated based on transactions.

Investigation: The *exact* values of UCP calculated based on steps and transactions differed significantly[8].

One of the reasons for such discrepancy is that steps are elements of use-case syntax (and not necessary semantics). Therefore, a single use case can be easily written using a different number of steps, without changing its abstraction level, or even phrases [23].

For instance, in the considered set of projects, the ratio between the number of steps and the number of transactions ranged from 1.50 to 4.04 (mean = 2.83, and SD = 0.75). This is because, some authors may follow the Jacobson's suggestion that each step of a use case should constitute a transaction [11, 23]; other authors may express most of transactions with the use of two steps (actor's request, and system response). Unfortunately, many other approaches are also possible (or a mixture of different approaches depending on the functionality being described).

The difference between the number of steps and the number of transactions impacts the value of UCP. The observed ratio between UCP-S and UCP-T for the considered projects ranged from 1.05 to 2.12 (mean = 1.73, and SD = 0.28).

Although, the values of UCP-S and UCP-T differed significantly, a strong correlation between them was observed (Pearson's product-moment coefficient $r = 0.99$, p-value < 0.001). Therefore, one can try to predict the value of UCP-T based on the value of UCP-S (e.g., by constructing a linear regression model as presented in Figure 3). The accuracy of such predictions depends on similarities (or differences) in use-case writing style between the projects. For instance, projects B and E were developed by the same enterprise, however, the ratios between UCP-S and UCP-T were 1.1 and 1.8. More convergent writing styles were observed in projects H, I, and K, which were also developed by a single organization (the ratios were 1.7, 2.0, and 1.8).

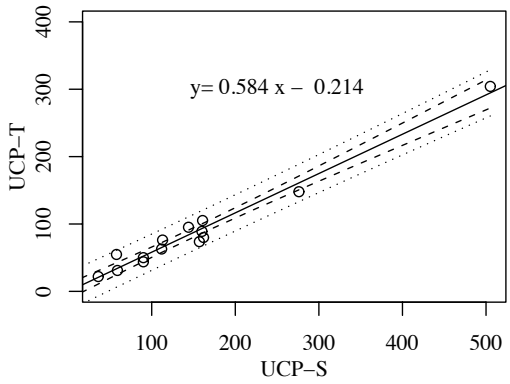


Figure 3: Scatter plot with linear regression presenting dependency between UCP-S and UCP-T for the projects introduced in Section 3 (dashed lines – 95% confidence interval; dotted lines – 95% prediction interval)

Summary: The values of UCP calculated based on steps and transactions might not be the same, therefore, it is recommended to not mix them together within a single historical database.

However, if an organization is forced to use historical data concerning UCP coming from different sources (without the access to the requirements specifications, e.g., data published in scientific papers), where UCP is calculated based on steps, for some of the projects, and based on transactions, for the rest of them, it may consider using regression model to predict the values of UCP, and unify the measurements within the data set (e.g., multiply the values of UCP-S by 0.584 according to the model presented in Figure 3 to obtain the approximated values of UCP-T). Unfortunately, such conversion will introduce an error, which value depends on the level of differences between the writing style of use cases (which cannot be assessed without the access to the requirements specification documents).

In addition, it is important to provide information whether UCP was calculated based on steps or transactions when reporting any study concerning UCP (which

[7] Six experts counted transactions in 30 use cases. The highest obtained total number of transactions was nearly 2 times greater than the smallest one.

[8] Two-tailed Wilcoxon signed-rank test with significance level α set to 0.05 (p-value = 0.001); calculated effect size: $d = 0.65$; retrospective power: $(1\text{-}\beta) = 0.55$.

is a seldom observed practice).

Although, the values of UCP calculated based on steps and transactions are not equivalent, the question arises about the equivalence of the accuracy of effort estimation based on UCP-S and UCP-T.

Thesis: The *accuracy* of UCP calculated based on steps is not worse than if calculated based on transactions.

Investigation: In order to investigate whether the choice of steps or transactions to calculate UCP impacts the accuracy of effort estimation, we compared the accuracy of effort prediction based on the corresponding variants of UCP (also UUCP and UUCW) calculated based on steps and transactions.

The observed differences in the accuracy of effort estimation between the corresponding variants of UCP (e.g., UCP-S vs. UCP-T) were not significant. However, all variants of UCP calculated based on steps provided on-average more accurate predictions. The differences between the average values of MMRE were 0.06 (UUCW), 0.07 (UUCP), and 0.10 (UCP) – all in favour of the variants calculated based on the number of steps. However, lower variability of MMRE between the data sets was observed for the variants of UCP calculated based on the number of transactions.

The similar observation was made for the prediction quality. The differences between the average values of Pred(0.25) were 0.11 (UCP), 0.12 (UUCW), and 0.14 (UUCP) – in favour of variants based on steps. Again, lower variability was observed for UUCP and UUCW calculated based on transactions (in the case of UCP the variability was slightly lower for UCP-S). We also observed that UCP-S provided systematically more accurate estimates than UCP-T. (It was not observed, however, in the case of UUCP and UUCW.)

The main reason for such visible on-average differences in the accuracy between the step-based and the transaction-based variants of UCP was that the variants based on steps performed visibly better for more homogeneous data sets (i.e., ADM, EF1, EF2, and TCF1).

All the variants of UCP (no matter if calculated based on steps or transactions) had a minor tendency to overestimate the effort, however, in the case of more homogeneous sets the mean RE was usually close to zero.

Summary: The observation made in this study was that the variants of UCP calculated based on steps provided not worse (or even slightly better) accuracy of effort estimation than the corresponding variants calculated based on transactions.

Therefore, it seems that if an organization has a standardized style of writing use cases for all the projects, it may consider simplifying UCP by calculating its value based on steps in use cases.

8. Counting Steps Instead of UCP

It seems that UCP can be simplified by using steps to calculate UUCW instead of transactions. The question arises, whether it can be further simplified to count just the total number of steps in all use cases?

Thesis: The *Steps* metric, which is the total number of steps in all use cases describing the system, can be used to estimate effort with similar accuracy to UCP.

Investigation: To investigate the thesis, we compared the accuracy of estimation based on UCP calculated based on steps (UCP-S, UUCP-S, and UUCW-S) with the Steps metric.

The observed accuracy of effort estimation for UCP-S and Steps were on-average similar (no significant differences were observed). The average values of MMRE were the same for both metrics, however, a slightly lower variability was observed for UCP-S. In addition, UCP-S provided better on-average prediction quality than Steps (by 0.06). The Steps metric performed slightly better for more homogeneous subsets and when the multiple regression model was used instead of the single regression model.

In comparison between Steps and two other step-based metrics UUCP-S and UUCW-S, lower on-average MMRE was observed for Steps (the differences between the average MMREs were 0.05 and 0.07). The null hypotheses about the equality of median values of MRE could be rejected for Uni, Web, and Java sets. However, both UUCP-S and UUCW-S provided slightly higher on-average values of Pred(0.25), mainly because they performed visibly better than Steps in the case of more homogeneous subsets.

Summary: Counting the total number of steps instead of UCP could simplify the effort estimation procedure. However, one should be aware that this technique could be exposed to the problems related to the differences in use-case writing style and the differences in abstraction levels of use cases (see Sections 7 and 9).

9. Long Live Transactions!

If the variants of UCP calculated based on the number of steps and the Steps metric itself could provide effort estimates with the similar accuracy to the UCP calculated based on the number of transactions, then, are transactions still worth considering?

Thesis: Use-case transactions can be used to provide effort estimation with better accuracy than UCP calculated based on steps.

Investigation: Use-case transactions are supposed to express the semantics of interaction between actors. Therefore, they should be less prone to different author writing styles (or different levels of abstraction in use cases).

In this study, we were not able to assess the scale of influence of use-case writing style on the accuracy of the effort estimation based on step-based size metrics. However, to present the potential problems of using steps instead of transactions, we compared two projects that have visible differences in writing style of use cases, but their actual effort was similar (projects B and I). If the post productivity factors of these projects were cross-used between them to estimate the effort, the ratios between the MRE for UCP-S and UCP-T would be 1.72 (B based on I) and 3.31 (I based on B). Therefore, UCP-T would be a better choice than UCP-S if there is a threat that use cases stored in the historical database differ in respect to the level of abstraction or writing style.

Recently two new functional size measures that are based on the concept of use-case transactions were proposed, i.e., Transactions [20], and TTPoints [29].

The Transactions metric is defined as the total number of transactions that can be identified in all use cases describing the system under development.

The TTPoints metric is also based on the number of transactions, but it includes additional information about the semantics of transactions (twelve semantic transaction types have been identified so far[9]).

The value of TTPoints is calculated according to Equation 12. The main difference between the Transactions and TTPoints is concerning the way they assess complexity of a single transaction. In the case of Transactions metric, all transactions are treated as equally complex. However, in TTPoints the complexity of a single transaction is calculated by multiplying together the weight assigned to its semantic type, the number of actors interacting with the system under development (within the transaction), and the number of objects being processed within the transaction.

$$TTPoints = \sum_{i=1}^{n} TT\_Weight_i \times Objects_i \times Actors_i \qquad (12)$$

where

- n is the number of semantic transactions identified in the requirements specification;
- TT_Weight_i is the weight assigned to the type of the i-th transaction, which corresponds to the number of *core actions* (actions that are necessary to preserve the semantic sense of a given type of transaction, e.g., a "provision of data" is a core action for the "create" type of transaction);
- $Objects_i$ is the number of domain objects processed by the i-th transaction;
- $Actors_i$ is the number of collaborating actors in the i-th transaction (other than actor representing the system under development).

The effort estimation capabilities of Transactions and TTPoints metrics were preliminary verified [29, 44]. The observations were made that they can be effectively used to estimate effort at the early stages of projects.

In order to investigate the differences between the accuracy of transactions-based and step-based size metrics, we compared prediction accuracy of Transactions and TTPoints with the accuracy of other size metrics calculated based on the number of steps.

In the performed comparison, the Transactions metric provided on-average a slightly worse prediction accuracy than the measures based on the number of steps. The differences between the average values of MMRE and average Pred(0.25) ranged from 0.03 to 0.10 – all in favour of step-based measures. However, lower variability of MMRE was observed for Transactions in all cases. (The variability of Pred(0.25) was not lower for Transactions only in the comparison with UCP-S.)

The observed differences in favour of step-based measures were mainly due to extremely poor accuracy of Transactions in the case of the sets ADM, EF2, and TCF1. If only the bigger sets of projects were considered (i.e., All, Ind, Uni, Web, and Java) Transactions would be on-average more accurate than UUCW-S and UUCP-S, and equally accurate to UCP-S and Steps (however, still providing estimates with lower variability of prediction error).

The second considered size metric – TTPoints provided estimates with better prediction accuracy than the size metrics calculated based on the number of steps. The differences observed between the average values of MMRE ranged from 0.14 to 0.22 – all in favour of TTPoints. A similar observation was made for the differences between the values of Pred(0.25), which ranged from 0.08 to 0.15.

[9]The semantic transaction-types defined in [29]: Create, Retrieve, Update, Delete, Link, Delete Link, Asynchronous Retrieve, Dynamic Retrieve, Transfer, Check Object, Complex Internal Activity, and Change State.

The differences in the median values of MREs between TTPoints and three other step-based size measures UUCP-S, UUCW-S, and Steps were significant for the sets: All, Ind, and Web (in the case of comparison with UUCW a significant difference was also observed for the Java set). The differences in median values of MREs between TTPoints and UCP-S were significant only for the Ind set.

There are few potential reasons that could explain why TTPoints performed better than measures based on counting the number of steps.

First of all, the semantic transaction types that are used in TTPoints are based on another definition of use-case transaction provided by Diev [14], which is similar to the definition of elementary process in FPA. It states that a use-case transaction is the smallest unit of activity that is meaningful from the actor's point of view, which is self-contained and leaves the business of the application being sized in a consistent state. Therefore, this definition of transaction is oriented towards achieving goals meaningful for actors. The semantic transaction types define a catalogue of such meaningful goals that can be found in use cases.

For example, in use case presented in Figure 1 one can identify a single transaction which semantic type is called "create". If the stimulus-response transaction definition was used, two transactions would be identified in the same use case (or even three if one considers an alternative ending of the same transaction as a new one).

Therefore the question arises what is the advantage of using the semantic-types approach to transaction identification? For instance, if in the same use case (presented in Figure 1) the action of submitting the "paper submission form" was divided into two sub-actions – providing general information about the paper and separately its content, then, one more transaction would be identified if the stimulus-response definition is followed. For semantic transaction types, such use cases would still contain a single "create" transaction.

The weights of semantic transaction types in the TTPoints method are based on the number of so-called "core actions" (actions that are crucial for preserving the meaning of a transaction type). This weight could be understood also as the minimum number of actions that have to be performed for a given type of transaction.

TTPoints also includes information about domain objects that appear in use-case scenarios. In previous studies [20, 29], it was observed that the size metric defined as the total number of objects found in use-case scenarios does not provide an accurate prediction when it is used for effort estimation.

Another observation was that domain objects differ in importance [29] (e.g., some of them are the subjects of many use cases). Therefore, the total number of domain objects is not a good candidate for being a standalone size measure. In TTPoints the number of domain objects is counted for each transaction (not for the whole requirements specification). In most cases, a single domain object is processed per transaction, however, a complex transaction can process more objects. The same relates to actors. It is seldom observed that there is more than one actor interacting with the system within a single transaction. However if it happens, the transaction is assumed to be more complex in TTPoints (e.g., some additional systems are taking part in the transaction).

Summary: The first observation that was made based on the results of the analysis was that the accuracy of the effort estimation based on Transactions was comparable to the accuracy observed for UCP-S or Steps. However, the main advantage of Transactions was lower variability of prediction error. In addition, both Steps and Transactions are simple measures that are mathematically valid, in contrast to UCP.

The second investigated size measure was TTPoints, which seemed to outperform all step-based metrics considered in this study. However, the procedure for calculating TTPoints seems to be slightly more complex than counting the total number of transactions or steps.

10. Threats to Validity

There are several threats to the validity of this study. Some of them regard *internal* or *conclusion validities*, which in this study relates to the quality and relevance of the collected data and application of statistical techniques. Other threats refer to *external validity*, which describes problems with generalization of observations made based on the considered data set for other projects.

10.1. Internal Validity

The projects analyzed in this study were developed by different organizations. Therefore, the maturity of development processes could affect the recorded effort. Moreover, in the case of some projects we did not have access to the detailed data regarding the types of activities included in the recorded effort.

Additional problems relate to the objectiveness of transaction-identification in use cases. In order to preserve consistency of this process, the same person identified all transactions. To investigate the potential problems related to reliability of transaction-identification,

we asked an external expert to review the transaction counts for the random sample containing 20% of all use cases. However, there still might be some inconsistencies in the results of transaction-identification process due to the different styles of writing use cases between the projects.

10.2. Conclusion Validity

There are also several threats to validity that relate to the significance of observation that were made. Although we performed statistical testing of hypotheses, one should treat the obtained results with caution.

First of all, due to the large number of direct comparisons a multiple testing procedure should be used – which could be done, e.g., by applying the Bonferroni correction (i.e., dividing the initial value of α by the total number of direct comparisons). However, the more important issue in the context of this study relates to statistical power[10] of the performed tests ($1–\beta$). It is important because whenever we investigated a potential way of simplifying UCP, it was sufficient for the simplified variant of the method to perform at least as good as the original one. Therefore, if the null hypothesis about the equality of median MREs between two compared variants of UCP was really true, it would also support the thesis that UCP can be simplified, without limiting its prediction accuracy.

To investigate this issue, we performed a *sensitivity analysis* [45] in order to find out what is the minimal *effect size* that could be detected with the assumed power of $1–\beta = 0.80$, for the size of samples ranging from 3 to 14, and significance level $\alpha = 0.05$. The minimal effect size that could be detected for a sample of 14 projects would be 0.83, which is a "large" effect size according to Cohen [46]. For the smallest subsets containing 3 projects such effect size could be detected if assumed statistical power of the test was reduced to 0.13. The observed effect size[11] was on-average "medium" ($\overline{d} = 0.48$); the "small" effect size was observed for 34% of the comparisons ($d \leq 0.2$); the effect size between "small" and "large" was observed for 47% of the comparisons ($0.2 < d < 0.8$); and "large" effect size was observed for the remaining 19% of the comparisons ($d \geq 0.8$).

Therefore, whenever we were not able to reject the null hypothesis, it could mean that either the null hypothesis was really true or the effect size was too small to be detected in this study.

10.3. External Validity

The main external threat to validity relates to the size of the data set which limits the strength of conclusions that can be made. However, the considered data set is still one of the largest data sets that have been published and used to analyze the UCP method[12].

Another threat to validity could be the heterogeneity of the analyzed set of projects. In order to mitigate that problem we decided to perform the analysis on the main set and its subsets that seemed to be homogeneous in some sense. In addition, we performed the same cross-validation procedure for all combinations of the main set containing from 3 to 14 projects. The observation that was made based on the analysis of the distributions of MMRE for the considered size metrics (see Figure 4) is that the different compositions of the main data set did not influenced visibly the findings of this study.

Another limitation related to external validity of this study is that the data set contained mainly data from the projects aimed to develop data-intensive systems (probably the most outlying project, in that context, was the project B). Therefore, the observations made in this study could be generalized mainly to this kind of systems. (However, the relation between the structure of use cases and the complexity of the system for different types of applications still remains unknown.)

11. Conclusions

In this study the construction of the Use Case Points method was investigated based on the data collected from 14 software projects. The aim was to search for potential ways to simplify the effort estimation based on use cases.

The first potential approach to simplify UCP is to reject UAW. We observed that in the case of the considered data set it had only minor impact on the accuracy of effort estimation based on UCP.

[10] The power of a statistical test ($1–\beta$) is the probability that the statistical test will reject a false null hypothesis.

[11] We performed the retrospective power analysis, however, this technique is often criticized because it is based on the questionable assumption that the sample effect size is identical to the effect size in the population from which it was drawn [47].

[12] Some of the largest data sets that were *published* (together with basic information about the projects, i.e., UCP, actual effort, context): Frohnhoff and Engles [38] – 15 industrial projects (728–136 320h), Ribu [18] – 12 projects (2 industrial 10 043–13 933h, and 10 students' 232–595h). Larger data sets were also reported to be used as a basis for the analysis of UCP (however without providing in-depth information about the projects): Arnold and Pedross [16] – 23 projects, Carrol [48] – more than 200 projects, Diev [49] – 30 projects.

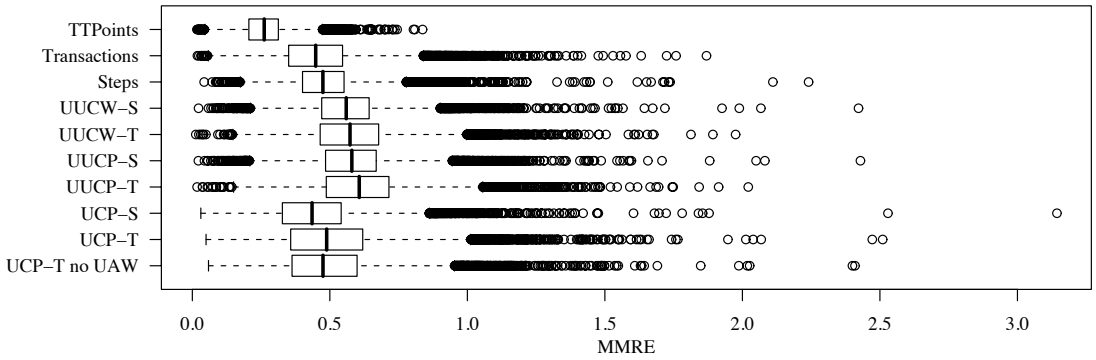


Figure 4: Box plots presenting distributions of the MMRE calculated for the subsets of the main data set (16 278 combinations of the main set that contained from 3 to 14 projects)

The next observation was that the adjustment factors used in UCP did not provide a significant improvement in the accuracy of effort estimation. In addition, the model of 21 adjustment factors seemed to be superfluous. In most cases, a single additional predictor added to the regression model was sufficient to provide estimates with similar accuracy as when TCF and EF were used. Finally, according to the results of the performed factor analysis the number of adjustment factors might be reduced to 2 environmental factors and 4 technical complexity factors.

Another way of simplifying the UCP-based effort estimation is to calculate the value of UCP based on the number of steps instead of transactions. We observed that the step-based variant of UCP provided not worse effort prediction than calculated based on the number of transactions in use cases. However, the exact values of both variants of UCP were not the same. In addition, the extremely simplified variant of UCP which is counting the total number of steps in all use cases provided similar prediction accuracy to other variants of UCP calculated based on steps.

We also investigated the accuracy of two recently proposed size metrics Transactions [20] and TTPoints [29].

The observation was made that the Transactions metric provided a similar effort prediction as the different variants of UCP calculated based on the number of transactions. (The step-based size metrics provided slightly better on-average accuracy than Transactions, however, with higher variability.) The TTPoints metric [29] (which extends the idea of Transactions) provided slightly better prediction accuracy than all considered variants of UCP. In addition, the measures based on steps seemed to be more sensitive to the differences in use-case writing style than transactions.

The main limitations of the observations made in this study are related to the small size of the considered set of projects and its heterogeneity.

Acknowledgements

We would like to thank Jakub Jurkiewicz for performing the review of transaction-counts in use cases, and Sylwia Kopczyńska for her valuable remarks.

We also thank the anonymous reviewers for their comments, which resulted in substantial improvements to this work.

This research project operated within the Foundation for Polish Science Ventures Programme co-financed by the EU European Regional Development Fund.

References

[1] A. Albrecht, Measuring application development productivity, Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium (1979) 83–92.

[2] C. Lokan, An empirical study of the correlations between function point elements, in: Software Metrics Symposium, 1999. Proceedings. Sixth International, IEEE, 2002, pp. 200–206.

[3] C. Lokan, An empirical analysis of function point adjustment factors, Information and Software Technology 42 (9) (2000) 649–659.

[4] R. Jeffery, J. Stathis, Function point sizing: Structure, validity and applicability, Empirical Software Engineering 1 (1) (1996) 11–30.

[5] B. Kitchenham, K. Kansala, Inter-item correlations among function points, in: Software Metrics Symposium, 1993. Proceedings., First International, IEEE, 2002, pp. 11–14.

[6] B. Kitchenham, The problem with function points, IEEE Software 14 (2) (1997) 29–31.

[7] C. R. Symons, Software Sizing and Estimating: Mk II FPA (Function Point Analysis), John Wiley & Sons, Inc., New York, NY, USA, 1991.

[8] A. Abran, J. Desharnais, S. Oligny, D. St-Pierre, C. Symons, The COSMIC Functional Size Measurement Method v3.0.1, Measurement Manual (May 2009).
[9] G. Karner, Metrics for objectory, Master's thesis, University of Linköping, Sweden (1993).
[10] I. Jacobson, Object-oriented development in an industrial environment, ACM SIGPLAN Notices 22 (12) (1987) 183–191.
[11] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard, Object-Oriented Software Engineering: A Use Case Driven Approach, Addison Wesley Longman, Inc, 1992.
[12] C. Neill, P. Laplante, Requirements engineering: the state of the practice, Software, IEEE 20 (6) (2003) 40–45.
[13] M. Braz, S. Vergilio, Software effort estimation based on use cases, Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06)-Volume 01 (2006) 221–228.
[14] S. Diev, Use cases modeling and software estimation: applying use case points, ACM SIGSOFT Software Engineering Notes 31 (6) (2006) 1–4.
[15] P. Mohagheghi, B. Anda, R. Conradi, Effort estimation of use cases for incremental large-scale software development, Proceedings of the 27th international conference on Software engineering (2005) 303–311.
[16] M. Arnold, P. Pedross, Software size measurement and productivity rating in a large-scale software development department, Proceedings of the 20th ICSE (1998) 490–493.
[17] B. Anda, H. Dreiem, D. Sjøberg, M. Jørgensen, Estimating software development effort based on use cases-experiences from industry, Fourth International Conference on the UML (2001) 487–504.
[18] K. Ribu, Estimating object-oriented software projects with use cases, Master's thesis, University of Oslo, Department of Informatics (2001).
[19] J. Ouwerkerk, A. Abran, An evaluation of the design of use case points (UCP), in: A. Abran, R. Dumke, M. Ruiz (Eds.), Proceedings of the International Conference on Software Process and Product Measurement MENSURA 2006, Publish Service of the University of Cádiz www.uca.es/publicaciones, 2006, pp. 83–97.
[20] G. Robiolo, R. Orosco, Employing use cases to early estimate effort with simpler metrics, Innovations in Systems and Software Engineering 4 (1) (2008) 31–43.
[21] M. Ochodek, 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 CEE-SET 2007, Vol. 5082 of LNCS, Springer Verlag, 2008, pp. 55–68.
[22] S. Adolph, P. Bramble, A. Cockburn, A. Pols, Patterns for Effective Use Cases, Addison-Wesley, 2002.
[23] A. Cockburn, Writing Effective Use Cases, Addison-Wesley Boston, 2001.
[24] M. Bundschuh, C. Dekkers, The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement, Springer-Verlag New York, Inc., 2008.
[25] N. E. Fenton, S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, PWS Publishing Co., Boston, MA, USA, 1998.
[26] G. Schneider, J. Winters, Applying Use Cases: A Practical Guide, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1998.
[27] J. Ward Jr, Hierarchical grouping to optimize an objective function, Journal of the American Statistical Association 58 (301) (1963) 236–244.
[28] A. Žiberna, N. Kejžar, P. Golob, A comparison of different approaches to hierarchical clustering of ordinal data, Metodološki zvezki 1 (1) (2004) 57–73.
[29] M. Ochodek, J. Nawrocki, Enhancing use-case-based effort estimation with transaction types, Foundations of Computing and Decision Sciences 35 (2) (2010) 91–106.
[30] J. Cohen, Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit, Psychological bulletin 70 (4) (1968) 213–220.
[31] J. Landis, G. Koch, The measurement of observer agreement for categorical data, Biometrics 33 (1) (1977) 159–174.
[32] E. Alpaydin, Introduction to Machine Learning (Adaptive Computation and Machine Learning), The MIT Press, 2004.
[33] J. Miles, M. Shevlin, Applying Regression & Correlation: A Guide for Students and Researchers, Sage Publications Ltd, 2001.
[34] D. Wright, K. London, Modern Regression Techniques Using R: A Practical Guide for Students and Researchers, Sage Publications Ltd, 2009.
[35] R. Jeffery, M. Ruhe, I. Wieczorek, A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data, Information and Software Technology 42 (14) (2000) 1009–1016.
[36] M. Jørgensen, K. Moløkken-Østvold, Reasons for software effort estimation error: Impact of respondent role, information collection approach, and data analysis method, IEEE Transactions on Software Engineering 30 (12) (2004) 993–1007.
[37] S. D. Conte, H. E. Dunsmore, V. Y. Shen, Software Engineering Metrics and Models, Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1986.
[38] S. Frohnhoff, G. Engels, Revised Use Case Point method – Effort estimation in development projects for business applications, in: I. Schieferdecker, S. Goericke (Eds.), Setting Quality Standards: Proceedings of the CONQUEST 2008, dpunkt.verlag, 2008, pp. 15–32.
[39] J. Kim, C. Mueller, Factor analysis: Statistical methods and practical issues, Sage Publications, Inc, 1978.
[40] ISO/IEC, ISO/IEC 9126-1: Software engineering – Product quality – Part 1: Quality model (2001).
[41] B. W. Boehm, B. K. Clark, E. Horowitz, A. Brown, D. Reifer, S. Chulani, R. Madachy, B. Steece, Software Cost Estimation with COCOMO II, Prentice Hall, Upper Saddle River, NJ, USA, 2000.
[42] D. Jeffery, G. Low, M. Barnes, A comparison of function point counting techniques, IEEE Transactions on Software Engineering 19 (5) (1993) 529–532.
[43] R. Clemmons, Project estimation with use case points, CrossTalk–The Journal of Defense Software Engineering 19 (2) (2006) 18–22.
[44] G. Robiolo, C. Badano, R. Orosco, Transactions and paths: two use case based metrics which improve the early effort estimation, in: Third International Symposium on Empirical Software Engineering and Measurement, 2009, pp. 422–425.
[45] F. Faul, E. Erdfelder, A. Lang, A. Buchner, G*Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences, Behavior Research Methods 39 (2) (2007) 175–191.
[46] J. Cohen, Statistical power analysis, Current Directions in Psychological Science 1 (3) (1992) 98–101.
[47] B. Zumbo, A. Hubley, A note on misconceptions concerning prospective and retrospective power, Journal of the Royal Statistical Society – Series D: The Statistician 47 (2) (1998) 385–388.
[48] E. Carroll, Estimating software based on use case points, Conference on Object Oriented Programming Systems Languages and Applications (2005) 257–265.
[49] S. Diev, Software estimation in the maintenance context, ACM SIGSOFT Software Engineering Notes 31 (2) (2006) 1–8.