КОМПЛЕКСНАЯ ОЦЕНКА ИНДИВИДУАЛЬНОГО ТРУДА РАЗРАБОТЧИКОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В.В. Марширов,
кандидат технических наук, старший научный сотрудник, доцент кафедры информационных систем и технологий Национального исследовательского университета «Высшая школа экономики» – Нижний Новгород

Л.Е. Марширова,
кандидат экономических наук, доцент кафедры бухгалтерского учета, анализа и аудита Национального исследовательского университета «Высшая школа экономики» – Нижний Новгород

E-mail: vmarshirov@hse.ru, lmarshirova@hse.ru
Адрес: г. Нижний Новгород, ул. Большая Печерская, д. 25/12

Для комплексной оценки индивидуального труда разработчиков программного обеспечения предложен метод расстановки приоритетов. Изложены подходы к определению факторов, влияющих на оценку результатов их труда, к ранжированию факторов и к ранжированию разработчиков по каждому фактору. Предложенная методика может быть использована при разработке регламентов управленческого учета с целью совершенствования управления персоналом с учетом стратегических и тактических задач организации.

Ключевые слова: программирование, управленческий учет, метод расстановки приоритетов, оценка труда разработчиков программного обеспечения, управление персоналом.

1. Введение

При разработке программного обеспечения, как и при любом другом производственном процессе, основными факторами, определяющими стоимость и сроки создания продукта, являются сложность и объем решаемой задачи, стоимость и доступность необходимых ресурсов. Поэтому большое значение имеют критерии и метрики, позволяющее оценивать затраты на создаваемый продукт, в том числе и трудовые затраты.

Однако существуют трудности в оценке затрат на разработку программного обеспечения, так как необходимо учитывать множество разнообразных факторов. Например, необходимый уровень абстракции при разработке, сложность, объем и др. [1]. Поэтому разработаны метри-

ки, позволяющие осуществлять количественные оценки. Например, McCabe's metric – учитывает число условий в программе, Halstead's metrics – число операторов, длину программы и др. [2]. Анализ развиваемых метрик представлен в работе [3]. Кроме формальных метрик, оценивающих затраты при разработке программного обеспечения, важное значение имеет человеческий фактор. Обзор основных источников в этой области представлен в работе [4], в которой перечислены основные фазы разработки программного обеспечения (определение, анализ, структурирование, кодирование и др.) и перечислены личностные факторы, которые влияют на способность к программированию (настойчивость, способность сосредотачиваться, высокая мотивация, стрессоустойчивость и др.).

Однако в указанных метриках рассмотрены лишь вопросы оценки затрат, но не затрагиваются вопросы активной мотивации на конечный результат. Вследствие того, что мотивация является одним из важнейших факторов успеха разработки программного продукта, в данной работе рассматриваются вопросы повышения мотивации через многогранную оценку труда программистов.

Особенность процесса разработки программного обеспечения состоит в преобладании в нем умственного труда и элементов творчества. Этот процесс включает в себя достаточно широкий спектр действий. Образное описание требований, предъявляемых к разработчикам программных средств, дано чл.-корр. АН СССР А.П.Ершовым, одним из пионеров теоретического и системного программирования в нашей стране: «... программист должен обладать способностью первоклассного математика к абстракции и логическому мышлению в сочетании с эдисоновским талантом сооружать все, что угодно, из нуля и единицы. Он должен сочетать аккуратность бухгалтера с проницательностью разведчика, фантазию автора детективных романов с трезвой практичностью экономиста. А кроме того, программист должен иметь вкус к коллективной работе, понимать интересы пользователя и многое другое» [5]. На основании многочисленных психологических и экспериментальных исследований определены требования к профессиональным и личностным качествам программистов. Из профессиональных – это способность к абстракции, понимание отношений, структурирование объектов и выделение иерархических связей между ними, наблюдательность, способность к аналогиям, богатство ассоциаций, гибкость, понимание деталей, ясность выражений и др. Из личностных – самостоятельность, добросовестность, последовательность, внимательность, аккуратность, направленность на сотрудничество.

Важно отметить, что большой объем и сложность современных программных средств не позволяют использовать для их разработки обособленных специалистов. В создании каждого программного продукта участвует коллектив разработчиков разной специализации и квалификации. Причем, в таких коллективах много времени и сил тратится на индивидуальные и групповые взаимодействия, и в этом случае важнейшую роль играют формы организации и мотивации сотрудников.

Однако умственный характер труда разработчиков, содержащий большое количество творческих элементов, и коллективный характер итогового результата затрудняют планирование, учет и контроль работ по созданию программных средств. Поэтому актуальны инструменты, позволяющие оценивать индивидуальный вклад каждого исполнителя в результаты коллективного труда и учитывающие особенности труда, а также стратегические и тактические цели организации.

2. Метод расстановки приоритетов

Для совершенствования механизмов мотивации сотрудников на конечные результаты предлагается методика комплексной оценки индивидуального труда разработчиков программного обеспечения. Методика использует метод расстановки приоритетов, который базируется на теории иерархий Саати [6].

В соответствии с методом расстановки приоритетов осуществляется попарное сравнение объектов по выбранному фактору $f(f=\overline{1,l})$. Результаты сравнения представляются в виде матрицы парных сравнений $A_f = \|a_{fij}\|$, причем, на пересечении строки i и столбца j проставляется оценка предпочтения объекта i над объектом j по фактору f:

$$a_{fij} = \begin{cases} 1+y, & \text{если объект } i \text{ предпочтительнее} \\ & \text{объекта } j \text{ по фактору } f; \\ 1, & \text{если объекты } i \text{ и } j \text{ равноценны по фактору } f; \\ 1-y, & \text{если объект } j \text{ предпочтительнее} \\ & \text{объекта } i \text{ по фактору } f, \end{cases}$$

где y — отклонение элементов матрицы A_f, которое может изменяться в пределах: $0 < y < 1$.

Суммируя по строкам матрицы A_f оценки предпочтения a_{fij}, для каждого объекта i определяется интегрированная сила первого порядка $P_{fi}(1)$ по фактору f:

$$P_{fi}(1) = \sum_{j=1}^{n} a_{fij}$$

где n – количество объектов.

Рассчитанные интегрированные силы первого порядка по фактору f по всем сравниваемым объектам можно представить в виде вектора

$$P_f(1) = \left[P_{f1}(1), P_{f2}(1), ..., P_{fi}(1), ..., P_{fn}(1)\right]$$

Однако показатель интегрированной силы первого порядка объекта i $P_{fi}(1)$ не учитывает «силу» других объектов, у которых объект i «выиграл» или которым объект i «проиграл» при сравнении по фактору f.

Поэтому для получения большей точности оценок для каждого объекта определяется нтегрированная сила второго порядка с учетом «сил» всех сравниваемых объектов:

$$P_{fi}(2) = \sum_{j=1}^{n} a_{fij} P_{fi}(1),$$

где $i = j = \overline{1, n}$; n– количество объектов

Дальнейшие итерации производятся аналогично и интегрированные силы порядка k рассчитываются по следующей формуле в матричном виде:

$$P_f(k) = A_f P_f(k-1),$$

где $P_f(0) = (1, 1, ...1)$.

Этот алгоритм расчета в отличие от простого суммирования оценок предпочтения позволяет учесть косвенные преимущества всех других объектов, с которыми производилось сравнение.

Вопрос определения количества итераций будет рассмотрен в разделах 3 и 4.

Для удобства использования и возможности сравнения интегрированные силы объектов порядка k нормируются, то есть представляются в шкале с фиксированной суммой оценок, равной единице:

$$P_{fi}^{отн}(k) = \frac{P_{fi}(k)}{\sum_i P_{fi}(k)},$$

где $P_{fi}^{отн}(k)$ – нормированная интегрированная сила (приоритет) порядка k объекта i по фактору f.

Поскольку $\sum_{i=1}^{n} P_{fi}^{отн}(k) = 1$, то численное значение приоритета $P_{fi}^{отн}(k)$ характеризует относительную степень выраженности фактора f у объекта i.

Для практического использования метода расстановки приоритетов была разработана информационная среда, с помощью которой методом парных сравнений можно:

- ✦ определять приоритеты факторов, влияющих на величину комплексной оценки индивидуального труда;
- ✦ определять приоритеты оцениваемых объектов (членов группы разработчиков) по каждому фактору;
- ✦ рассчитать значения комплексных оценок индивидуального труда разработчиков;
- ✦ осуществлять распределение премиального фонда (надбавок) между членами группы разработчиков с использованием комплексных оценок индивидуального труда.

Рассмотрим подробнее перечисленные выше алгоритмы расчетов.

3. Определение значимости факторов, влияющих на величину комплексной оценки индивидуального труда

При использовании этого режима необходимо:

- ✧ сформировать матрицу парных сравнений факторов, влияющих на величину комплексной оценки индивидуального труда ($B = \|b_{ij}\|$);
- ✧ определить величину отклонения элементов матрицы парных сравнений (y);
- ✧ задать порядок, используемый в методе расстановки приоритетов (k).

Матрицу парных сравнений факторов, влияющих на комплексную оценку индивидуального труда (КОИТ), должна сформировать группа экспертов (например, администрация подразделения), причем,

$$b_{ij} = \begin{cases} 1 + y, \text{ если фактор } i \text{ оказывает большее влияние на КОИТ, чем фактор } j; \\ 1, \text{ если факторы } i \text{ и } j \text{ оказывают одинаковое влияние на КОИТ;} \\ 1 - y, \text{ если фактор } j \text{ оказывает большее влияние на КОИТ, чем фактор } i. \end{cases}$$

Значение величины отклонения y предлагается определять в соответствии с *табл. 1.*

Таблица 1.

Значение величины отклонения элементов матрицы парных сравнений

Номер варианта	Различие в степени влияния факторов на величину комплексной оценки индивидуального труда разработчиков	Значение величины отклонения (y)
1	сильное	0,9 – 0,95
2	среднее	0,5
3	слабое	0,1 – 0,2

Поясним, что подразумевается под сильным, средним и слабым различием в степени влияния факторов на величину комплексной оценки индивидуального труда. Допустим, что на величину комплексной оценки влияют четыре фактора: f_1 (объем выполненных работ), f_2 (качество выполненных работ), f_3 (трудовая активность и инициатива), f_4 (трудовая и технологическая дисциплина).

Вариант 1.

В случае, когда по каким-либо причинам будет отмечаться снижение качества работ, выполняемых группой программистов, или при создании программного обеспечении руководство проектом считает самым важным качество конечного продукта, то фактор f_2 приобретает самое сильное влияние на величину комплексной оценки индивидуального труда.

При этом составляются, например, такие соотношения:

$$f_2 >> f_1 > f_3 = f_4 \text{ или } f_2 >> f_1 = f_3 > f_4 \text{ и другие.}$$

Поскольку различия в степени влияния факторов на комплексные оценки индивидуального труда сильные (присутствует знак >>), то в соответствии с *табл. 1* $y = 0{,}9 - 0{,}95$. Конкретное значение y устанавливают, исходя из практических соображений.

Такая же ситуация может возникнуть и в том случае, когда необходимо поднять значимость двух факторов (например, f_1 и f_2):

$$f_1 = f_2 >> f_3 > f_4 \text{ или } f_1 = f_2 >> f_3 = f_4.$$

Вариант 2.

При ранжировании факторов по степени их влияния на комплексную оценку индивидуального труда составляются следующие соотношения:

$$f_1 > f_2 > f_3 > f_4 \text{ или } f_1 > f_2 = f_3 > f_4$$

$$\text{или } f_1 = f_2 > f_3 > f_4 \text{ и др.}$$

Видим, что различия в степени влияния факторов на комплексную оценку среднее, то есть ни один из факторов резко не выделяется в его влиянии на комплексную оценку (знаки >> в соотношениях отсутствуют). При этом варианте в соответствии с *табл. 1* $y = 0{,}5$.

Вариант 3.

При ранжировании факторов по степени их влияния на величину комплексной оценки индивидуального труда составляются следующие соотношения:

$$f_1 > f_2 = f_3 = f_4 \text{ или } f_1 = f_2 = f_3 > f_4$$
$$\text{или } f_1 = f_2 > f_3 = f_4 \text{ и др.}$$

То есть различия в степени влияния факторов на комплексную оценку слабое (знаки >> в соотношениях отсутствуют, а знаки = преобладают). При этом варианте в соответствии с *табл. 1* предлагается использовать $y = 0{,}1 - 0{,}2$.

Например, для проекта разработки программного комплекса *X* руководство проектом решает использовать четыре фактора для оценки индивидуального труда программистов:

1. объем выполненной работы (f_1);
2. качество выполненной работы f_2) ;
3. активность и инициатива (f_3) ;
4. трудовая и технологическая дисциплина (f_4).

Для комплексной оценки индивидуального труда разработчиков программного обеспечения руководство проектом считает чрезвычайно важными два первых фактора, а третий фактор более важным, чем четвертый. То есть $f_1 = f_2 >> f_3 > f_4$ и в соответствии с *табл. 1* выбирается $y = 0{,}9$.

В *табл. 2* приведены результаты расчетов приоритетов (весов) факторов, причем, порядок $k = 4$. В первых пяти колонках *таблицы* приведена матрица парных сравнений факторов, причем $y = 0{,}9$. Суммируя по строкам матрицы оценки предпочтения, для каждого фактора определяется интегрированная сила первого порядка $Q_f(1)$. Следующая колонка *таблицы* содержит нормированные интегрированные силы первого порядка. Для получения большей точности оценок для каждого фактора определяются интегрированные силы второго, третьего и четвертого порядков ($Q_f(2)$, $Q_f(3)$, $Q_f(4)$), которые также нормируются ($Q_f^{отн}(2)$, $Q_f^{отн}(3)$, $Q_f^{отн}(4)$). После четвертой итерации приоритет

Таблица 2.

Результаты расчетов приоритетов (весов) факторов, влияющих на комплексную оценку индивидуального труда разработчиков программного обеспечения

i/j	f_1	f_2	f_3	f_4	$Q_f(1)$	$Q_f^{отн}(1)$	$Q_f(2)$	$Q_f^{отн}(2)$	$Q_f(3)$	$Q_f^{отн}(3)$	$Q_f(4)$	$Q_f^{отн}(4)$
f_1	1	1	1,9	1,9	5,8	0,36	20,0	0,40	58,0	0,42	160,44	0,42
f_2	1	1	1,9	1,9	5,8	0,36	20,0	0,40	58,0	0,42	160,44	0,42
f_3	0,1	0,1	1	1,9	3,1	0,19	6,7	0,14	16,0	0,11	41,71	0,11
f_4	0,1	0,1	0,1	1	1,3	0,08	2,8	0,06	7,4	0,05	20,63	0,05
итого					16	1	49,4	1	139,4	1	383,21	1

(вес) первых двух факторов составил по 42 процента, а веса третьего и четвертого факторов – соответственно 11 и 5 процентов.

Порядок, используемый в методе расстановки приоритетов *(k*), рекомендуется брать равным двум или трем, так как последующие итерации мало уточняют значение относительных приоритетов, что видно в *табл. 2.*

Необходимо обосновать используемые в *табл. 1* диапазоны значений величины отклонений (*y*). В разработанном Саати методе анализа иерархий [6] при сравнении объектов по некоторому фактору используется шкала интенсивности от 1 до 9. Оценки имеют следующий смысл: 1 – равная важность, 3 – умеренное превосходство одного над другим, 5 – существенное превосходство одного над другим, 7 – значительное превосходство одного над другим, 9 – очень сильное превосходство одного над другим, 2, 4, 6, 8 – соответствующие промежуточные значения. Если, например, при проведении попарного сравнения по мнению экспертов объект *i* умеренно превосходит объект *j*, то $b_{ij} = 3$, а $b_{ji}=1/3$.

Авторы статьи не стремились усовершенствовать метод Саати, а предлагают лишь одну из возможных модификаций этого метода. Считаем, что эксперту при сравнении объектов легче ответить «лучше (важнее, предпочтительнее)», «хуже (менее важно или менее предпочтительно)», «равноценны», чем, используя шкалу от 1 до 9, вместо оценки 4 поставить, например, оценку 6 и ошибиться.

Поэтому предлагаемые в *табл. 1* диапазоны параметра *y* и должны учитывать тот разброс оценок, который можно оценить перед началом сравнения объектов. Предложенные три варианта выбора параметра *y* дают возможность ранжировать объекты и увеличивать (уменьшать) степень их влияния на величину комплексной оценки индивидуального труда в зависимости от сложившейся ситуации и от стоящих задач. При таком подходе появляется мощный рычаг управления деятельностью коллективов разработчиков программных средств.

Представленные в *табл. 1* диапазоны параметра можно использовать в качестве рекомендуемых и в дальнейшем возможна их корректировка в зависимости от конкретных целей.

4. Процедуры ранжирования разработчиков программного обеспечения по каждому фактору, влияющему на комплексную оценку индивидуального труда

После расчета приоритетов факторов необходимо определить приоритеты членов группы разработчиков программных средств по каждому выбранному фактору. При использовании этого режима необходимо для каждого фактора *f* задать:

- ✦ матрицу парных сравнений объектов (членов группы разработчиков) по фактору $f(A_f)$;
- ✦ величину отклонения элементов матрицы парных сравнений для фактора $f(y_f)$;
- ✦ порядок, используемый в методе расстановки приоритетов (*k*).

Матрица парных сравнений объектов по фактору *f* должна заполняться руководителями проекта, возможно, с учетом мнения самих разработчиков, по алгоритму, предложенному в разделе 2.

Значение величины отклонения элементов матрицы парных сравнений объектов (разработчиков) по фактору $f(y_f)$ предлагается определять в соответствии с *табл. 3.*

Таблица 3.

Значение величины отклонения элементов матрицы парных сравнений разработчиков по некоторому фактору

Номер варианта	Различие в степени варьирования фактора у оцениваемых по нему объектов (членов группы)	Значение величины отклонения (y_f)
1	сильное	0,9 – 0,95
2	среднее	0,5
3	слабое	0,1 – 0,2

Если один (несколько) членов группы по оцениваемому фактору имеют значительно лучшие (большие) показатели, чем другие члены группы, то есть $x_1 = ... = x_i >> x_{i+1} > x_{i+2=} ... > x_n$, где x_i – значение (оценка) фактора у *i*-го члена коллектива, то используется вариант 1 и $y_f = 0,9 — 0,95$.

Если все члены группы имеют приблизительно одинаковые значения оценки по некоторому фактору, то используется третий режим и $y_f = 0,1 — 0,2$.

Во всех остальных случаях предлагается использовать второй режим, при котором $y_f = 0,5$.

Нормированная интегрированная сила объекта (разработчика) по фактору *f* порядка *k* определяется по таким же формулам, как расчет весов (приоритетов) факторов, которые были приведены выше.

Фактор «объем выполненных работ». Поскольку объем выполненных работ каждым сотрудником определить количественно достаточно сложно, представляется целесообразным осуществить попарное сравнение разработчиков по этому фактору с использованием *табл. 4*. Чем выше в этой таблице расположено качественное значение фактора «объем выполненных работ», тем более предпочтительным оно является.

В *табл. 5* приведены матрица парных сравнений членов группы из пяти человек по фактору «объем выполненных работ» и результаты расчетов приоритетов разработчиков по этому фактору для $y = 0,5$ и $k = 3$.

Видим, что самый высокий приоритет по фактору «объем выполненных работ» у Лукина (0,29), а самый низкий – у Ежова (0,12). Эта *таблица* демонстрирует также влияние количества итераций на итоговые результаты. Результаты второй и третьей итераций расчетов совпадают, соответственно, для практически целей достаточно двух – трех итераций.

Для расчета приоритетов разработчиков про-

Таблица 4.

Ранжирование качественных значений каждого фактора, влияющего на комплексную оценку индивидуального труда разработчиков программного обеспечения

Факторы	Качественные значения факторов
Объем выполненных работ	Индивидуальный план выполнен досрочно, выполнены сверхплановые работы
	Индивидуальный план выполнен досрочно
	Индивидуальный план выполнен полностью
	Индивидуальный план выполнен не полностью, но это не повлияло на выполнение плана группы в отчетном периоде
	Индивидуальный план не выполнен, что привело к невыполнению плана группы в отчетном периоде
Качество выполненных работ	Работы выполнены на высоком уровне, замечаний по качеству нет
	Выполненные работы имели отдельные замечания
	Работы неоднократно переделывались исполнителем
	Работы выполнялись на низком уровне и неоднократно передавались на доработку другим членам группы
Активность и инициатива	Разработчик принимал активное участие во всех совместных работах, помогал другим членам группы, при необходимости совмещал профессии и выполнял сверхурочную работу
	Разработчик принимал участие в некоторых совместных работах, помогал другим членам группы, при необходимости совмещал профессии и выполнял сверхурочную работу
	Разработчик редко принимал участие в совместных работах, не стремился помогать другим членам группы и к совмещению профессий
	Разработчик пассивен, не участвовал в совместных работах, не помогал другим членам группы
Трудовая и технологическая дисциплина	Замечаний по трудовой и технологической дисциплине нет
	Имеются разовые нарушения трудовой и технологической дисциплины
	Имеются неоднократные нарушения трудовой и технологической дисциплины
	Имеются прогулы, грубые нарушения технологической дисциплины

Таблица 5.

Результаты расчетов приоритетов разработчиков программных средств по фактору «объем выполненных работ»

i/j	Котов	Лукин	Селин	Ежов	Ухов	$P_{fi}(1)$	$P_{fi}^{отн}(1)$	$P_{fi}(2)$	$P_{fi}^{отн}(2)$	$P_{fi}(3)$	$P_{fi}^{отн}(3)$
Котов	1	0,5	0,5	1,5	0,5	4	0,16	17,5	0,15	80,5	0,15
Лукин	1,5	1	1,5	1,5	1,5	7	0,28	34,0	0,29	156,3	0,29
Селин	1,5	0,5	1	1,5	1	5,5	0,22	25,0	0,22	114,3	0,22
Ежов	0,5	0,5	0,5	1	0,5	3	0,12	14,0	0,12	64,8	0,12
Ухов	1,5	0,5	1	1,5	1	5,5	0,22	25,0	0,22	114,3	0,22
					итого	25	1,00	115,5	1,00	530,0	1,00

граммного обеспечения для остальных трех факторов применялся тот же алгоритм. Попарное сравнение разработчиков по каждому фактору проводилось с использованием *табл. 4.* Параметры расчетов приведены в *табл. 6.*

Таблица 6.

Параметры расчетов, которые были использованы при расчете приоритетов разработчиков по трем факторам

Факторы оценки индивидуального труда разработчиков	Величина отклонения элементов матрицы парных сравнений	Порядок расчетов (количество итераций)
Качество выполненных работ	0,9	3
Активность и инициатива	0,5	3
Трудовая и технологическая дисциплина	0,2	3

5. Расчет комплексных оценок индивидуального труда разработчиков программного обеспечения

После расчета приоритетов факторов, влияющих на величину комплексной оценки индивидуального труда, и определения приоритетов оцениваемых объектов по каждому фактору предусмотрен расчет комплексной оценки индивидуального труда каждого сотрудника по формуле

$$P_i^{комп} = \sum_{f=1}^{l} P_{fi}^{отн}(k) \times Q_f^{отн}(k),$$

где $P_i^{комп}$ – комплексная оценка индивидуального труда разработчика i;

$P_{fi}^{отн}(k)$ – приоритет разработчика i порядка k по фактору f;

$Q_f^{отн}(k)$ – приоритет фактора f порядка k;

l – количество факторов.

Распределение премии (стимулирующих надбавок) группе разработчиков на основе комплексных оценок индивидуального труда осуществляется по следующей формуле:

$$d_i = DP_i^{комп}$$

где d_i – сумма премии (надбавки), причитающаяся i-му члену группы ;

D – общая сумма премии (надбавок), подлежащая распределению между членами группы.

В *табл. 7* представлены результаты расчетов комплексных оценок индивидуального труда для группы разработчиков из пяти человек и распределения премии (100000 рублей) между ними в зависимости от этих оценок. Причем, в третьей строке таблицы даны значения приоритетов факторов, которые были рассчитаны в *табл. 2.* Для каждого разработчика приведены приоритеты по каждому из четырех выбранных факторов, которые были рассчитаны в *табл. 5* и в соответствии с *табл. 6.*

Таблица 7.

Результаты расчетов комплексных оценок индивидуального труда разработчиков программных средств и распределение премиального фонда

Разработчики программного обеспечения	факторы				Комплексная оценка индивидуального труда	Распределение премии
	объем	качество	инициатива	дисциплина		
	0,42	0,42	0,11	0,05		
Котов	0,15	0,14	0,17	0,22	0,15	15236
Лукин	0,29	0,37	0,24	0,22	0,32	31720
Селин	0,22	0,07	0,29	0,22	0,16	16473
Ежов	0,12	0,04	0,17	0,17	0,10	9596
Ухов	0,22	0,37	0,12	0,18	0,27	26974
Итого	1,00	1,00	1,00	1,00	1,00	100000

Табл. 7 наглядно отражает ранжирование членов группы по каждому фактору, приоритеты каждого фактора и делает «прозрачным» алгоритм распределения премии в зависимости от комплексной оценки индивидуального труда разработчиков.

Эту *таблицу* можно использовать и для других целей: формирование групп для новых проектов, повышение окладов, необходимость проведения тренингов персонала не только для улучшения профессиональных, но и личностных качеств. В зависимости от целей оценки персонала можно вводить дополнительные факторы оценки разработчиков: профессиональные, умственные и физические способности, личностные качества (стрессоустойчивость, внимательность, выносливость, старательность, неконфликтность) и другие.

Использование метода расстановки приоритетов позволяет получать более обоснованные значения приоритетов оцениваемых объектов (разработчиков программных средств), чем при использовании традиционного метода балльных оценок, которому присущ субъективизм. В результате попарного сравнения результатов деятельности сотрудников по каждому фактору определяется не количественная оценка, а проставляются лишь знаки предпочтения, выраженные в качественной форме: больше (лучше), равно, меньше (хуже). Использование особой матричной записи и итеративной процедуры расчета позволяет качественные показатели индивидуального труда перевести в количественные. Важно отметить, что прозрачность оценок и несложный алгоритм расчетов позволят не только обоснованно оценивать персонал, но и дадут возможность каждому члену группы разработчиков программных средств принимать активное участие в оценке своего труда.

Преимущество предлагаемого подхода еще и в том, что выбор факторов, периодичность расчетов и регламент формирования матриц попарного сравнения может изменяться в зависимости от задач, стоящих перед группой разработчиков или перед организацией в целом.

6. Заключение

Таким образом, авторами получены следующие результаты:

1. Предложен новый подход к комплексной оценке индивидуального труда разработчиков программного обеспечения.

2. Разработан алгоритм метода расстановки приоритетов, который позволяет качественные показатели индивидуального труда перевести в количественные.

3. Выявлены факторы, оказывающие влияние на комплексную оценку индивидуального труда разработчиков программного обеспечения, и предложен алгоритм расчета значимости факторов в зависимости от задач, которые стоят перед коллективом разработчиков.

4. Предложены процедуры ранжирования разработчиков по каждому фактору, влияющему на комплексную оценку индивидуального труда, с учетом степени варьирования этого фактора у оцениваемых объектов.

5. При апробации алгоритма представлены расчеты комплексных оценок индивидуального труда разработчиков программного обеспечения.

Практическая значимость работы заключается в том, что предложенные методика и алгоритм комплексной оценки индивидуального труда могут быть использованы при разработке регламентов управленческого учета с целью совершенствования управления персоналом с учетом стратегических и тактических задач организации. ■

Литература

1. Heemstra F.J. Software cost estimation // Information and Software Technology. – 1992. – Vol. 34, Issue 10. – P. 627-639.
2. Samadzadeh M.H., Nandakumar K. A study of software metrics // Journal of Systems and Software. – 1991. – Vol. 16, Issue 3. – P. 229-234.
3. Kitchenham B. What's up with software metrics? – A preliminary mapping study // Journal of Systems and Software. – 2010. – Vol. 83, Issue 1. – P. 37-51.
4. Human factors in software engineering: A review of the literature // Journal of Systems and Software. – 1985. – Vol. 5, Issue 1. – P. 3-14.
5. Ершов А.П. О человеческом и эстетическом факторах в программировании // В сб.: А.П.Ершов. Избранные труды. – Новосибирск: Наука, 1994.
6. Саати Т. Принятие решений – метод анализа иерархий / Пер. с англ. - М.: Радио и Связь, 1993.