

Метою даного дослідження є розроблення методу оцінювання кількості автоматично генерованого програмного коду для підрахунку кількості рядків автоматично генерованого програмного коду та реальної кількості рядків коду, створеного саме програмістом, яка дасть можливість більш точно і достовірно оцінити трудомісткість розроблення програмного забезпечення на основі LOC-оцінки.

Ключові слова: програмне забезпечення (ПЗ), трудомісткість розроблення ПЗ, метод алгоритмічного моделювання, LOC-оцінка, автоматично генерований програмний код.

T.O. HOVORUSHCHENKO, V.M. STECYUK, Y.L. NOVYTSKYI
Khmelnitsky National University

METHOD OF EVALUATING THE QUANTITY OF AUTOMATICALLY GENERATED SOURCE CODE

The aim of this study is the development of: the method of evaluating the quantity of automatically generated source code for the calculating the number of lines of automatically generated source code and the real quantity of lines of code, which created by the programmer. It will provide the more accurate and reliable evaluations of the efforts for software development on the basis of the LOC-evaluations. Software development is not always successful due to non-compliance with the planned project deadlines or budget overruns, which is often due to an incorrect assessment of the efforts for the software development. One of the main methods for evaluating the efforts for software development is algorithmic simulation - the method, in which the dependence of the efforts for the project on a quantitative indicator of software is determined. The most influential factor of evaluating the efforts in these models is the number of lines of code or LOC-evaluation. As the analysis showed, LOC-evaluation has many problems, including the distortion of LOC-evaluation, because it is impossible to separate the number of lines of code, which is automatically generated by the development environment, and the number of lines of code, which is created manually by the programmer. In this paper, the production rules and the method for evaluating the quantity of automatically generated source code are developed. These rules and method provide: evaluating the number of lines of automatically generated code; evaluating the number of using of each automatically generated element for the concrete programming language; evaluating the number of lines of code, which is written by the programmer. The prospect for further authors' research is the development of the software tool for evaluating the quantity of automatically generated source code on the basis of the developed production rules and method of evaluating the quantity of automatically generated source code. This tool will analyze the source code and will evaluate the number of lines of automatically generated code, the number of using of each automatically generated element for the concrete programming language, and the number of lines of code, which is written by the programmer.

Keywords: software, efforts for the software development, method of algorithmic simulation, LOC-evaluation, automatically generated source code.

[1–3].

[1, 4–6]:

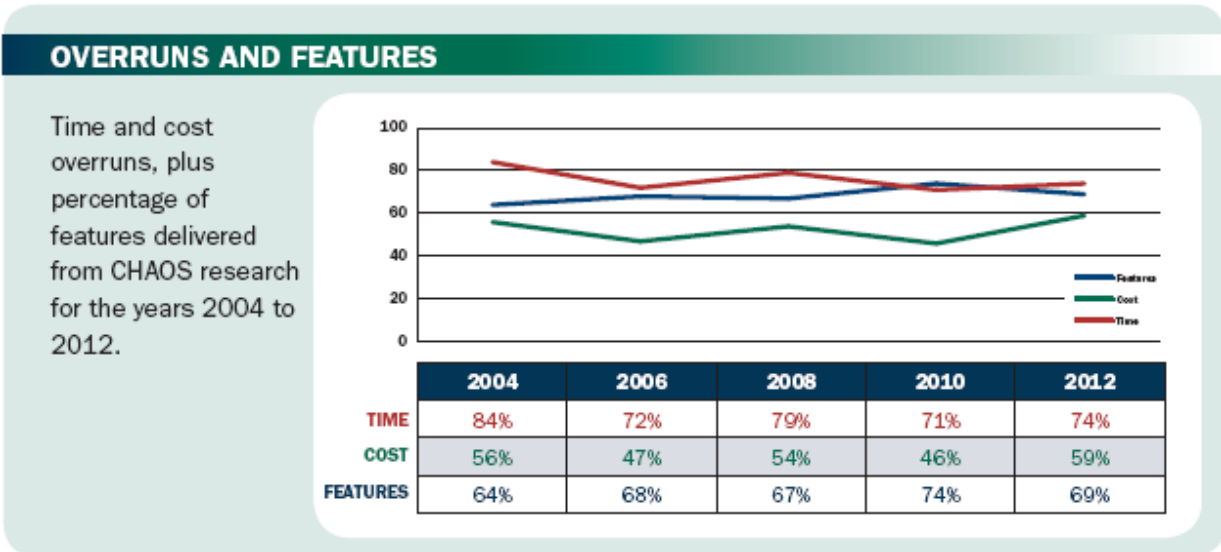
- 1) ();
- 2) ();
- 3) ();
- 4) ().

[7, 8].

The Standish Group International (CHAOS reports) [7, 8], . 1. [7, 8], 16% i

9% McKinsey & Company [9] University of Oxford, i

15 , : 66%, – 33%, – 17% (. 2, 3 [9]).



.1.

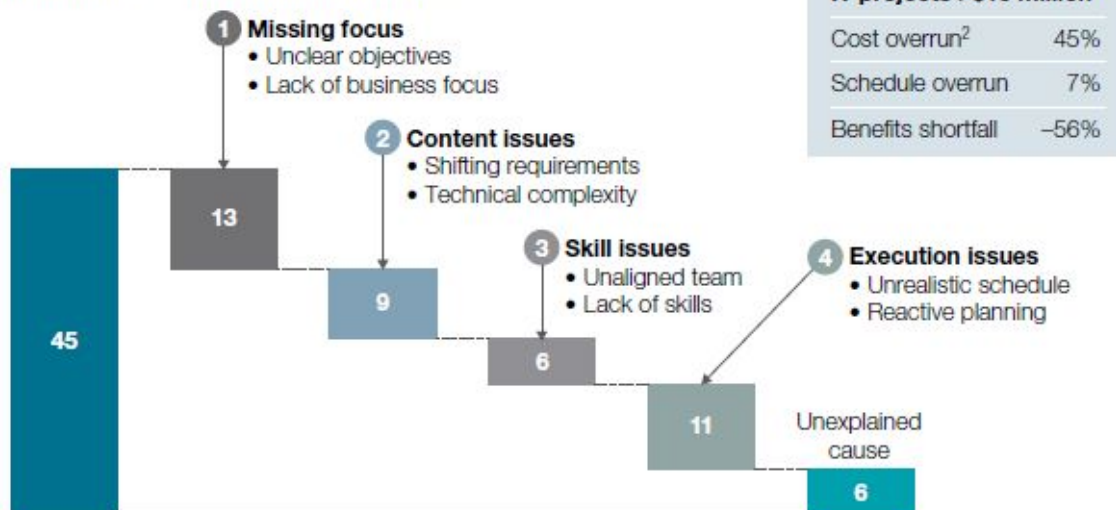
% , projects >\$15 million, in 2010 dollars

Project type	Average cost overrun	Average schedule overrun	Average benefits shortfall
Software	66	33	17
Nonsoftware	43	3.6	133
Total	45	7	56

Source: McKinsey–Oxford study on reference-class forecasting for IT projects

.2.

Rough cost-overrun disaggregation, %



¹With cost overrun, in 2010 dollars.

²Cost increase over regular cost.

Source: McKinsey–Oxford study on reference-class forecasting for IT projects

.3.

$$AC = \{el_1, el_2, \dots, el_n\}, \tag{3}$$

$$SC = \{line_1, line_2, \dots, line_{LOC}\}, \tag{4}$$

CodeStyle, ..).

- 1) $el_1 \in line_1, LOC_{auto_1} = LOC_{auto_1} + 1;$
- 2) ...
- 3) $el_i \in line_1, LOC_{auto_i} = LOC_{auto_i} + 1;$
- 4) ...
- 5) $el_n \in line_1, LOC_{auto_n} = LOC_{auto_n} + 1;$
- 6) ...
- 7) $el_1 \in line_j, LOC_{auto_1} = LOC_{auto_1} + 1;$
- 8) ...
- 9) $el_i \in line_j, LOC_{auto_i} = LOC_{auto_i} + 1;$
- 10) ...
- 11) $el_n \in line_j, LOC_{auto_n} = LOC_{auto_n} + 1;$
- 12) ...
- 13) $el_1 \in line_{LOC}, LOC_{auto_1} = LOC_{auto_1} + 1;$
- 14) ...
- 15) $el_i \in line_{LOC}, LOC_{auto_i} = LOC_{auto_i} + 1;$
- 16) ...
- 17) $el_n \in line_{LOC}, LOC_{auto_n} = LOC_{auto_n} + 1.$

$$AC = \{el_1, el_2, \dots, el_n\}$$

$$AC_{concrete} = \{el_{1_{concrete}}, el_{2_{concrete}}, \dots, el_{n_{concrete}}\},$$

[12]
 $AC_{concrete},$
)

$$LOC_{auto} = \sum_{i=1}^n LOC_{auto_i}, \tag{5}$$

);
 4) ;

$$LOC_{by_hand} = LOC - LOC_{auto}, \quad (6)$$

$$LOC - LOC_{auto} = \dots, \quad (5).$$

LOC-

LOC-

1)

2)

3)

()

1. // , 22–24 2015 .
2. , 2015. – . 75–82.
3. / . . – : 1136 .
4. / . . – : " ", 2013. – 896 .
5. « » / . . – New York : John Wiley & Sons, 2014. – 76 .
6. / . . – : « - », 2010. – 336 .
7. The Standish Group International: CHAOS Manifesto – Think big, act small. Technical report, CHAOS Knowledge Center (2013). URL: <http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf> (viewed on February 5, 2018). – Title from the screen.
8. Hastie Shane. Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch / Shane Hastie, Stéphane Wojewoda. URL: <http://www.infoq.com/articles/standish-chaos-2015> (viewed on February 5, 2018). – Title from the screen.
9. . . .

