УДК 004.412.3

МЕТОДИКА ОЦЕНКИ ТРУДОЕМКОСТИ ВЫПОЛНЕНИЯ ПРОГРАММНОГО ПРОЕКТА

Варламов А.С.
Научный руководитель – канд. техн. наукЯкунин Ю.Ю.
Сибирский федеральный университет

В обеспечении конкурентных преимуществ организаций, занимающихся разработкой программного обеспечения, не малую роль играют методы оценки трудоёмкости и стоимости программных проектов. На практике ошибка оценки трудозатрат программного проекта относительно фактических может достигать 150-200%, поэтому увеличение точности оценки трудозатрат является важной проблемой, требующей решение. На точность оценки влияет множество факторов [2], таких как: неопределённость требований к системе, человеческий фактор, неправильная оценка требуемых работ, неопытность эксперта или неточность модели оценки, риски и др. Методы оценки направлены на расчёт ***сроков выполнения*** проекта и ***ресурсов*** (стоимость, персонал, помещения и др.).

Для оценки сроков выполнения проекта существует множество методик, но зачастую в них не учитывается человеческий фактор. Разные программисты сделают одну и ту же работу за разные промежутки времени, потому что обладают разными навыками, опытом и знаниями. Поэтому для повышения точности оценки необходимо учитывать навыки и квалификацию каждого программиста.

Предлагаемая методика основана на принципах, изложенных в [1], и использует подход функциональных точек. Для более точной и качественной оценки в методику включены механизмы учёта квалификации программистов, а принципы расчёта комбинируют подходы микрооценки и макрооценки [3]. Исходными данными для предлагаемой методики являются статистические данные по выполненным ранее проектам, которые имеют следующую структуру.

1. Общая информация о проектах: L^k – длительность выполнения проекта, T^k – трудозатраты на выполнение проекта, β^k – удельный вес проекта (относительная достоверность и важность информации в проекте), EC^k – количество ошибок в проекте, список блоков. Здесь $k=1,n$, где n – количество анализируемых проектов.

2. Информация по каждому блоку каждого проекта: L_i^k – длительность выполненияблока, T_i^k – трудозатраты на выполнение блока, EC_i^k –количество ошибок на блок. Здесь $i=1,p$, где p–количество блоков в проекте.

3. Информация по каждому элементу блока: $ElC_{l,m,j}^{k,i}$–количество элементов данного типа в блоке, $T_{j,l,m}^{k,i}$ – трудозатраты на элемент l-го типа в этом блоке. Здесь $j=1,o$, где o – количество номер группы элементов в блоке, $l=1,r$, где r – количество типов элементов, $m=1,s$, где s – количество исполнителей.

4. Информация по типам элементов: σ_l – относительная сложность, T_l–средние трудозатраты на данный тип элемента, L_l – средняя длительность выполнения. Здесь $l = 1,r$, где r – количество типов элементов.

5. Информация по исполнителям: $Skill^m$ – навык работы (если он рассчитывался ранее), $ek^{m,j}$ – коэффициент обучаемости. Здесь $m=1,s$, где s – количество исполнителей.

6. Список ошибок с указанием исполнителя, который их совершил и номера элемента, блока и проекта где находится ошибки.

Статистические данные о ранее выполненных проектах должны пройти определённую подготовку перед использованием в предлагаемой методике. Для этого необходимо выполнить следующие шаги.

1. Задаются коэффициенты относительной сложности каждого типа элементов (если это требуется). Они могут определяться экспертом, либо рассчитываться по имеющимся данным по уже выполненным проектам, причем:

$$\sum_i \sigma_i = 1 \tag{1}$$

2. Далее необходимо определить структуру программного проекта, т.е. определяются блоки и элементы в них. Структура определяется исходя из функциональных требований к будущему проекту.

3. Определяются коэффициенты значимости для проектов. Распределение коэффициентов значимости проектов может быть любым, в зависимости от требований организации. Например, они могут быть установлены в зависимости от процентного совпадения структуры будущего проекта с уже выполненным. Чем больше структуры совпадают, тем выше коэффициент значимости. Причем это коэффициент нормированный (2), т.е. в сумме коэффициенты значимости всех проектов дают 1.

$$\sum_{k=1}^{n} \beta_k = 1 \tag{2}$$

После определения структуры будущего проекта и определения значимости выполненных проектов можно переходить к алгоритму оценки, который состоит из следующих шагов.

1. Рассчитываются средние трудозатраты на каждый тип элемента в каждом блоке каждого проекта с помощью коэффициентов относительной сложности.

$$T_{i,l}^{k} = \frac{\sigma_l \cdot E_{l,i}^{k}}{\sum_j \sigma_l \cdot E_{l,i}^{k}} \cdot T_l^{k}, \tag{3}$$

,где $E_{l,i}^{k}$- количество элементов *l*-го типа в *i*-ом блоке k -го проекта;

2. Рассчитываются трудозатраты на каждый тип элемента (*l*) по всей базе знаний о выполненных проектах, вне зависимости от исполнителя.
   2.1. Проверяется, есть ли рассчитываемый тип элемента в каждом проекте. Если нужного типа элемента в проекте нет, то перерассчитываются коэффициенты значимости проекта по формуле:

$$\beta^k = \beta^k + (\beta^k / \Sigma_k \beta^k \cdot \beta_p^k), \tag{4}$$

   где k- номерпроекта, p- номер проекта в котором нет нужного типа элемента, причем $k \neq p$.

   2.2. С помощью формулы (3) рассчитываются трудозатраты на каждый тип элемента по информации о всех проектах:

$$T^l = \Sigma_k \beta^k \cdot \left(\Sigma_i T_i^{k,l} / N \right), \tag{5}$$

   где *N*- количество блоков в k-ом проекте, *l* – номер типа элементов.

3. Для каждого исполнителя рассчитываются средние трудозатраты на каждый тип элемента с помощью формулы (3)для каждого проектапо формуле:

$$T_l^{k,m} = \Sigma_i \frac{\left(\frac{\sigma_l \cdot E_{l,i}^{k}}{\Sigma_i \sigma_l \cdot E_{l,i}^{k}} \right) \cdot T_i^{k}}{N}, \tag{6}$$

где $T_i^{k,m}$ - трудозатраты на тип элемента в конкретном проекте, выполненный данным программистом.

Этот параметр необходим для определения трудозатрат на каждый тип элемента для конкретного программиста. Далее рассчитываются средние трудозатраты на каждый тип элемента, для каждого исполнителя по всем проектам:

$$T_i^m = \sum_i \beta^k \cdot T_i^{k,m}, \quad (7)$$

где T_i^m- трудозатраты на *l*- ый тип элемента для *m*- го исполнителя.

4. Оценка квалификации программиста.Рассчитываются навык работы программиста и его коэффициент обучаемости для каждого типа элемента.Под «навыком работы» понимается показатель отношения скорости работы программиста к качеству выходного продукта. Таким образом, критерий «навык работы» характеризует обобщенный показатель скорости и качества работы. Навык работы ($Skill^{m,l}$) рассчитывается по формуле:

$$Skill^{m,l} = WS^{l,m}/CS^{l,m}, \quad (8)$$

где $WS^{l,m}$ – скорость работы *m*-го исполнителя над *l*-м типом элемента, $CS^{l,m}$ – стабильность кода программиста.

Этот параметр отражает скорость и качество работы программиста.

4.1. Скорость работы программиста $WS^{l,m}$ – это средняя скорость работы программиста над элементами того или иного типа (*эл/час*). Позволяет оценить трудозатраты требуемые программисту для выполнения функционального элемента. Он рассчитывается по формуле:

$$WS^{l,m} = \frac{\sum_k \frac{ElC^{l,m}}{t^{l,m}}}{n}. \quad (9)$$

4.2. Стабильность кода – это критерий, который показывает качество кода программиста. Это показатель сложности исправления ошибки и как следствие, дополнительные трудозатраты на их исправление. Этот критерий рассчитывается по формуле:

$$CS^{l,m} = SEC^{l,m} \cdot SCL^{l,m}. \quad (10)$$

4.3. Среднее количество ошибок, совершаемых исполнителем на определенный тип элемента ($SEC^{l,m}$), рассчитывается как среднее количество ошибок на каждый тип функционального элемента по формуле:

$$SEC^{l,m} = \frac{\sum_k \frac{ErC_k^{l,m}}{ElC_k^{l,m}}}{n}. \quad (11)$$

4.4. Средний уровень критичности ошибок ($SCL^{l,m}$), показывает средний уровень критичности ошибок, совершаемых исполнителем при выполнении определенного типа элемента. Рассчитывается по формуле:

$$SCL^{l,m} = \frac{\sum_k \frac{\sum_i ECL^{l,m}}{ErC^{l,m}}}{n}. \quad (12)$$

4.5. Рассчитывается коэффициент обучаемости программиста для каждого типа элемента, как приращение $Skill$. Это критерий показывает прирост скорости работы и качества. Рассчитывается по формуле:

$$ek^{l,m} = Skill_n^{l,m}/Skill_{n-1}^{l,m}. \quad (13)$$

5. Оптимизация распределения.Для каждого типа элементов (или для каждого блока) определяется исполнитель. Для этого выполняется поиск оптимального кандидата на выполнение каждого элемента или же блока.

$$T_i^{\text{оц}} = min_m\{T^{l,m}\}, \quad (14)$$

где l −номер типа элемента,m −номер программиста,T – показатель средних трудозатрат на разработку типа элемента конкретного программиста.

В зависимости от требуемых результатов можно использовать следующие критерии выбора программиста:навык работы *Skill*формула (8), скорость работы *WS*формула (9), качество работы (стабильность кода *CS*формула (10)), прогнозируемый навык работы (*f(ek,Skill)*).Критерии можно учитывать совместно, тогда задача преобразуется в задачу многокритериальной оптимизации.

$$f\left(\bar{k}_m\right) \to \min_{\bar{k}_m \in \bar{K}} \quad (17)$$

где$\bar{K} \in (Skill, CS, WS, f(ek, Skill))$, m = 1,n−количество исполнителей.

6. После суммирования получается оценка трудоемкости программной системыT^k.

$$T_{k,l}^{\text{оц}} = \sum_i T_{i,k,l}^{\text{оц}} \times ElC_i^{k,l}, \text{причем} \quad (16)$$

$$T_{i,k,l}^{\text{оц}} = min\{T_{l,m}^{\text{оц}}, T_l\} \quad (17)$$

Описанная выше методика, позволяет учитывать квалификацию каждого программиста. Оценка производится по критериям скорости и качества работы. Использование описанных выше критериев обеспечивает более высокую точность оценки будущих трудозатрат.

Блок оптимизации может помочь проектно-ориентированным организациям более эффективно распределить имеющиеся трудовые ресурсы в зависимости от требований к конечному продукту (будь то скорость выполнения или качество). Это позволит увеличить конкурентоспособность организации в условиях рыночной экономики.

Библиографический список

1. Якунин, Ю.Ю. Оценкатрудоёмкостиразработкипрограммнойсистемы / Ю.Ю. Якунин // Вестниксибирскогогосударственногоаэрокосмического университета имени академика М.Ф. Решетнёва. – Красноярск: СибГАУ, 2008. – Вып. 2(19). – С. 87-91.

2. Andrew Stellman and Jennifer Greene (2005). AppliedSoftwareProjectManagement. Sebastopol, MA: O'Reilly Media. ISBN0-596-00948-8.

3. Longstreet D. Function Point Analysis Training Course. / Longstreet Consulting Inc., 2004. – 280 p.