Methodology for Estimating Working Time Effort of the Software Project

Jakub Štolfa, Svatopluk Štolfa, Ondřej Koběrský, Martin Kopka, Jan Kožuszník, and Václav Snášel

Department of Computer Science
VŠB – Technical University of Ostrava, Faculty of Electrical Engineering and Computer Science
17.listopadu 15, Ostrava-Poruba, Czech Republic
{jakub.stolfa, svatopluk.stolfa, ondrej.kobersky, jan.kozusznik, vaclav.snasel}@vsb.cz, martin.kopka@c4u.cz

Abstract. The precise estimation of the time effort of the project is one of the key limits of its success. One of the ways how to achieve a correct valuation of the project is developing of a detailed analysis, which output is a structured solution that uses use cases. This paper focuses on developing a methodology for estimating working time effort of the project for one particular company. An important part of the methodology is to build up and maintain comparative database of valued use cases and time progress of realized projects. The aim of the methodology is to deliver data for evaluating a new project.

1 Introduction

The proper estimation of the project is a goal which wants to achieve almost every project manager. It is not easy quest and it is not essential. It is hard task which takes a lot of effort to do it right. And the question is how to do it right. There are several ways how to fulfill this task. Which one is the best is depending on the concrete company, concrete types of the projects etc.

However, one thing is clear, if we know supposed project progress (supposed project progress of its activities), we can find out in which phase the project is. Thereby we can figure out if the project plan is in time, late or ahead. So, we can determine the effort and plan resources of the project. For example, since some point of time we know that project will not need analyst activities anymore, so we can move analysts (they do analyst activities) out of the project, to the another project.

1.1 State-of-the-art of estimation project approach

Many formal methods were published in the area of effort estimation for software development projects. Heemstra wrote down the basic ideas why, when and how to estimate projects in paper "Software cost estimation .In Information and Software

Technology" [12] in early 90s. This paper speaks about importance of estimation of the project. One of the mentionable information is that lot of companies do not make an estimation. And even if they make estimation, it is mainly not corresponding with a reality. Since our method is based on the use cases, we will mention only some methods utilizing the use case approach here (referenced also as parametric models).

The COCOMO methodology [1] computes the effort of the software projects as a function of program size and a set of cost drivers on separate project phases. It is the Constructive Cost Model, which has been originally developed by Dr. Barry Boehm and Publisher in 1981 [13]. He found out a formula computing the classification of separate cost drivers.

Similar access as COCOMO basic is used in [2] where author estimates that size of a project is a function of the size of definition that was written into the use cases definition. Function points are popular method to estimate size of proposed application. The ISO/EIR organization [6] created functional size metric standard which supports that method.

Methodology introduced by [3] computes the project estimation using complexity of use cases and its transactions applying the set of adjustment factors. Building the database from really measured projects with several technologies is important approach.

J. Smith in 1999 speaks about use cases and their complexity [4]. It thinks about how big should a use case be and about the complexity of the big or small use cases and how much effort particular use case takes. It brings us an idea that we used for categorizing standardized use cases in our assessed company.

Almost all methods, which use estimation based on use case points, are based on method of Gustav Karner. He first described use case points. [2] Use case point is described like function of the following:

- the number and complexity of the use cases in the system,
- the number and complexity of the actors on the system,
- various non-functional requirements (such as portability, performance, maintainability) that are not written as use cases,
- the environment in which the project will be developed (such as the language, the team's motivation, and so on).

Like Gustav Karner says, the basic formula for converting these parameters into single measure, which is mentioned use case point, there is that it is necessary to weight the complexity of the use cases and actors and then to adjust their combined weight to reflect the influence of the nonfunctional and environmental factors.

The case studies, which are based even on use case points, are described in the paper of Bente Anda [11].

We can say that our methodology is even based on the use case point, but it looks more precisely on the use case realization, that means text of the use case. It evaluates rows, words and paragraphs of the standardized use case realization. How it works, there is described later in our paper.

Our paper is organized as follows: Section 2 introduces the problem of the project estimation in the particular company; Section 3 describes the concept of our methodology: filling of the database by the data from finished projects, categorization of the use cases and example that describes new project effort estimation. Concluding Section 4 provides a summary and discusses the planned future research.

2 Definition of the problem

Our goal was to set up an estimation working time effort of projects in the particular company. Even that the company has 130 workers and lot of finished projects, people in that company were not used to estimate a new project by some methodology. The common issue was to estimate a new project working time effort by theirs own decisions. That decisions are based on experiences obtained by solving past projects. The problem of that solution of the estimation of the project's working time effort is that it is so much human dependable. For example it means that an estimation can be wrong because the particular worker had not enough experience to do that, or he simply don't know how to make the particular estimation. The solution to this problem is the supporting tool for estimating time effort of new projects based on our methodology. This can help workers to estimate a new project by showing them average project progress of similar projects. Thanks to that methodology, worker simply knows the progress of projects with similar working time effort and can easily estimate a new one.

It is important to mention that out methodology is based on particular company and their software developing standards. The methodology was supposed to use only in that particular company at the beginning. It means we do not declare that this is general approach which can essentially work in other companies. On the other hand the aim of this paper is to provide also the guidance for other companies, which develop software in same way like our one. They can apply our methodology to their processes as well as we did it in that particular company.

2.1 Initial state

Initial state of the estimation in the company was described before. Thorough it is important to mention how the company makes analysis and how is a progress of the project captured.

The analysis in the company is made by use cases. These use cases are standardized. That means, if the use case deals with same repeatable issue, then it is almost similar to other use cases that are dealing with the same issue. It gives us the possibility to use these standardized use cases for the projects comparisons.

The capturing of the project progress is made by CRM system. In the CRM system you can see how much time was spent on each activity. More information about captured activities is written in next sections.

3 Proposal of the methodology

The following chapter is focusing on the logical principles of the methodology. Methodology consists of two main processes. Rectangle is an activity; oval is input or output data to the activity.

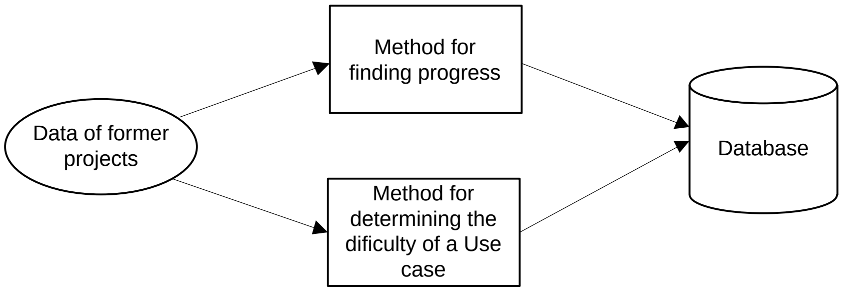

Method for finding progress
Data of former projects
Database
Method for determining the dificulty of a Use case

Fig. 1. Filling of the database.

The first process is named “Filling of the database” (Fig. 1). It is a process where data about a recent project are filled into the database. It starts with data of recent projects in particular company. These data are separately executed in two main methods. First one is the “Method for finding progress” and second one is the “Method for determining the difficulty of a Use case”.

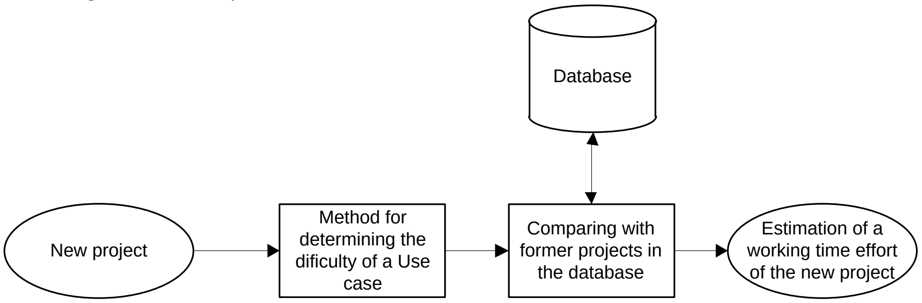

Database
New project
Method for determining the dificulty of a Use case
Comparing with former projects in the database
Estimation of a working time effort of the new project

Fig. 2. Estimation of the new project

In case, that database is filed by the first process, the second process can be run. The Name of that process is “Estimation of the new project”. The input of that process is a new project. The exact input is the finished analysis of that new project. It means that we have all required use cases. These use cases are elaborated in the activity “Method for determining the difficulty of a Use case”. It is the same method like in the process named “Filling of the database”. After that, process continues with the method named “Comparing with former projects in the database. The method compares former project in the filled database with data of the new project, finds similar project and estimate working time effort based on the similar projects progress. When the project is finished, the database could be extended by the new finished project. Next sections of this paper introduce particular steps of both processes.

3.1 Method of finding Progress

The supposed project progress, it means how much effort and time particular tracked activities should consume, is discovered by the comparison to the data of finished projects in the company. According to these data, we can find out if the company is following project methodology. If it is true, than all projects have the same or almost same progress of tracked activities.

Our method tracks progress of five main activities of the project.

- Consultation (in the system(CRM) like PORA)
- Analysis (ANAL)
- Programming (PROG)
- Testing (TEST)
- Implementation (IMPL)

Steps of the method finding progress.

1) **Input project data**. We need to know number of worked hours in each day for particular activities, when that activity was practiced (see the table below).
2) **Setting number of segments** to which we will split timeline of the project. Methodology set default number of segments to 10, but we can set even another number (5, 20, 100, etc.). The reason why to split timeline to the same segments is that, we need to normalize timelines of projects. So that we can compare projects whit each other. For example, one project last 10 months a second one last 1 month. If it is slip up only to the weeks, then first project is split to the 40 segments and second one to the 4 segments. Comparing these two projects is impossible in this way. If we split up these two projects to the same number of segments, we can compare them. First project has a 10 segments and one segments contains data of 28 days (project last 40 weeks = 280 days, we split up to 10 segments = 28 to each segment). Second project contains in one segment 2,8 days.
   This example shows that the last segment is not same as the others every time. It depends on the technique in the methodology, if we round up (default) or down. If we round up, one segment contains 3 days (2,8 days > 3 days). A it means that last segment contains only 1 day (28 - 28/3 = 1). It was proved by the experiments that this is not significantly important for the comparison of the projects. It is only good to have in on your mind for later refinement of the method.
3) **Choosing the technique of the evaluation.** Our methodology has two types of evaluation of the project duration, which is later split up to the, before mentioned, segments.

a. **Counting all days.** It means that we count all days – from the first day to the last day, when some of tracked activities were performed. Then the duration of the project (number of days) is counted. Number of days is divided by number of segments (default 10). So that we know how many days the segment contains. After that calculation we add to first date number of days in segment. That date is border line. Then we add number of day in the segment to that border line and establish second border line. We have second segment. That technique we have to do same way to the penult segment. To the last segment we put remaining days of the project. For example first date is 1.7.2010 and one segment has 14 days. First segment contains data of dates from 1.7.2010 to 14.7.2010. Second segment from 15.7.2010 to 28.7.2010, etc.

b. **Counting only days, where there was some work done.** It means that, we count on only days, when some of the tracked activities were performed. We do not count empty days. Days where there was not done any work on tracked activities. We count all effort hour and divide them by number of segments. Then, there is a difference from the first technique, we count only days, where there was some work. So segment of size 11 hours may contain for example 1.7.2011 5 hours of analysis, 13.7.2011 6 hour of programming.

   This technique has one week aspect. When we reach the end of the segment, actual summary of hours in that segment is 9 and next date contains for example 3 hours, so we close segment with actual summary 9 hours. Because we do not want to past out the limit of the segment (limit is 11, 9+3=12, 12>11). It has an effect that segments could contain various numbers of hours. This might be a problem if we count projects with low number of total hours. The affection is minimal to the projects with big number of total hours.

4) **Calculation.** Inputs are fulfilled segments (day o hour method). Then we need to count number of hours spent on particular activity in each segment.

5) **Saving a result to the database.** Saving the result of the methodology is made by vectors. We can group these vectors and so that we can find out similar projects. Vectors are easily transferable to the graph.

   a. **Structure of the vector** (5 segments):

   First is a name of the project. Then three numbers of particular difficultly of use cases. And then are worked hours of the activity in particular segment. First is consultation, then analysis, programming, testing and last implementation. Each activity has 5 numbers divided by semicolon. It shows how much hours was worked in each segment.

Name of the project	Difficulty of Use cases			Hours of consultation per each segment					Hours of analysis per each segment					Hours of programming per each segment					Hours of testing per each segment					Hours of implementation per each segment				
Project	H	M	E	13,75	0	18,25	255,75	237,25	0	0	0	122,5	116,5	0	0	0	138	2704	0	0	0	0	1071	4	0	11,75	8,5	171,25

Table 1. Structure of the vector. Table shows structure of the vector. It is one log table, but for that paper was divided to two.

3.2 Method for determining the difficulty of a Use case

Following chapter describes how we evaluate particular use case.

Basic complexity according to the number of rows.

Complexity of a project is given by the difficulty of UC realizations. The difficulty of UC is hard to determine, there is no easy benchmark for their comparison. Simplest way is determining the complexity based on number of rows in every UC. We have set 3 levels of difficulty based on number of rows:

- E (easy) – number of rows < 70
- M (medium) – number of rows ε <70;110>
- H (hard) – number of rows > 110

We can obtain number of hard, medium and easy use cases for one project with this type of evaluation.

Extended complexity.

In terms of our method objectivity we decided to extend complexity with number of paragraphs and number of words in each UC. The overall difficulty of UC is then derived from the individual complexities.

Difficulty according the number of words.

- E (easy) – number of words < 200
- M (medium) – number of words ε <200;500>
- H (hard) – number of words > 500

Difficulty according the number of paragraphs.

- E (easy) – number of paragraphs <= number of paragraphs + X -> easy
- M (medium) - number of paragraphs > number of paragraphs + X AND number of rows < number of paragraphs + Y -> medium
- H (hard) – number of paragraphs => number of paragraphs + Y -> hard

Where X=10, Y=25

The overall difficulty.

Overall difficulty of UC can be determined as follows. Difficulty of words has the highest weight, then difficulty of rows and difficulty of paragraphs. First of all we compare difficulty of words with difficulty of rows, If they are in the same level, then the overall difficulty is in this level. If not, we compare the difficulty of words with difficulty of paragraphs, if they are on same level, then the overall difficulty is in this level. If any of them differs then we compare the difficulty of rows with paragraphs in the same way. If the comparison does not help us, overall difficulty is difficulty of words, because we consider it the most important aspect. The difficulty types and levels were checked by the correlation matrix. The result was that the actual dependency setting varies between 60-100 %. Most of the difficulty types and levels more than 2/3) are more than 95% dependable.

3.3 Comparing to former projects in the database

Input of this activity is evaluated by use cases of a new project. This overall difficultness of project use cases was determined in the activity Method for determining the difficulty of a Use case.

After that we find recent project are in the database which have almost similar difficultness of a use cases. The similarity is determined by next proclamation. We proclaim that two projects are similar if their particular difficulties differ by +/- 2. This simple differ was set up by several experiments made on real data in the company.

3.4 Example

This section describes an example of using a described methodology. Inputs are analysis of four projects (Project1, Project2, Project3, Project4). It is only the example so that's why, we show only four representative projects. We cannot publish the names of the project, and that is the reason why we call them like that. Important is that all of these projects are development projects. That means they include all steps of the life cycle of the project (consultation, analysis, programming, testing, implementation).

We fulfill the database by data (vectors) of these projects. After that we take a new project, evaluate its use cases (by difficultness of UC) and find out estimated progress and difficulty.

1) **Determination of difficultness of UC of particular projects:**

Difficulty according to the number of rows

	T	S	L
Project1	6	8	57
Project2	1	1	12
Project3	4	0	17
Project4	0	2	4

Table 2. Difficulty according to the number of rows.

Difficulty according to the number of paragraphs

	T	S	L
Project1	12	17	42
Project2	1	11	2
Project3	3	5	13
Project4	0	4	2

Table 3. Difficulty according to the number of paragraphs.

Difficulty according to the number of words

	T	S	L
Project1	16	11	44
Project2	3	7	4
Project3	5	3	13
Project4	2	2	2

Table 4. Difficulty according to the number of words.

Total difficultness

	T	S	L
Project1	12	15	44
Project2	1	9	4
Project3	3	4	14
Project4	0	4	2

Table 5. Total difficultness.

At the table 5 we can see the total difficultness of particular projects. This view is most important for further processing. The section 3.2 describes how the particular difficultness was set up.

2) Compilation of vectors and graphical representation of these vectors for the projects progress

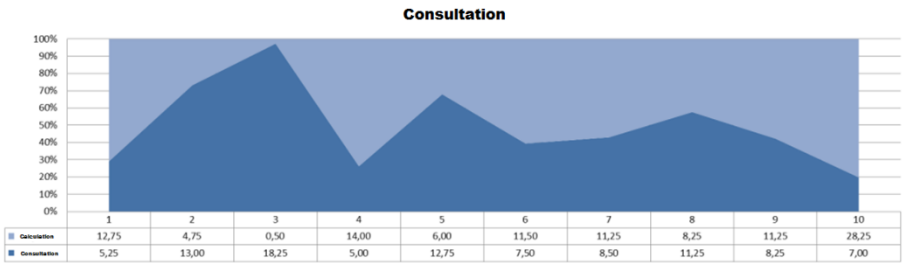
This section shows the vector for the project progress of the particular project. Structure of these vectors was described in the table 1 above.

- Project1;12;15;44;284,75;28;48;36,75;29;60,5;27,55;2,5;0;122,5;0,75;8,25;19;27,5;27,25;22,2;2;4,5;0;52,5;444,75;332,75;317,5;341;320,5;256,75;275,25;259;0;0;10,5;93,25;109,5;91,25;7;9,5;160;188,75 ;185,5;0;24,25;3;9,25;1,75;0;0,5;27,25;16;37;0
- Project2;1;9;4;23,75;20;12;30;15;40,25;20;10,5;1;130,50;20;10;18;25,5;20,25;10,25;1;3;2;2;20,25;130,75;256;432;203;150,75;25,25;259;30;0;14,5;23,25;19,75;19,75;10;17;163;129,75;174,25;2;12,25;6;10,25;4,75;0;3,5;20,25;13;20;35

- Project3;3;4;14;5,25;13;18,25;5;12,75;7,5;8,5;11,25;8,25;5;2;10,75;1,5;0;10;6;5,5;0,5;0;5,5;3,75;1,5;2;0;0;1;0;6;9;8,25;5,75;9,75;10,25;0;3,25;0,5;3;0;0;1,75;0;0;1,5;1;0;0;0;0;0;0;0;0;0;0;0,5
- Project4;0;4;2;12;8;6;14;12,25;31,25;5;2;1;10,50;3;7,75;20,50;9,75;15;13,50;5;7,75;3,25;5;10,75;30,50;26;42;20;10,25;2,75;29;18;2;5,5;2,25;9,75;4,75;1;8;16;29,25;4,25;1;6,75;6,50;2,25;3;1;0,0;0;25;13,50;18,25

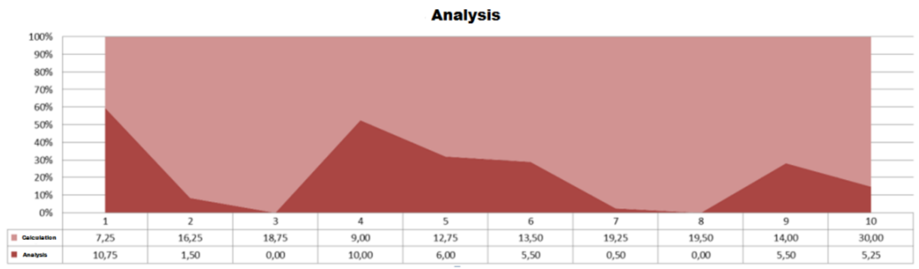
3) Estimation of a new project – Project5

- Input is the new project – Project5
- Determination of difficultness of UC of the Project5:
  - Hard; Medium; Easy = 4;6;13
- Finding similar projects in database:
  - 4+-2; 6+-2; 13+-2 – we find out project with difficultness of UC +-2 according a new project. From these specified projects we make average of their progress. And that is set as estimation of the new project.
  - In our example it is only Project3, so we do not need to do average.
  - Estimated progress of the Project5 is:



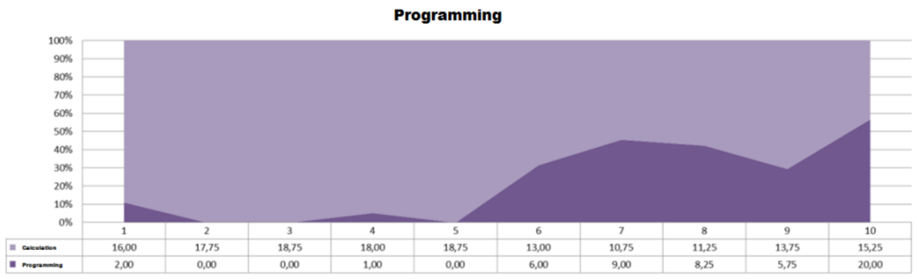
	1	2	3	4	5	6	7	8	9	10
Calculation	12,75	4,75	0,50	14,00	6,00	11,50	11,25	8,25	11,25	28,25
Consultation	5,25	13,00	18,25	5,00	12,75	7,50	8,50	11,25	8,25	7,00

Fig. 3. Progress of the consultation activity. It shows estimation of working hours of current activity in particular segments of the project.



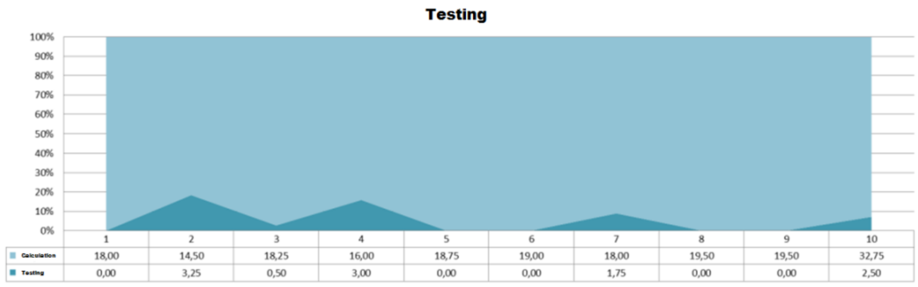
	1	2	3	4	5	6	7	8	9	10
Calculation	7,25	16,25	18,75	9,00	12,75	13,50	19,25	19,50	14,00	30,00
Analysis	10,75	1,50	0,00	10,00	6,00	5,50	0,50	0,00	5,50	5,25

Fig. 4. Progress of the analysis activity. It shows estimation of working hours of current activity in particular segments of the project.



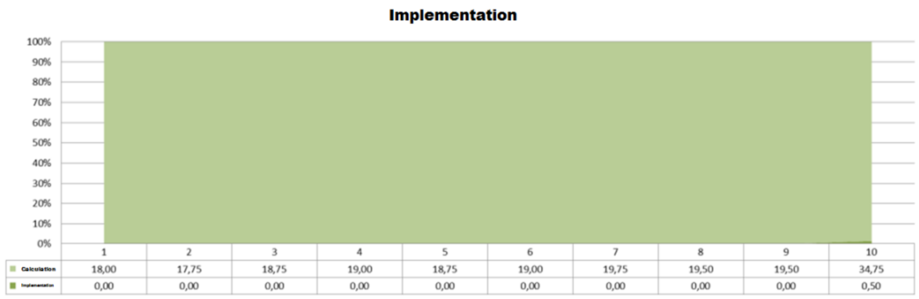
	1	2	3	4	5	6	7	8	9	10
■ Calculation	16,00	17,75	18,75	18,00	18,75	13,00	10,75	11,25	13,75	15,25
■ Programming	2,00	0,00	0,00	1,00	0,00	6,00	9,00	8,25	5,75	20,00

Fig. 5. Progress of the programming activity. It shows estimation of working hours of current activity in particular segments of the project.



	1	2	3	4	5	6	7	8	9	10
■ Calculation	18,00	14,50	18,25	16,00	18,75	19,00	18,00	19,50	19,50	32,75
■ Testing	0,00	3,25	0,50	3,00	0,00	0,00	1,75	0,00	0,00	2,50

Fig. 6. Progress of the testing activity. It shows estimation of working hours of current activity in particular segments of the project.



	1	2	3	4	5	6	7	8	9	10
■ Calculation	18,00	17,75	18,75	19,00	18,75	19,00	19,75	19,50	19,50	34,75
■ Implementation	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,50

Fig. 7. Progress of the implementation activity. It shows estimation of working hours of current activity in particular segments of the project.

These figures show progress of particular activities of the Project5. We can see estimation of working hours of particular activity in particular segments of the Project5. We can have a look for example on activities consultation and programming. It is interesting to note that estimation of working hours of consultation activity is almost stable for every segment. On the other hand estimation of working hours of programming activity is divided to the two parts. At the beginning of the project are hour spend on programming activity minimal, but at the end are major. It is essential because at the begging is not too much programming work.

4 Conclusion and Future Work

Our methodology was tested on the 20 past projects made by the company. Fifteen projects were taken to the process of filling the database. Five projects were proclaimed as new projects. The result of our methodology – supposed project progress (effort of each activity) was compared to the historical data of these projects. The difference between the predicted progress and the actual data for the tested projects was approximately 30%. The 30% inaccuracy seems not to be a good, but the previous ad-hoc human based estimation had inaccuracy 40%. We found out these data by comparing their original estimations on the beginning of the projects with result of these projects at their end. In 40 percent there was huge deflection of estimation and the real execution. Therefore our methodology brings approximately 10% improvement, which is a good result of the methodology. But we know that 30% is still huge number of inaccuracy, so that there is place to improve our methodology in future.

In the future, we plan to improve our methodology by neuron nets as tools which find out groups of similar projects. Otherwise, we plan to elaborate parameters describing the influence of the customer to every single project. Then we have to elaborate if is better to use more parameters to describe particular use cases for an execution of the methodology. At least but not last we have to have on our mind that we estimate project after first steps of analysis. Especially, after the use cases are finished. That's important to mention, because we have to include that work to the estimation of the project.

I any case the topic of estimation project's effort is huge place for research and improvements. The methodology that works for one company does not have to work for others. Our goal is to overcome that gap by trying to develop methods and that will be used in more companies than one and the results will be more accurate.

5 References

1. Boehm, B., Abts, Ch., Brown, W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D., Steece, B.: Software Cost Estimation with COCOMO II. Englewood Cliffs, NJ:Prentice-Hall, 2000. ISBN 0-13-026692-2
2. Cohn, M.: Estimating With Use Case Points, Methods and Tools, Fall 2005 (Volume 13, number 3), ISSN 1661-402X.
3. Ochodek, M., Nawrocki, J., and Kwarciak, K.: Simplifying effort estimation based on Use Case Points. Information and Software Technology, 53(3):200–213, 2011.
4. Smith, J.: The Estimation of Effort Based on Use Cases, Rational Software white paper, 1999
5. Ruhe, M., Jeffery, R., Wieczorek, I.: Using web objects for estimating software development effort for Web applications. In: Proceedings of the ninth international software metrics symposium, Sydney, Australia 3–5 September 2003, p 30
6. ISO/IEC 14143-1:1998 (1998) Functional size measurement.www.iso.org
7. Ribu, K.: Estimating Object-Oriented Software Projects with Use Cases. 2001. Master of Science Thesis, 2001, University of Oslo, Department of Informatics.
8. Ochodek, M., Nawrocki, J.: Enhancing use-case-based effort estimation with transaction types. Foun- dations of Computing and Decision Sciences, 35(2):91–106, 2010.

9. Kemerer, Ch.: An empirical validation of software cost estimation models, Communications of the ACM, Volume 30, Issue 5, New York 1987.
10. Anda, B.: Comparing Effort Estimates Based on Use Case Points with Expert Estimates, Empirical Assessment in Software Engineering (EASE), Staffordshire 2002.
11. Bente A, Hege D., Dag I. K. Sjoberg, and Magne Jorgensen. 2001. Estimating Software Development Effort Based on Use Cases-Experiences from Industry. In Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools (UML'01). Springer-Verlag, London, 2001.
12. F. J. Heemstra. Software cost estimation. In Information and Software Technology, Vol. 34, No 10, October 1992, Elsevier.
13. Boehm B.: Software Engineering Economics, Prentice Hall, 1981.