Comparative analysis of UCP and SLIM methods to estimate the development of software products

E A Markevich¹, V V Kukartsev¹2, A I Strokan¹, E A Nozdrenko¹, N N Lysyannikova¹ and S Ch Mongush³

¹Siberian Federal University, 79, Svobodny pr., Krasnoyarsk, 660041, Russian Federation
²Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russian Federation
³Tuva State University, 36, Lenin st., Kyzyl, 667000, Russian Federation

E-mail: kempf@sibsau.ru

Abstract. This study is devoted to analysis different methods to estimate the software products development. The following most popular methods were selected for the study: Use Case Points and SLIM. The considered methods are algorithmic and are used in case of insufficient experience of the project team to use empirical calculation methods. A comparative analysis of these methods was also carried out based on calculating the forecasting error for IT projects for various platforms, such as a mobile device and a personal computer. The disadvantages of using the considered methods for developing applications for mobile platforms were identified. Based on the research results, it was concluded that today algorithmic methods for assessing labor intensity cannot give the best results and must be changed to adapt them to mobile software, considering its specific specifics, which makes it advisable to continue researching this industry in direction of mobile development.

1. Introduction

Software development is a rather laborious process [1]. Small projects can very easily be evaluated by an experienced project manager who has previously been involved in similar developments, and, as a rule, super-precision in such minor projects is not required. However, with the growth of software products, the issue of accurately predicting the duration and cost of software becomes more and more urgent. Inappropriate evaluation of the project can affect the success of the development, in particular, lead to its failure. Since the accuracy and reliability of labor intensity estimates are very important to the competitiveness of a company's software, enterprises and researchers have focused their best efforts on developing models for estimating the labor intensity of projects with the highest accuracy.

Dynamically changing technologies have led the economics of software engineering to create new assessment methods and evolve old ones. There are many methods of assessment that can be classified based on their formulation: expert judgment [2], estimates based on analogies [3], algorithmic methods (including empirical methods) [4], rule induction methods [5], approaches based on artificial neural networks [6], approaches based on Bayesian networks [7], decision tree [8] and estimation methods based on fuzzy logic [9]. Thus, it became possible to calculate the values most accurately, which made it possible to more adequately plan any forthcoming activity.
2. Materials and Methods
The Project Management Institute offers information on uncertainty, where the accuracy of the estimate is reduced to an asymmetric cone called the “Software Product Uncertainty Cone”. As you can see from the graph in Figure 1, estimates created very early in a project are prone to high error rates. Estimates created at the initial concept stage may differ from real values up to 4 times. The deviation between the calculated and actual labor intensity should be within the boundaries of the cone, otherwise, the methods should be rejected as inaccurate.

![Figure 1. The cone of uncertainty of a software product.](image)

Unfortunately, in software engineering, labor-intensive estimation is still not the preferred approach. Not all companies use methods for assessing the development labor intensity on an ongoing basis, preferring to rely on their previous experience, using measurement estimates, mainly as a method for additional verification. The refusal to use evaluations and analysis in practice raises an important question - whether the measurement process is generally applicable in software projects. In other branches of technology, scientific results and laws based on measurements are successfully applied in practice, giving valuable help engineering teams in managing their own projects. To determine whether an objective scientific approach can give sufficiently accurate indicators of labor intensity, we will evaluate several well-known methods of measurement and analysis, apply them on several real projects and determine whether these methods can take place in predicting the labor intensity of project development.

Another important question is whether software teams should use the same measurement method when analyzing applications developed for different platforms, or they should look for the most appropriate method for the chosen platform. It can also be challenging related to time, market, user interface constraints, power consumption, and network outages [10, 11]. In practice, the development team is required to provide various evaluations throughout the entire project lifecycle. At the beginning of the project, some rough estimates are needed with minimal involvement and analysis.

At later stages of the project, more accurate and binding estimates are needed. In the open literature, various assessment methods of varying complexity are proposed. Hence, they provide different accuracy. But will this accuracy be the same when evaluating a mobile app or a personal computer app? The most effective way to assess the effort required is to correctly combine different measurement methods with different development platforms. Applying assessment methods to the “right” projects and at the right time points in the project life cycle allows for a continuous assessment process, where
the accuracy of the assessment converges to the desired value as the project develops. The purpose of this assessment is to develop a methodology for appropriate measurement methods for a software project.

**Description of methods.**

Use Case Points methodology considers various criteria in the assessment, such as non-functional requirements, organizational risks, and so on.

The Use Case Points method was developed by Gustav Karner shortly in 1993. This method was developed based on use cases.

The following equation is used for calculations (1):

\[ UCP = UUCP \times TCF \times EF, \]  

(1)

where \( UUCP \) is a generalized weighting indicator; \( TCF \) is the technical complexity of the project; \( EF \) is the skill level of the developers.

When calculating \( UUCP \), it is necessary to determine the actors, divided into three types: simple, actors who use the system using a predefined programming interface (API), middle ones, who use more flexible APIs such as TCP/IP, and complex ones, which represent a person using a graphical interface (GUI) or web page. Each type of characters has a corresponding weighting coefficient: for simple - 1, for medium - 2, for complex - 3.

Further, the use cases and their types are determined, each of which has a corresponding weighting factor: for simple - 5, for medium - 10, for complex - 15. The value of the weighting factor is determined depending on the number of transactions in the event streams (main and alternative). In this case, a transaction is understood as an atomic sequence of actions that is completely executed or canceled.

\( UUCP \) is calculated using the following equation (2):

\[ UUCP = \sum m_i \times a_i + \sum n_j \times b_j, \]  

(2)

where \( m \) is the number of use cases; \( n \) is the number of actors of a particular type; \( a \) is the weight of the actor; \( b \) is the weight of the use case.

To calculate \( TCF \), each of the technical complexity indicators must be assigned a significance level in the range from 0 to 5. The \( TCF \) value is calculated using the following equation (3):

\[ TCF = 0.6 + (0.01 \times \left( \sum T_i \times \text{Weight}_i \right)), \]  

(3)

where \( T \) is the significance of the indicator; \( \text{Weight} \) is the weight of the indicator.

Indicators of technical complexity and their weights are shown in Table 1.

**Table 1. Indicators of technical complexity.**

| Indicator | Description                              | Weight |
|-----------|------------------------------------------|--------|
| T1        | Distributed system                        | 2      |
| T2        | High performance                          | 1      |
| T3        | The work of end-users in the online mode  | 1      |
| T4        | Sophisticated data processing             | 1      |
| T5        | Code reuse                                | 1      |
| T6        | Easy to install                           | 0.5    |
| T7        | Ease of use                               | 0.5    |
| T8        | Portability                               | 2      |
| T9        | Ease of making changes                    | 1      |
| T10       | Parallelism                               | 1      |
| T11       | Special requirements for safety           | 1      |
| T12       | Special requirements for safety           | 1      |
| T13       | Special requirements for user training    | 1      |
When calculating EP, it is also necessary to determine the significance of each indicator of the level of developer skills; the indicators are presented in Table 2.

### Table 2. Developers’ skill level indicators.

| Indicator | Description                               | Weight |
|-----------|------------------------------------------|--------|
| F1        | Acquaintance with technology             | 1.5    |
| F2        | Application development experience       | 0.5    |
| F3        | Experience in using an object-oriented approach | 1     |
| F4        | The presence of leading analyst          | 0.5    |
| F5        | Motivation                               | 1      |
| F6        | Stability of requirements                | 2      |
| F7        | Part-time employment                     | -1     |
| F8        | Complex programming languages            | -1     |

After determining the values, it is necessary to calculate the result using the equation (4):

\[
EF = 1.4 + (-0.03 \times (\sum F_i \times \text{Weight})),
\]

where \( F \) is the significance of the indicator; \( \text{Weight} \) is the weight of the indicator.

Next, using equation (3), we calculate the \( UCP \), after which we convert this value into man-hours. To do this, it is necessary to select the \( K_{ucp} \) coefficient, measured in man-hours, reflecting the standard adopted at a particular enterprise/firm.

The total complexity of development is calculated by the equation (5):

\[
T = UCP \times K_{ucp},
\]

In addition, this study uses the SLIM (Software Lifecycle Management) method. Of the techniques under consideration, SLIM was developed earlier than anyone else in 1978 by Lawrence Putnam. The development of the method was based on empirical data from software developments of the US Department of Defense. The model describes the time and effort required to complete a software project of a given size.

Putnam suggests that staffing levels rise smoothly during a project and then plummet during acceptance testing. The SLIM model is expressed as two equations (the programmatic equation and the labor force accumulation equation) that describe the relationship between development effort and schedule.

The central part of Putnam’s model, called the program equation, is described by the equation (6):

\[
\text{Size} = C \times E^{1/3} \times t^{4/3},
\]

where \( \text{Size} \) is the size of the software product (the number of lines of code); \( C \) is the technological factor; \( E \) is the complexity of the project in person-years; \( t \) is the time of the project implementation in years.

The value of the technological factor \( C \) is usually in the range from 2000 to 12000 and is selected based on the following definitions: \( C = 2000 \) implies low development progress, which manifests itself in the absence of a methodology, incomplete (weak) documentation, and so on. \( C = 8000 \) assumes a reasonably good level of development. \( C = 11000 \) assumes a high level of development (use of automated tools).

The equation for \( E \), the labor accumulation equation, is as follows (7):

\[
E = D \times t^3,
\]

where \( D \) is a coefficient expressing the amount of required labor; \( t \) is the time of project implementation in years.
The $D$ value is determined depending on the type of project. If the software product is new and has a large number of interfaces and interactions with other systems, then $D = 12.3$. For stand-alone software products, $D = 15$. When reworking or changing an existing system, $D = 27$.

After carrying out the transformations, we get (8):

$$E = \frac{(S/C)^{9/7}}{7} \times D^{4/7}.$$  \hspace{1cm} (8)

Methods were tested on two real software projects: traditional PC application “Messages” – to improve the educational process of the Institute by automating the communication of the dean’s office staff with students; mobile application “Kukla Theater” – to attract a new audience and thereby increase the volume of ticket sales.

The models proposed in this study may not be accurate enough to estimate future labor intensity. In order to improve the accuracy and reliability of the proposed methodologies, a certain criterion of accuracy was used, namely, the forecast error should be the smallest. The accuracy and reliability of the method is very important as it directly affects the success or failure of the project as a whole.

The number of lines of source code was predicted using the Function Points method [10, 11].

3. Result

Calculations of the assessment of labor intensity for the implementation of the project aim to develop the module “Messages”.

Use Case Points: let us define the characters and their types:

- User is a complex type.
- Administrator is a complex type.
- Database - medium type.
- Server - medium type.

Next, let us define the use cases:

- User authorization - complex type.
- Choice of the addressee - medium type.
- Attachment of attachments - simple type.
- Sending messages - complex type.
- View message history - simple type.
- Search by messages - simple type.
- Add messages to favorites - simple type.
- Removing messages from favorites - simple type.

Using equation (2), we calculate (9):

$$UUCP = (1 \times 0 + 2 \times 2 + 3 \times 2) + (5 \times 5 + 10 \times 1 + 15 \times 2) = 75.$$  \hspace{1cm} (9)

To assess the technical factors, we use Table 3. Let us assign values to the technical indicators; in this case the factor $T6$ has no significance. Then by equation (3), we get (10):

$$TCF = 0.6 + (0.01 \times (2 \times 2 + 1 \times 5 + 1 \times 5 + \cdots + 5 + 1 \times 1 + 1 \times 1)) = 0.925.$$  \hspace{1cm} (10)

The skill level of developers is determined in a similar way. The level of qualification of developers is determined considering the indicators presented in table 4 according to equation (5), we get (11):

$$EF = 1.4 + (-0.03 \times (1.5 \times 4 + 0.5 \times 3 + 1 \times 3 + 0.5 \times 2 + 1 \times 2 + 2 \times 3 + (-1) \times 3 + (-1) \times 2)) = 0.965.$$  \hspace{1cm} (11)

The final $UCP$ (12) value is found using equation (1):

$$UCP = 75 \times 0.925 \times 0.965 = 66.95.$$  \hspace{1cm} (12)
To convert $UCP$ into man-hours, look at Table 2. Equations (1)-(6) have two values less than 3 and equations (7)-(8) have 1 value greater than 3. Therefore, you should take 28 man-hours per $UCP$. We get (13) according to the equation (7):

$$T = 66.95 \times 28 = 1874.6 \text{ person } \times \text{ hour} = 11.7 \text{ person } \times \text{ month}. \quad (13)$$

SLIM. For the selected project, we have the following values: $Size = 2800$, $C = 8000$, $D = 23$. Using equation (6), we calculate the complexity (14):

$$E = (2800/8000)^{9/7} \times 23^{4/7} = 1.6 \text{ person } \times \text{ year} = 19.2 \text{ person } \times \text{ month}. \quad (14)$$

Calculations of the assessment of labor intensity for the implementation of a project to develop a mobile application for a puppet theater.

Use Case Points: let us define the characters and their types:
- User - medium type.
- Administrator is a complex type.
- Database - medium type.
- Server - medium type.
- Use cases and their types:
  - User authorization - complex type.
  - Viewing the latest news - simple type.
  - Viewing Posters - simple type.
  - Viewing the schedule - simple type.
  - View events - simple type.
  - View information about the theater - simple type.
  - View Theatre contacts - simple type.
  - Going to the theater group on the Vkontakte / Instagram / Facebook site is a simple type.

Using equation (1), we calculate (15):

$$UUCP = (1 \times 0 + 2 \times 3 + 3 \times 1) + (5 \times 7 + 10 \times 0 + 15 \times 1) = 59. \quad (15)$$

To assess technical factors, we use Table 3. Assigning values to technical indicators and find $TCF$ using equation (2), we get (16):

$$TCF = 0.6 + (0.01 \times (2 \times 2 + 1 \times 3 + 1 \times 2 + 1 \times \cdots + 2 + 1 \times 2 + 1 \times 3 + 1 \times 2 + 1)) = 0.855. \quad (16)$$

The level of qualification of developers is determined considering the indicators presented in Table 2. Considering certain data, we find $EF$ (17) by equation (3):

$$EF = 1.4 + (-0.03 \times (1.5 \times 4 + 0.5 \times 4 + 1 \times 3 + 0.5 \times 3 + 1 \times 4 + 2 \times 3 + (-1) \times 2 (-1) \times 2)) = 0.845. \quad (17)$$

The final $UCP$ (18) value is found using equation (1):

$$UCP = 59 \times 0.855 \times 0.845 = 42.6. \quad (18)$$

To convert to man-hours, look at Table 2, the equations (1)-(6) indicators have one value less than 3, and equations (7), (8) do not have values greater than 3, therefore, you should take 20 man-hours per $UCP$. By equation (7), we get (19):

$$T = 42.6 \times 20 = \text{person } \times \text{ hour} = 5.3 \text{ person } \times \text{ month}. \quad (19)$$

SLIM: the following values are available: $Size = 5000$, $C = 8500$, $D = 13$. Using equation (6), we calculate the labor intensity (20):

$$E = (5000/8500)^{9/7} \times 13^{4/7} = 2.2 \text{ person } \times \text{ year} = 26.4 \text{ person } \times \text{ month}. \quad (20)$$
To estimate the forecast error, the absolute prediction error was the actual labor intensity for the module "Messages" = 4 months × 2 people = 8 people × months; the actual labor intensity for the "Puppet Theater" application = 2 months × 1 person = 2 people × month.

Table 3 shows the calculation results.

| Assessment method | Module "Messages" | Application "Puppet Theatre" |
|-------------------|-------------------|-----------------------------|
| Use Case Points   | 46.25%            | 165%                        |
| SLIM              | 140%              | 1220%                       |

The results are also shown in the histogram in Figure 2.

Based on the results of Table 3, we can conclude that the method showing the smallest forecast error when developing an application both for a personal computer and for a mobile platform is the Use Case Points method.

4. Conclusion

The low effectiveness of the SLIM method on these projects is due to its focus on larger projects containing more than 70,000 lines of code and the fact that it is expected that evaluation using SLIM will not be carried out before design and coding. This model assumes that effort for software projects is distributed similarly to the collection of Rayleigh curves, but this relationship has not been confirmed in projects with a small project team and a small amount of work. Such a high error rate is not uncommon for this method, as was shown in a study by Chris Kemerer [12, 13].

In the modern world, technologies are rapidly developing, becoming more complex and diversified. Based on the research results, it can be concluded that, since the most popular algorithmic methods for assessing labor intensity appeared before the creation of mobile platforms, today they cannot give the best results. Moreover, they must be changed in order to adapt them to mobile software with considering its specific specifics [14-16], which makes it advisable to continue researching this industry in the direction of mobile development.

References

[1] Evdokimov I V 2009 The problem and indicators of the quality of software Proceedings of Bratsk State University, Series: Economics and Management 1 121-4 Evdokimov I V 2009 The problem and indicators of the quality of software Proceedings of Bratsk State University.
Series: Economics and Management 1 121-4

[2] Chiu N H and Huang S J 2007 The Adjusted Analogy-Based Software Effort Estimation Based on Similarity Distances Journal of Systems and Software 80(4) 628-40

[3] Kaczmarek J and Kucharski M 2004 Size and Effort Estimation for Applications Written in Java Journal of Information and Software Technology 46(9) 589-60

[4] Jeffery R, Ruhe M and Wieczorek I 2001 Using Public Domain Metrics to Estimate Software Development Effort In: 2001 Proceedings of the 7th International Symposium on Software Metrics, IEEE Computer Society pp 16-27

[5] Srinivasan K and Fisher D 1995 Machine learning approaches to estimating software development effort IEEE Transactions on Software Engineering 21 126-37

[6] Subramanian G H, Pendharkar P C and Wallace M 2006 An Empirical Study of the Effect of Complexity, Platform, and Program Type on Software Development Effort of Business Applications. Empirical Software Engineering 11 541-53

[7] Selby R W and Porter A A 1988 Learning from examples: generation and evaluation of decision trees for software resource IEEE Transactions on Software Engineering 14 1743-57

[8] Huang S J, Lin C Y and Chiu N H 2006 Fuzzy Decision Tree Approach for Embedding Risk Assessment Information into Software Cost Estimation Model Journal of Information Science and Engineering 22(2) 297-313

[9] Rose K H 2013 A guide to the project management body of knowledge (PMBOK® Guide)—Fifth Edition Project management journal 3(44) 123-47

[10] John T M D, Bharathi C R 2017 E-Commerce System Size using User Based Function Points International Journal of Applied Engineering Research 12(16) 6115-22

[11] Kukartsev V V and Sheenok D A 2012 Estimation of the costs of upgrading the software of systems critical in terms of reliability Bulletin of Siberian State Aerospace University named after A.I. Academician M.F. Reshetnev 5(45) 62-5

[12] Altaleb R A and Gravell M A 2018 Effort Estimation across Mobile App Platforms using Agile Processes: A Systematic Literature Review Journal of Software 13(4) 242-59

[13] Kaur A and Mahl A 2019 Effort Estimation for Mobile Applications using Use Case Point (UCP) Smart Innovations in Communication and Computational Sciences pp 163-72

[14] Xanthopoulos S and Xinogalos S 2013 A comparative analysis of cross-platform development approaches for mobile applications In: CI ’13 Proceedings of the 6th Balkan Conference in Informatics pp 213-20

[15] Hammershoj A, Sapuppo A and Tadayoni R 2010 Challenges for mobile application development In: Proceedings of the 14th Int. Conf. Intell. Next Gener. Networks (ICIN 2010) Second Int. Work. Bus. Model. Mob. Platforms (BMMP 2010) pp 1-8

[16] Wasserman A I 2010 Software engineering issues for mobile application development In: Proceedings of the FSE / SDP Work. Futur. Softw. Eng. Res. ACM pp 397-400