

Revised Use Case Point Method - Effort Estimation in Development Projects for Business Applications

Stephan Frohnhoff,
sd&m AG, software design & management, Berliner Str. 76, D-63065 Offenbach,
Germany

Gregor Engels,
sd&m AG, software design & management, Berliner Str. 76, D-63065 Offenbach,
Germany
s-lab, University of Paderborn, Warburger Str. 100, D-33098 Paderborn, Germany

Abstract

Rapid and precise effort estimation of software development projects is crucial in IT industry. In a case study, the Use Case Point (UCP) method was applied to 15 commercial software development projects. The estimated efforts were compared with the incurred project efforts. We measured a standard deviation of 42 %. This is not acceptable for industrial usage. Therefore, we propose appropriate improvements of the Use Case Point method leading to significantly higher estimation accuracy with only 20 % standard deviation. The contribution of this paper is a detailed description of the improved Use Case Point method.

1 Introduction

Effort estimation of business application projects in software industry is commonly based on a rough specification resulting from the early inception phase. Very often, these specifications are based on UML [3, 16] and could be characterized as use case oriented.

Therefore, at sd&m (a company of Capgemini), we are looking for an estimation method that directly calculates efforts by counting use cases.

From our point of view, an estimation method based on Use Case Points (UCP) [10, 13] fulfils most suitable these requirements of a use case orientated top-down estimation method. Such an approach is easy to learn for experienced software engineers and results in very efficient effort estimations: According to our experiences a project sized 300 person-days (approx. 25.000 lines of code) can be estimated within one hour.

The original UCP method was developed by Karner in an industrial context [10]. So far, only little research has been done to evaluate and eventually improve the method [2].

In a case study, the UCP method was applied to 15 software development projects and the estimated effort was compared with the incurred efforts after project close. In a first step, the original UCP method (summarized in section 2) has been applied to several projects. A tool kit has been developed for its use in an industrial environment [8] (section 3 and 4).

The original UCP method reveals an insufficient standard deviation for industrial usage. We analysed that the environmental cost factors calibrating the complexity of the project setting are not state of the art and have to be improved. This was also observed by functional size measurement methods like the well established function points analysis [1], its enhancements [12] and COCOMOII2000 [4], which have accordingly improved these cost factors. We follow the same path and present here an enhanced Use Case Points method. The appropriateness is evaluated with a different set of projects (section 5). We show that the enhanced UCP method is a provable improvement of the original one (section 6).

2 Background

The original Use Case Point method [10] assigns points to each use case by its complexity (simple = 5 points, normal = 10 points, complex = 15 points) and accumulates all points. Actors are classified by type (API = 1 point, protocol interface = 2 points, user interface = 3 points). These points are added to the last preceding points. The total is named *Unweighted Use Case Points* (UUCP).

The complexity caused by project organisation and project environment is estimated by an environmental factor with 8 influencing variables E_i , e.g. familiarity with UML, experience with application development or with object orientation, etc. Each variable is rated from 0 (= no experience in the subject) to 5 (= expert), multiplied by a weight g_i (with a range from -1 to 2) and summed up to the *Environmental Factor* EF. This is standardized to 1.0 (= all values rated to 3 points) with a range from 0.425 to 1.70:

$$EF := 1.4 - \sum_{i=1..8} (E_i * g_i * 0.03)$$

Complexity caused by the expected technical environment is estimated very similarly by 13 influencing variables T_i . They are weighted (g_i), added, standardized to 1.0 and accumulated to the *Technical Complexity Factor* (TCF) with a range from 0.6 to 1.3.

The *weighted Use Case Points* (UCP) are defined as

$$UCP := UUCP * TCF * EF.$$

PF defines a Productivity Factor in the range of 20 to 40 person-hours per UCP and has to be calibrated to the productivity of the corresponding organisation. The *estimated project effort* according to Karner is then defined as

$$PE_{\text{Karner}} := UCP * PF.$$

3 Case study setting

This case study relies on 15 commercial software development projects in various sectors. The applications were developed by the IT-company sd&m AG in the role as a custom software supplier.

First we had to define how to measure the quality of an estimation method in general. In our opinion the best way is to compare its results with the incurred efforts after project close. The deviation would then give a measurement for the quality of the estimation method.

The 15 projects of this case study are based on a rough system specification coming from an inception phase according to RUP [11]. From praxis we know that in general specifications reveal insufficiencies during project run (i.e. faults, inconsistencies and uncertainties) resulting in an estimation error E_s . E_s is independent of the quality of the estimation method but influences the total deviation between UCP estimation and incurred effort after project close. In order to diminish E_s we have adjusted the specification after project close according to the known specification insufficiencies. The UCP estimations of this case study are based on these adjusted specifications.

Table 1. Project data (in person-hours)

sector	A: project effort [h]	B: estimated effort [h]	deviation (B-A)/A
Apparel Industry	728	1.205	66%
Automotive 1	15.500	11.667	-25%
Automotive 1	136.320	114.023	-16%
Finance 1	2.992	1.002	-67%
Finance 2	3.680	3.301	-10%
Insurance 1	4.800	2.115	-56%
Logistics 1	944	1.406	49%
Logistics 2	2.567	1.751	-32%
Logistics 3	7.250	8.840	22%
Logistics 4	61.172	52.219	-15%
Public 1	46.900	39.030	-17%
Public 2	13.200	19.442	47%
Telco 1	2.456	3.588	46%
Telco 2	2.432	3.186	31%
Telco 3	1.056	1.518	44%
total	301.997	264.291	-12%
average value:			5%
standard deviation:			42%

Applying the UCP method does require a rough system specification, which uniquely indicates all use cases and its complexities. This might be provided by a specification based on use case diagrams or other methods like e.g. Event driven Process Chains (EPC) or prosaic descriptions of the requirements.

Table 1 summarizes the investigated projects. They cover different industrial sectors, they focus mainly on business software systems and they were delivered by different project teams. Column A shows the incurred project effort.

In order to avoid redundant table plots, two additional columns are already given: Column B shows the estimated effort according to the original UCP method defined by Karner. Estimations were performed by the project leader together with an UCP expert. The last column shows the quality of the method measured by the deviation in percentage of the project effort. This will be discussed later in chapter 4.

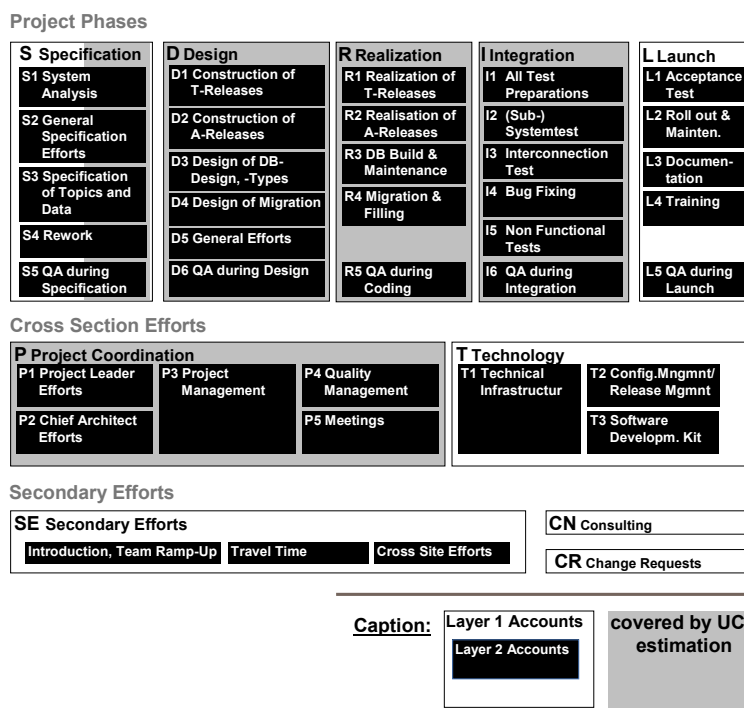


Figure 1. Form of accounts for assignment of project activities to track efforts

For all projects, the effort scopes consistently the phases from detailed specification to preparation for acceptance test (figure 1). In the Rational Unified Process (RUP) [11] these are the phases Elaboration and Construction. This is somehow in accordance with the original UCP method but has now been defined more precisely for the case study setting. This means in particular, that efforts for preparatory studies, initial operation / go live, training, and warranty are not considered. In addition, we exclude indirect project efforts like general job training, team building especially during team ramp up in big projects, travelling time, overhead for cross centre work or offshore delivery. No effort is estimated for providing general project equipment, in particular software development environment, configuration management, release management and any technical infrastructure.

During project run the project tasks of all team members were recorded daily with a granularity of 15 minutes and were stored in an accounting system. To provide a comparable basis for the measurement of project efforts, it is crucial that the project team members have a common understanding of how to assign activities to project tasks. For example: Does the time for debugging during module test assign to the task “quality assurance” or “module development” and is it covered by the project estimation? Does your daily Email reading belong to the project effort or not? What about team meetings? Therefore, we have first established a company-wide standard form of accounts (figure 1). It is shown in shadow colour, which efforts are covered by the UCP estimation. The efforts covered by the method were added after project close to retrieve the overall incurred efforts. All other project efforts are left out.

This results in a clear and uniform classification about efforts to be charged on project accounts or to be outside the project. Thus, on this basis we are now able to compare project efforts of different project teams in different industrial sectors.

4 Suitability of the original UCP method

Within the case study, the original UCP method revealed only a moderate mapping between estimation and incurred efforts. The average value (see table 1) only is 5 % higher than the incurred efforts and thus, fits quite well, but the standard deviation of about 42 % is not acceptable for industrial usage.

To find out possible reasons we analysed the cost drivers in all projects by discussing independently with the experts (i.e. project leader and chief architect). We gathered information on how the projects ran and searched for similarities and differences. Within several rounds with the panel of experts we found hypotheses for deviation between the estimation and the incurred project efforts. Detailing the full list of these hypotheses would exceed the limits of this paper, we list the two most important once:

1. We found that the way in which a use case is written is not standardized with respect to its complexity. Use cases have to be characterized in a consistent manner with similar granularity by different analysts. In almost the same manner the description of TCF and EF is not standardized. This leads to different results by different estimators working on the same project. Successful effort estimation using the UCP method will therefore require a standardization of the method. A solution for that is presented in 5.1
2. The most promising hypothesis was that the followed software development process model might strongly influence the project effort. We found three clusters of projects applying different process models and different levels of formal requirements regarding process control and documentation. Regarding the process model we differentiate these three types:
 - A.) projects with „standard development process“, where a large team (7 people or more) cooperates based on a formalized process model with clearly defined types of outputs (chapter 4.1),
 - B.) projects with a less formal and “lean development process”, usually staffed with a small team (chapter 4.2),
 - C.) projects with “heavy-weight development process” (chapter 4.3).

The original UCP method does not take these types of process models into account. In the following each type is characterized. In section 5.3 we explain how the UCP method has to be enhanced to be more precise by taking the applied process model into account.

4.1 Standard development process

Examples for standard development process models are RUP [11] or the V-model [7]. In such projects some or all of the following characteristics apply:

All cross sectional roles (project leader, chief architect, quality manager, configuration manager etc.) are staffed with dedicated persons, no one has two roles at the same time. There are cross sectional efforts that overlap between different projects (program management, chief design etc.). All documents relevant in the process are delivered. Everything is fixed in formal protocols and reports. Formal ways of making decisions have to be applied. Formal reviews and acceptance statements are required.

4.2 Lean development process

In projects with lean development process it is neither necessary to set up a complete project organisation with dedicated persons in all cross sectional roles nor to set up formal communication structures.

To establish such a lean process the following prerequisites are required:

- The project has low criticality
- The project environment, especially the requirements, are quite stable
- The relationship between customer and supplier is well established and both parties trust in each other
- The project team in its different roles and its working process is well established („jelled team“ [6]).

In this case, a project can be organized efficiently. Reduced cross sectional efforts make the project lean. The team organizes itself, less project management effort is required. Secondary milestones can be omitted. Organizational and communication overheads are economized. Documents that describe the development process do not have to be produced, e.g. project handbooks, quality plans in prose, project diaries, test plans, documents about quality goals etc. Still required are test case descriptions and quality plans in graphical and table form. Specifications must not necessarily contain descriptions about the actual situation. Formal reviews of intermediate results are not required in any case. The requirements of reporting are relatively low, e.g. no or only informal status reports are created, no or only a few minutes are taken.

4.3 Heavy-weight development process

In the case study this type is represented only by one project: Finance 1 in table 1. It was delivered on times and materials basis where as all other projects were delivered by fixed price models. Project Finance 1 encountered extensively extra process steps to meet the customer needs and in addition showed a very high rate of change requests. We

will not go into details in the following since for a reliable definition more projects of this type would be necessary. This is matter of further investigations.

As already said, the original UCP method does not take these three types of process models into account. In the following section 5.3 we explain how the UCP method has to be enhanced to be more precise by taking the applied process model into account.

5 Enhanced UCP method

In a first step, we evaluated the UCP method as a whole to understand the influencing effects and to get a more clear structure for further enhancements. From industrial point of view it would help to separate influences on project effort caused by the requirement definition and those caused by the way of project delivery. This leads to a new interpretation of the UCP terms characterizing different types of efforts:

A-factor: Use cases define the functional requirements related to the scope of the project. In business information systems these requirements are implemented in form of application software (A-software) [14]. The effort for implementation is proportional to the unweighted use case points (UUCP), we call it A-factor.

T-factor: It corresponds to the „technical complexity factor“ (TCF) in the original UCP method. It stands for the technological (T-) aspects of the overall efforts.

M-factor: The management (M-) factor defines the complexity caused by project organisation and has been derived from the original „Environmental Factor“ (EF) of the original UCP method. The EF has been enhanced to a new advanced M-factor.

The A-factor represents the application specific functional requirements of the project. The T-factor represents the non functional requirements. Both address system requirements which are mainly under determination of the buyer of a software project. The M-factor represents the influence of the project process on the effort. The constant productivity factor PF calibrates the delivery efficiency rate. Both are mainly determined by the supplier of a software project. The product of all these terms gives the total project effort PE of the enhanced UCP method. These interrelations are summarized in figure 2.

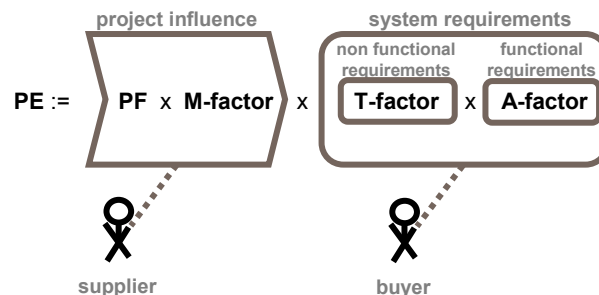


Figure 2. Project effort estimation in the enhanced UCP method

With this nomenclature, we systematically differentiate between different sources of efforts in software development projects dealing with business applications. Further enhancement of the UCP method, especially dealing with the complexity factors should follow this structure.

As stated in chapter 4 an essential enhancement to the UCP method has been invented by standardizing the complexity values represented by the A-, T- and M-factor. The standardization is based on examples and best practices as described in sections 5.1 to 5.3

5.1 A-factor

Now, we aim for getting the A-factor more precisely defined, especially focussing on definite granularity of a use case specification. This will be achieved by standardizing the weighting {simple, medium, complex} of use cases.

To ensure standardized levels of complexity, each use case is rated by the number of its main scenarios, steps and dialogues within the course of a use case.

A **scenario** is defined by its faultless courses achieving the main business goal or alternative goals of the use case. Fault scenarios are only counted if they contain business logic.

A **step** is defined by a self-contained business part of the use case being clearly separated by the adjacent steps, e.g. by change of the actor, by intermediate results or by splitting into scenarios.

A **dialogue** is any kind of interface of human interaction and includes e.g. print output.

A detailed description is provided in the extensive and detailed user guide [9]. We define the following levels of complexity:

- Simple: at most 3 main scenarios, steps and dialogues => 5 points
- Medium: at most 7 main scenarios, steps and dialogues => 10 points
- Complex: 8 or more main scenarios, steps and dialogues => 15 points

The point values {5, 10, 15} are chosen according to the original UCP method [10]. The metric for steps and dialogues is based on large (unpublished) statistical data of the Capgemini group in the field of use case.

We count actors according to the original UCP method. According to our experiences, the influence of the actor counts A_i to the use case points can be neglected. In future times, it might be possible to omit the actors from the enhanced use case method.

If n is the number of use cases, m the number of actors and U_i the point values of the use cases, we define the A-Factor to

$$\text{A-Factor} := \sum_{i=1..n} U_i + \sum_{j=1..m} A_j$$

5.2 T-factor

We took over the influencing variables of the T-factor from the original UCP method. But we standardized the values of complexity (0 to 5 points) by presenting examples and analogies. In table 2 column 3, the standardized values are given for each influencing

variable. Column 4 defines the corresponding weight g_i according to the original method [10]. The T-Factor is calculated by multiplying the influencing variables T_i with the weights g_i and by summarizing all weighted values:

$$T\text{-Factor} := 0.58 + \sum_{i=1..13} (T_i * g_i * 0.01)$$

Table 2. T-factor

T-factor ("Technical Factor")			
T_i	Description	Guideline for scoring (0 to 5 points)	weight g_i
T1	Distributed System	Degree of distribution of the system architecture? 0: Monolithic System 3: 3-tier with boundary systemes 5: highly distributed architecture	2,0
T2	Performance and load requirements	Level of requirements concerning performance and load balancing? Is response time one of the important criteria? 0: no requirements 3: average performance and load requirements 5: high performance and load requirements, e.g. load balancing, number crunching	1,0
T3	Efficiency of the user interface	How's the end user's efficiency? 0: no requirements, e.g. batch application 3: average user interface requirements, e.g. Web GUI, simple swing GUI 5: highly integrated, most efficient user interface, e.g. for power user, macros	1,0
T4	Complexity of business rules and calculations	How complex are the system's business rules? 0: only simple rules, no calculations 3: usual complexity 5: highly sophisticated rules / calculations	1,0
T5	Reusability	Do we intend to keep the reusability of the source code high? 0: no reusability requirements, e.g. software designed for one single use 3: normal requirements 5: high requirements, e.g. framework	1,0
T6	Easy to install	Is client looking for installation ease? 0: no requirements 3: normal installation requirements, i.e. dedicated client department will install a few instances 5: high installation requirements, i.e. self-contained installation by a lot of users	0,5
T7	Easy to use	Is user friendliness a top priority? 0: no requirements (no user) 3: normal requirements (GUI, help system) 5: high requirements (e.g. GUI-versions for different user types, internationaliation, wizards, fault tolerance)	0,5
T8	Portability	Is the customer also looking for cross platform implementation? 0: no requirements, e.g. software only runs once 3: normal requirements (one platform, normal level of abstraction) 5: high requirements, e.g. cross-platform, Windows + Unix	2,0
T9	Easy to change	Is the customer looking for high customization in the future? 0: no requirements, e.g. software runs only once 3: normal requirements, e.g. customizable 5: high requirements, e.g. customizable by templates / skins, plugin interfaces	1,0
T10	System availability	Is the customer looking at large number of users working with locking support? What level of system availability is required? 0: no requirements (no parallel transaction) 3: normal availability requirements 5: high availability requirements, i.e. 7x24 h operations, 99.x% availability, high number of parallel transactions	1,0
T11	Special security features	Is the customer looking at having heavy security requirements? 0: no requirements 3: standard security requirements (authentication, authorization) 5: high security requirements (certificates, encrypted communication, cryptography)	1,0
T12	Direct access for third parties	Does the project depend on using third party controls? 0: no requirements 3: system usage by end customer (B2C) 5: e.g. external interfaces, B2B-systems, trading platforms	1,0
T13	Special user training facilities	Will the software from user perspective be so complex that separate training has to be provided? 0: no user training necessary, self-explanatory 3: standard user training or self-study by examples and tutorial 5: training is precondition for using	1,0

5.3 M-factor

Within a first study [8], we have reduced the number of influencing variables of the M-factor from 8 to 6 in comparison to the original UCP method.

The factors E1: Familiar with the Rational Unified Process software process and E3: Paradigm experience (OO) do not have a significant influence on the estimation results since at sd&m all employees do have the same high education level.

Furthermore, each single influencing variable of the M-factor has been standardized with regard to the level of complexity by giving examples and analogies. The values range from 0 to 5 points. This best practice approach is detailed in table 3. In the fourth column you find the weights g_i corresponding to the influencing variables M_i .

Table 3. M-factor

M-factor ("Management Factor")			
M_i	Description	Guideline for scoring (0 to 5 points)	weight g_i
M1	Application experience	How much is the application experience in the team? 0: overall new 3: application and context is known in parts by the team or by a part of the team 5: application and context is well known by the team (characteristic of high release numbers or minor releases)	0,5
M2	Chief architect	Experience of the chief architect? 0: inexperienced with the task; little knowledge about the specific application domain 3: normal experience, application domain is well known 5: very experienced with the task and application domain is very well known	0,5
M3	Motivation	Is the team motivated for working on the project and for achieving the objectives? 0: unmotivated 3: motivated 5: excellent motivation	1,0
M4	Stable requirements	Is the client clear of what they want? Do we have stable requirements? 0: high rate of change requests also regarding fundamental requirements 3: normale change request rate, standard CR management 5: very stable requirements, no CR management necessary	2,0
M5	Part-time workers	Is there part-time staff in projects? 0: all team members are working on the project at least with 90 % 3: part of the team is working less than 70 % 5: significant part of the team is underavailable	-1,0
M6	Programming language / SDK	How expressive is the programming language, what about the complexity of the Software Development Kit (SDK)? 1: simple (e.g. Perl, PHP) 3: standard (e.g. Java, Cobol) 5: complex, exotic, not well understood; e.g. Assembler, APL, ...	-1,0

In table 4 the projects have been classified according to their process model. The column "UCP" names the weighted use case points according to section 5.1, i.e. UUCP times T-factor (table 2) times M-factor (table 3). The column "B: estimated effort" is calculated by UCP times PF, the productivity factor PF has been taken to 28.7 h/UCP based on a calibration of the productivity factor for sd&m purposes.

For the following, we introduce a new influencing variable **M7** representing the process model with scoring:

- lean development process => 0 points
- standard development process => 3 points
- heavy weight development process => 5 points

If we plot these “process model points” versus “deviation (B-A)/A” (see table 4), a clear correlation can be observed, indicated by the resulting line of linear regression analysis (least square method, [17]) in figure 3. This correlation plot gives evidence that the UCP method can be enhanced by taking into account the process model.

Table 4. Project data and process model

sector	A: project effort [h]	UCP	process model	process model points	B: estimated effort [Bh]	deviation (B-A)/A
Apparel Industry	728	42.0	lean	0	1.205	66%
Automotive 1	15.500	406.5	standard	3	11.667	-25%
Automotive 1	136.320	3.972.9	standard	3	114.023	-16%
Finance 1	2.992	51.4	heavy wt.	5	1.475	-51%
Finance 2	3.680	115.0	standard	3	3.301	-10%
Insurance 1	4.800	73.7	standard	3	2.115	-56%
Logistics 1	944	49.0	lean	0	1.406	49%
Logistics 2	2.567	61.0	standard	3	1.751	-32%
Logistics 3	7.250	308.0	standard	3	8.840	22%
Logistics 4	61.172	1.819.5	standard	3	52.219	-15%
Public 1	46.900	1.359.9	standard	3	39.030	-17%
Public 2	13.200	677.4	lean	0	19.442	47%
Telco 1	2.456	125.0	lean	0	3.588	46%
Telco 2	2.432	111.0	lean	0	3.186	31%
Telco 3	1.056	52.9	lean	0	1.518	44%
total	301.997				264.764	-12%
average value:						6%
standard deviation:						40%

One might expect that larger projects require more process effort in general. A detailed analysis of the size of the project versus the “deviation (B-A)/A” does not show any significant correlation, neither measuring the size in lines of code nor in use case points.

In order to integrate the new influencing variable M7 – process model into the M-factor, an appropriate weight g_7 has to be determined. It is the value for which the “deviation (B-A)/A” reaches its minimum. By numerical iteration we get $g_7 = -4.5$.

We performed further correlation analysis for the other influencing variables in the M-factor. For the 15 projects we found the following statistical trends:

M1 – Application experience: Very low application experience leads to a highly positive deviation, i.e. overestimation of the effort. Normal experience (= 3 points) leads to underestimation. This stands in contradiction to Karner’s formula. Therefore, we change the weight of this M1 factor from 0.5 to -0.5 points which leads to a better fit. An explanation could be that estimators with only little knowledge tend to estimate by fear higher values in order to cover their uncertainty.

M2 – chief architect: Superior chief architect experience in the project team (4 points) leads to underestimation of the efforts whereas a normal experience results in overesti-

mation. A good fit can be achieved if the weight is reduced from 0.5 to -1.5. This means that an experienced chief architect generates higher efforts than a less experienced one which does not make sense to us. On the other hand, in the data set only projects are represented, where M2 has 3 or 4 points. For a detailed and sustainable statistical analysis, this spread of points is too low, yet. Therefore, we keep the weight of M2 at zero to avoid the influences of this variable.

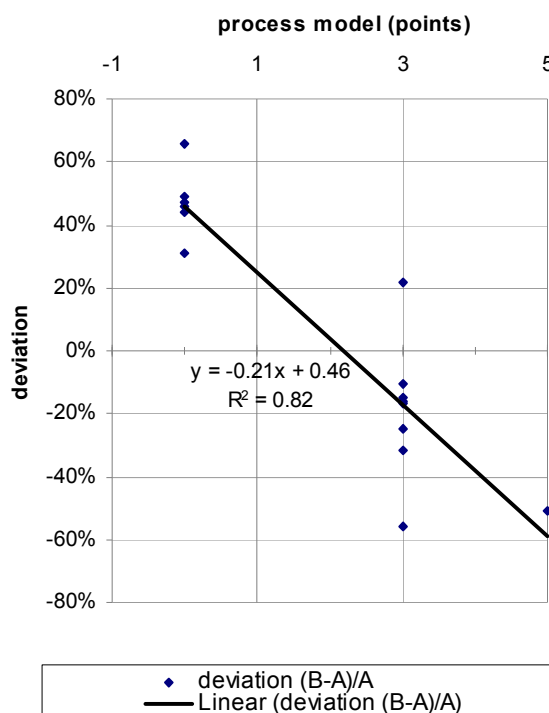


Figure 3. Correlation plot (regression analysis)

M3 – Motivation: we found a slight correlation between M3 and the deviation. This indicates that the weight for M3 could be increased. A higher motivation would lead to higher efficiency and thus to lower project efforts. The best fit has been achieved by increasing the weight from 0.5 to 2.5 points.

M4, M5 – we keep them unchanged.

M6 – Programming language: Since M6 is rated by 3 (= normal) in 13 of 15 of the analyzed projects, we do not find a significant dependency of the projects to M6. We are convinced that M6 may be neglected, because today, due to powerful software development tools, the project efforts do not depend to that extensity on the programming languages as at the time when the method has been invented.

Team size & time – we would expect this being a strong influencing factor [4] but this is not considered by the original UCP method. In this case study we did not detect a correlation between deviation and team size or project duration, therefore we left this unchanged out.

After all, by numerical iteration we can set up a new formula to calculate the M-factor:

$$\text{M-Factor} := 0.8875 - \sum_{i=1..7} (M_i * g_i * 0.025)$$

M_i and g_i are defined in table 5. The M-factor ranges from 0.33 to 1.64 and has been standardized to 1.0 (i.e. all factors rated with 3 points). This standardization causes a rescaling of the productivity factor PF from 28.7 to 35.0 h/UCP.

Table 5. New enhanced M-factor

M_i	Description	weight g_i
M1	application experience	-0,5
M2	lead analyst capability	0,0
M3	motivation	2,5
M4	stable requirements	2,0
M5	part-time workers	-1,0
M6	Programming language	0,0
M7	process model	-4,5

5.4 Suitability of the enhanced UCP method

The enhanced UCP method has been applied to the 15 projects of the case study. The results are listed in table 6.

The enhanced method shows a good fit between the estimated effort and the incurred project effort. There is a low deviation, expressed by an average value of -0.5 % and a standard deviation of 20 %. In comparison, applying Karner's method we achieved an average value of 5 % and a standard deviation of 42 %.

Table 6. Results of the enhanced UCP method

sector	A: project effort [h]	Actors	UUCP	M- factor	T- factor	UCP	B: estimated effort [h]	deviation (B-A)/A
Apparel Industry	728	5	42	0,66	0,87	27	943	29%
Automotive 1	15.500	18	400	0,92	1,06	405	14.181	-9%
Automotive 1	136.320	95	2.988	1,18	1,07	3.858	135.030	-1%
Finance 1	2.992	11	60	1,29	0,89	81	2.832	-5%
Finance 2	3.680	9	115	0,98	1,03	125	4.358	18%
Insurance 1	4.800	11	53	1,10	1,05	73	2.567	-47%
Logistics 1	944	6	70	0,50	0,68	26	904	-4%
Logistics 2	2.567	8	45	0,91	1,14	55	1.921	-25%
Logistics 3	7.250	8	260	0,91	1,14	278	9.715	34%
Logistics 4	61.172	86	1.605	1,00	0,95	1.598	55.930	-9%
Public 1	46.900	106	1.158	0,99	1,06	1.323	46.308	-1%
Public 2	13.200	22	556	0,69	1,06	422	14.753	12%
Telco 1	2.456	10	160	0,44	0,94	70	2.434	-1%
Telco 2	2.432	14	115	0,48	1,04	63	2.220	-9%
Telco 3	1.056	8	60	0,54	0,90	33	1.151	9%
total:	301.997						295.246	-2%
average value:								-0,5%
standard deviation:								20%

If all counted values with the UCP method were statistically independent than we could take the same data for calibrating the parameters and for evaluating the approach. The statistical independency has not yet been proven. Therefore, we validated the enhanced UCP method with a new and different set of projects and achieved a comparable good result.

6 Discussion and future work

A quantitative comparison with other estimation methods is out of scope of this paper.

In the literature, the UCP method has been questioned for its ambiguity such as in [15]. The level of abstraction of use cases might significantly influence the results of the estimation. This topic has been addressed in our enhanced UCP method by setting up a **user guide** which defines the level of abstraction and the graduate of complexity of a given use case. The reproducibility of estimation results of different estimators is increased and, thus, the organisational-wide application of the UCP method in software development projects is improved.

The **detailed classification** of the A-, T- und M-factors provides a uniform and normalized approach for IT organizations to perform a use case based estimation. This way, estimations of efforts can be more easily reproduced by different persons. We proved this e.g. by asking five independent persons to calculate the number of use case points based on a requirements specification in a request for proposal in the public sector. As a result, all the five persons estimated the amount of use case points with a deviation only of 5 %. Further work has do be done to prove this reproducibility of the UCP method.

In literature, we only found a few statements, to which project types the UCP method is applicable and where it does not suit. Our enhanced UCP method has been successfully

applied to custom build business applications. The method does not suit if the scale of a system adaptation can hardly be scoped by use cases. In the event of a pure technical system migration for example, the business logic (A-factor) does change too little. Now, it has to be questioned to which other types of projects the method is applicable.

Another important contribution of this paper is a **new UCP formula** (section 5). Altogether, the proposed enhanced and revised UCP method has cut in half the standard deviation and, thus, increased the overall estimation accuracy significantly.

Future investigations are necessary for getting a better understanding of the cohesion between project organization, project process and project effort. As a first step, we introduced the **process models** “lean”, “standard” and “heavy-weight”. In particular the heavy-weight process model needs further detailed investigations. The data basis is too small right now to define the criteria for the different process models precisely.

We expect that in future work we could be able to add a few new influencing variables to the M-factor to increase the estimation accuracy. In particular, we suggest taking into account the impact of project management topics like maturity level of the organization, number of hierarchy levels, complexity of team communication and team skill with regards to the different project stages (first release versus second release). In addition, the concept of a linear formula (figure 3) to calculate the total effort is only a first approximation. In accordance with the COCOMOII2000 [4] concepts we expect exponential factors, too.

More over, it might be worthwhile to improve the T-factor. We still expect a linear contribution to the A-effort. Nevertheless, a further improvement of the M-factor is more urgent.

References

- [1] A. Albrecht, J. Gaffney: “Software function, source lines of code, and development effort prediction: a software science validation“, *IEEE Transactions on Software Engineering* 9, 1983, pp. 639-648, 1983.
- [2] B. Anda et al: „Estimating Software development Effort based on Use Cases – Experiences from Industry“, 4th International Conference on the Unified Modeling Language (UML2001) Toronto, Canada, October 1-5, 2001, pp. 487-502, LNCS 2185, Springer-Verlag, 2001.
- [3] S. Azzouz, A. Abran: “A proposed measurement role in the Rational Unified Process (RUP) and its implementation with ISO 19761: COSMIC-FFP”, in: *Software Measurement European Forum - SMEF 2004*, Rome, 2004.
- [4] B. Boehm et al: “*Software Coast Estimation with COCOMO IP*”, Prentice Hall, 2000
- [5] A. Cockburn: *Writing Effective Use Cases*, Addison-Wesley, 2001.
- [6] T. DeMarco, T. Lister: *Peopleware*, Dorset House, New York, 1977.
- [7] W. Dröschel, M. Wiemers: *Das V-Modell 97*, Oldenbourg, 1999.
- [8] S. Frohnhoff, V. Jung, E. Engel: “Use Case Points in der industriellen Praxis“ in : A. Abran et al. Eds.: *Applied Software Measurement - Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik Kongress*, Shaker Verlag, 2006, pp. 511-526.
- [9] S. Frohnhoff, K. Kehler, R. Dumke.: „Modellbezogene Use-Case-Identifikation für die UCP-basierte Aufwandsschätzung, Preprint Nr. 9, Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, 2007

-
- [10] K. Karner: „Metrics for Objectory“. Diploma thesis, University of Linköping, Sweden, No. LiTHIDA-Ex-9344:21, December 1993.
- [11] P. Kruchten: *The Rational Unified Process: An Introduction*, 3rd ed., Addison-Wesley, 2003.
- [12] M. Lother, R. Dumke: “Points Metrics - Comparison and Analysis” in: Dumke et al. (Eds.): *Current Trends in Software Measurement - Proceedings of the 11th IWSM, Montréal*, Shaker Verlag, Aachen, 2001, pp. 228-267.
- [13] G. Schneider: *Applying Use Cases: A practical guide*, 2nd ed., Addison-Wesley Object Technology Series, 2005.
- [14] J. Siedersleben: *Moderne Software-Architektur*, dpunkt.verlag, 2004.
- [15] J. Smith: „The Estimation of Effort Based on Use Cases“, Rational Software, Cupertino, CA.TP-171, October 1999. <http://whitepapers.zdnet.co.uk/0,39025945,60071904p-39000629q,00.htm>
- [16] OMG, “UML™ Resource Page“, <http://www.uml.com>, updated 09th January 2007.
- [17] D. Urban, J. Mayerl: *Regressionsanalyse: Theorie, Technik und Anwendung*, 2. überarb. Auflage, VS Verlag, Wiesbaden, 2006.