



БИБЛИОТЕКА ПРОГРАММИСТА



С. Макконнелл

Сколько стоит программный проект

Книга поможет вам:

- ⌚ составить оценку проекта прямо сейчас
- ⌚ повысить личную квалификацию в области оценок
- ⌚ улучшить свою репутацию в организации
- ⌚ разобраться в области оценки программных проектов в целом

Microsoft
Press

РУССКАЯ РЕДАКЦИЯ

ПИТЕР

Лист 16. Формирование отчета оценки на немецком языке	125
Лист 17. Стандартизация отчетов оценки	128
Часть III. КонкRETные документы оценки	
Лист 18. Оценка финансовых активов	200
Лист 19. Оценка финансовых активов других секторов	205
Лист 20. Оценка финансовых активов других секторов	210
Лист 21. Оценка недвижимого имущества	215
Лист 22. Оценка земельных участков	225
Лист 23. Оценка недропользования в недрах	235
Лист 24. Оценка базы доходов Акционерной компании	245
Лист 25. Оценка недропользования других секторов	255
Лист 26. Оценка недропользования в недрах	265
Лист 27. Оценка недропользования в недрах	275
Лист 28. Оценка недропользования в недрах	285
Лист 29. Оценка недропользования в недрах	295
Лист 30. Оценка недропользования в недрах	305
Введение	13
Благодарности	17
Об авторе	19
От издательства	20

Чего нет в книге

С чего начать

Часть I. Основные принципы оценки

Глава 1. Что такое «оценка»?	22
Глава 2. Какой из вас оценщик?	34
Глава 3. О точности оценки	39
Глава 4. Откуда берутся ошибки оценки?	50
Глава 5. Факторы, влияющие на оценку	71

1.1. Оценка, цели и обязательства

1.2. Связь между оценками и планами

Часть II. Базовые методы оценки

Глава 6. Знакомство с методами оценки	90
Глава 7. Подсчет, вычисление, экспертная оценка	95
Глава 8. Калибровка и исторические данные	102
Глава 9. Индивидуальные экспертные оценки	114
Глава 10. Декомпозиция и сводные оценки	122
Глава 11. Оценка по аналогии	136
Глава 12. Опосредованные оценки	143
Глава 13. Экспертные оценки в группах	156
Глава 14. Оценочные программы	163
Глава 15. Использование альтернативных оценок	170

6 Краткое содержание

Глава 16. Формирование оценок в правильно оцениваемых проектах . . . 175
Глава 17. Стандартизированные процедуры оценки 185

Часть III. Конкретные проблемы оценки

Глава 18. Специфические проблемы при оценке размера	200
Глава 19. Специфические проблемы при оценке объема работ	211
Глава 20. Специфические проблемы при оценке сроков	224
+ Глава 21. Оценка параметров планирования	235
Глава 22. Стили представления оценок	252
Глава 23. Политика, переговоры и решение проблем	261
Приложение А. Проверка разумности оценки	272
Приложение Б. Ответы на вопросы теста из главы 2	274
Приложение В. Рекомендации по оценке программных проектов	275
Библиография	286
Алфавитный указатель	295

www.ipotsem.com

Левицкий И. А. «ОТЧЕТ ОБ АВТОМАТИЗАЦИИ ПРОЦЕССА ПРОДУКЦИИ». Ученая степень кандидата технических наук. Ученая степень кандидата технических наук.

LUB99 2. KSKON N3 B9C DFLGHTNKS
LUB99 3. Q.TOMHCLN OTHRN

20 4. Otkrytija gdebyta omonomija otrehivajus Lusba

УДК 004.4
ББК 32.973-018-02

Ляга 6. ЗБІГОМСТВО С МЕТОДІМ ОТРИХНУТИ

Лист 8. Каналы связи с производством и складами в Магнитогорске-Челябинске-Сибирь-Урал

Лягаев А. НЕДАЧНАЯ ПРОГНОЗИРОВАНИЕ АКЦИЙ И КОМПАНИЙ

LUBBS J. O. *DECOMPOSITION OF STARCH*. U.S. Pat. No. 2,435,153.

Lusaa 15. Quidcdeqashhrie orghin

Либес Т.А. Оценочные процедуры в биологии. Учебно-методическое пособие. СПб.: Издательство НИУ ВШЭ, 2007.

LJ989 12. NCUNP03089HNS 91576981NBNPQ OLEHOK

Введение	13
Глава 1. Что такое «оценка»?	22
1.1. Оценки, цели и обязательства	22
1.2. Связь между оценками и планами	23
1.3. Общение по вопросам оценок, целей и обязательств	24
1.4. Оценка как вероятностное утверждение	25
1.5. Распространенные определения «хороших» оценок	28
1.6. Оценки и управление проектом	30
1.7. Настоящее значение оценки	32
1.8. Рабочее определение «хорошей оценки»	33
Дополнительные ресурсы	33
Глава 2. Какой из вас оценщик?	34
2.1. Простой тест по оценке	34
2.2. Обсуждение результатов теста	35
Насколько достоверна «90-процентная достоверность»?	35
Как выбирать ширину диапазона?	37
Откуда возникает стремление к сужению диапазонов?	37
Насколько показателен тест для реальной оценки программных проектов?	38

Содержание

8 Содержание

Глава 3. О точности оценки	139
3.1. Что лучше — переоценка или недооценка?	39
Аргументы против переоценки	40
Аргументы против недооценки	40
Сравнивая аргументы	42
3.2. Достижения в области оценки программных проектов	42
Насколько велики нарушения сроков?	44
По данным одной компании	44
Общесистемная проблема программной отрасли	45
3.3. Преимущества точных оценок	45
3.4. Значение предсказуемости по сравнению с другими желательными атрибутами проекта	47
3.5. Недостатки распространенных методов оценки	49
Дополнительные ресурсы	49
Глава 4. Откуда берутся ошибки оценки?	50
4.1. Источники неопределенности в оценках	51
4.2. Конус неопределенности	52
Можно ли победить конус неопределенности?	54
Конус не сужается автоматически	55
Учет конуса неопределенности в оценках программных проектов	56
Связь между конусом неопределенности и обязательствами	57
Конус неопределенности и итеративная разработка	57
4.3. Хаотические процессы разработки	58
4.4. Нестабильные требования	59
Оценка роста требований	60
Пропущенные операции	61
Необоснованный оптимизм	63
Субъективность и пристрастие	64
Импровизированные оценки	66
Излишняя четкость	68
Другие источники ошибки	69
Дополнительные ресурсы	69
Глава 5. Факторы, влияющие на оценку	71
5.1. Размер проекта	71
Почему в книге размер задается в количестве строк программного кода?	72
Издержки масштаба	73
Когда можно смело игнорировать издержки масштаба	76
Важность издержек масштаба при оценке программных проектов	77
5.2. Тип программы	78
5.3. Факторы персонала	79
5.4. Язык программирования	80
5.5. Другие факторы	81
5.6. Снова об издержках масштаба	86
Дополнительные источники	88
Часть II. Базовые методы оценки	
Глава 6. Знакомство с методами оценки	90
6.1. Выбор методов оценки	90
Что именно оценивается	90
Размер проекта	91
Стиль разработки	91

Стадия разработки	92
Возможная точность	93
6.2. Таблицы применимости методов	93
Глава 7. Подсчет, вычисление, экспертная оценка	95
7.1. Сначала подсчет	96
7.2. Что считать	96
7.3. Используйте вычисления для преобразования счетных показателей в оценки	98
7.4. Используйте экспертное суждение только в крайнем случае	100
Дополнительные ресурсы	101
Глава 8. Калибровка и исторические данные	102
8.1. Улучшение точности и другие преимущества исторических данных	103
Учет организационных факторов	103
Предотвращение субъективизма и необоснованного оптимизма	104
Снижение влияния политических факторов при оценке	105
8.2. Сбор данных	106
Проблемы определения размера	106
Проблемы измерения объема работ	107
Проблемы измерения календарного времени	108
Проблемы измерения дефектов	108
Другие проблемы сбора данных	109
8.3. Способ калибровки	109
8.4. Уточнение оценок по данным проекта	110
8.5. Калибровка средними отраслевыми данными	111
8.6. Итоги	113
Дополнительные ресурсы	113
Глава 9. Индивидуальные экспертные оценки	114
9.1. Структурированные экспертные суждения	115
Кто создает оценки?	115
Гранулярность	115
Диапазоны	116
Формулы	117
Контрольные списки	119
9.2. Сравнение оценок с фактическими значениями	119
Дополнительные ресурсы	121
Глава 10. Декомпозиция и сводные оценки	122
10.1. Вычисление ожидаемого случая	122
10.2. Закон больших чисел	124
10.3. Насколько мелкими должны быть оцениваемые фрагменты?	124
10.4. Декомпозиция с использованием WBS	126
10.5. Риск суммирования оценок для лучших и худших случаев	127
Осторожно: впереди математика!	128
Где произошел сбой?	128
10.6. Создание осмысленных общих оценок для лучшего и худшего случаев	129
Вычисление сводных оценок лучшего и худшего случаев при малом количестве задач (упрощенная формула стандартного отклонения)	130
Вычисление сводных оценок лучшего и худшего случаев при большом количестве задач (сложная формула стандартного отклонения)	131
Создание сводных оценок для лучшего и худшего случаев	133
Предупреждения по поводу достоверных оценок	134
Дополнительные ресурсы	135

10 Содержание

Глава 11. Оценка по аналогии	136
11.1. Основные принципы оценки по аналогиям	137
Шаг 1. Получение подробных данных об итоговом размере, объеме работ и затратах для предыдущего аналогичного проекта	137
Шаг 2. Сравнение размера нового проекта с аналогичным прошлым проектом	138
Шаг 3. Построение оценки размера нового проекта в процентах от размера старого проекта	139
Шаг 4. Создание оценки объема работ на основе размера нового проекта, полученного сравнением с размером предыдущего проекта	140
Шаг 5. Проверка согласованности предположений между старым и новым проектами	140
11.2. Замечания по поводу неопределенности в оценке Triad	141
Неопределенность оценки, планы и обязательства	142
Глава 12. Опосредованные оценки	143
12.1. Нечеткая логика	144
Как получить средние размеры	144
Классификация новой функциональности	145
Как не следует использовать нечеткую логику	145
Расширения нечеткой логики	146
12.2. Стандартные компоненты	146
Использование стандартных компонентов с процентилями	148
Ограничения метода стандартных компонентов	149
12.3. Абстрактные рейтинги	149
О шкале абстрактных рейтингов	151
12.4. Метод футбольки	152
12.5. Другие применения опосредованных методов	155
Дополнительные ресурсы	155
Глава 13. Экспертные оценки в группах	156
13.1. Групповое обсуждение	156
13.2. Широколосный Дельфийский метод	158
Эффективность широколосного Дельфийского метода	159
«Истина где-то рядом»	160
Когда применять широколосный Дельфийский метод	161
Дополнительные ресурсы	162
Глава 14. Оценочные программы	163
14.1. Для чего необходимы оценочные программы	163
14.2. Данные, необходимые для калибровки программ	168
14.3. Ограничения оценочных программ	168
14.4. Основные оценочные программы	168
Дополнительные ресурсы	169
Глава 15. Использование альтернативных оценок	170
Дополнительные ресурсы	174
Глава 16. Формирование оценок в правильно оцениваемых проектах	175
16.1. Формирование оценки в неправильно оцениваемом проекте	175
16.2. Формирование оценки в правильно оцениваемом проекте	176
16.3. Хронологическая последовательность формирования оценки для всего проекта	177
Формирование оценки в крупных проектах	179
Формирование оценки в мелких проектах	179
16.4. Уточнение оценок	179

16.5. Представление измененной оценки другим ключевым сторонам проекта	180
Когда представлять повторные оценки	181
Что делать, если начальство запрещает переоценку?	182
16.6. Критерии правильной оценки проекта	183

Глава 17. Стандартизированные процедуры оценки 185

17.1. Типичные элементы стандартизированной процедуры	186
17.2. Адаптация оценки для процесса поэтапного контроля	186
17.3. Пример стандартизированной процедуры оценки для последовательных проектов .	189
17.4. Пример стандартизированной процедуры оценки для итеративных проектов .	192
17.5. Пример стандартизированной процедуры оценки от прогрессивной организации .	194
17.6. Улучшение стандартизированной процедуры	197
Дополнительные ресурсы	197

Часть III. Конкретные проблемы оценки

Глава 18. Специфические проблемы при оценке размера 200

18.1. Проблемы при оценке размера	201
Роль строк кода в оценке размеров	201
18.2. Функциональные пункты	203
Преобразование функциональных пунктов в строки кода	205
18.3. Упрощенные методы вычисления функциональных пунктов	206
Голландский метод	207
Элементы GUI	207
18.4. Сводка методов оценки размера	208
Дополнительные ресурсы	209

Глава 19. Специфические проблемы при оценке объема работ 211

19.1. Факторы, влияющие на объем работ	211
19.2. Вычисление объема работ по размеру	213
Вычисление оценки объема работ посредством неформального сравнения с прошлыми проектами	213
Какой объем работ включается в эту оценку?	214
19.3. Вычисление оценки объема работ научными методами	214
19.4. Среднеотраслевые графики объема работ	215
19.5. Метод ISBSG	220
19.6. Сравнение оценок объема работ	222
Дополнительные ресурсы	223

Глава 20. Специфические проблемы при оценке сроков 224

20.1. Базовая формула для вычисления срока	224
20.2. Вычисление срока посредством неформальных сравнений с прошлыми проектами	226
20.3. Метод оценки первого порядка	227
20.4. Вычисление оценки срока научными методами	228
20.5. Сжатие графика и размер групп	228
20.6. Соотношения между сроком и объемом работ	231
Сокращение срока и размер группы	231
20.7. Оценка срока и ограничения численности группы	233
20.8. Сравнение результатов, полученных разными методами	234
Дополнительные ресурсы	235

Глава 21. Оценка параметров планирования 236

21.1. Оценка операционной структуры проекта	236
Оценка объема работ по разным техническим операциям	237

12 Содержание

Глава 1. Оценка объема работ	237
Оценка объема работ по требованиям	237
Оценка объема работ по управлению	238
Оценка общего объема работ	238
Поправки для типов проектов	239
Пример распределения объема работы	239
Соотношения между численностью разработчиков и тестеров	240
21.2. Оценка срока для разных операций	241
21.3. Преобразование оценки в планируемый объем работы	242
21.4. Оценка стоимости	243
Сверхурочные работы	243
Выбор типа затрат	244
Другие прямые затраты	244
21.5. Оценка дефектообразования и исправления	244
Оценка работы по исправлению дефектов	245
Пример оценки эффективности исправления дефектов	246
21.6. Оценка риска и резервные буферы	248
21.7. Другие эмпирические правила	249
Дополнительные ресурсы	250
Глава 22. Стили представления оценок	252
22.1. Представление предположений, заложенных в оценку	252
22.2. Выражение неопределенности	254
Квалификаторы «плюс/минус»	254
Квантификация рисков	254
Коэффициенты достоверности	255
Сценарные оценки	256
Грубая оценка дат и периодов времени	258
22.3. Диапазоны (любого типа)	258
Полезность диапазонного представления оценок	259
Диапазоны и обязательства	260
Дополнительные ресурсы	260
Глава 23. Политика, переговоры и решение проблем	261
23.1. Особенности руководства	261
23.2. Влияние политических факторов на оценки	262
Внешние ограничения	263
Бюджет и даты	263
Переговоры по поводу оценок и обязательств	263
Что делать, если ваша оценка не принята	264
Ответственность технического персонала за обучение	264
23.3. Решение проблем и принципиальные переговоры	265
Переговоры как совместное решение проблем	266
Дополнительные ресурсы	271
Приложение А. Проверка разумности оценки	272
Результат	273
Приложение Б. Ответы на вопросы теста из главы 2	274
Приложение В. Рекомендации по оценке программных проектов	275
Библиография	286
Алфавитный указатель	294

-как и сама оценка, дает слишком расплывчатой и эвристичной проблемой для анализа. Важно избежать спонтанной оценки, она приводит к ошибкам; это означает, что оценка должна быть строгой и точной, а не спонтанной и неустановленной. Ключевое значение имеет то, что оценка должна быть строгой и точной, а не спонтанной и неустановленной. Ключевое значение имеет то, что оценка должна быть строгой и точной, а не спонтанной и неустановленной.

Введение

Основные преимущества книги

Ориентируясь на неформальное искусство «оценки», книга освещает ряд важных вопросов:

- * Что такое «оценка»?

Похоже, три самых тяжелых года в обучении специалистов по оценке приходятся на школьный курс арифметики. Кто-то изучает математику в школе, кто-то изучает ее в университете, кто-то изучает ее в институте. Но даже самые лучшие студенты могут испытывать затруднения в изучении математики. Это неизбежно, потому что математика — это язык, который требует точности и ясности. Оценка программных проектов не так уж сложна. Эксперты ведут исследования и пишут на эту тему уже около 40 лет; за это время был разработан целый ряд методов, обеспечивающих точную оценку программных проектов. Создание точной оценки — дело простое и прямолинейное... когда вы понимаете, как ее создать. Но далеко не все методы оценки очевидны на интуитивном уровне, и даже очень умный человек не сможет найти все полезные методы самостоятельно. Если кто-то является квалифицированным программистом, это еще не делает его хорошим специалистом по оценке.

Норман Р. Огастин

Многие аспекты оценки программных проектов не видны с первого взгляда. Нередко так называемые проблемы оценки обусловлены неверным пониманием того, что же такое «оценка», или путаницей с похожими, но не идентичными концепциями. Некоторые методы оценки кажутся полезными, но не дают точных результатов. Сложные формулы порой приносят больше вреда, чем пользы, тогда как обманчиво простые методы работают на удивление хорошо.

В этой книге изложена суть четырех десятков лет исследований и еще более долгой практической работы, направленной на то, чтобы разработчики, руководители проектов и специалисты по тестированию могли эффективно работать в области оценки программных проектов. Знания в области оценки программных проектов вообще полезны, потому что факторы, влияющие на оценку, также влияют и на сам процесс разработки программного обеспечения.

Оценка программных проектов: искусство или наука?

В настоящее время исследования в области оценки программных проектов сосредоточены на совершенствовании методов оценки, чтобы передовые организации могли прогнозировать результат с точностью $\pm 5\%$ вместо $\pm 10\%$. В этих методах

задействованы интенсивные вычисления, их понимание требует хорошей математической подготовки и усердного, направленного изучения, а необходимая обработка данных выходит далеко за рамки того, что можно проделать на ручном калькуляторе. Такие методы лучше всего работают в виде специализированных коммерческих пакетов оценки программных проектов. Я буду называть их *научными методами оценки*.

С другой стороны, типичная фирма-разработчик вовсе не стремится к повышению точности с $\pm 10\%$ до $\pm 5\%$. Она пытается избежать оценок, ошибочных на 100 % и более. (На то есть много причин, которые подробно рассматриваются в главах 3 и 4.)

Мы от природы склонны верить, что сложные формулы вида

$$\text{Effort} = 2.94 * (\text{KSLOC})^{[0.91 + 0.01 * \sum_{j=1}^5 SF_j]} * \prod_{i=1}^{17} EM_i$$

всегда обеспечивают более точные результаты, чем простые формулы

$$\text{Затраты} = \text{КоличествоФакторов} \times \text{СредниеЗатратыНаФактор}$$

Однако сложные формулы не всегда лучше. Программные проекты находятся под влиянием множества факторов, противоречащих многим допущениям, задействованным в сложных формулах научной оценки. Соответствующая динамика будет рассмотрена далее в книге. Более того, многие практические работники не имеют ни свободного времени, ни желания изучать сложную математику, необходимую для освоения научной оценки.

По этой причине в книге основное внимание уделяется эмпирике, процедурам и простым формулам, высокоеффективным и одновременно понятным для практикующих профессионалов в области программного обеспечения. Такие методы не обеспечивают оценок с точностью до $\pm 5\%$, но они сократят ошибку оценки до $+25\%$ и менее; для большинства проектов этого вполне достаточно. Я буду называть совокупность этих методов *неформальной оценкой*.

Материал книги основан как на научных, так и на неформальных методах, но в целом книга ориентируется на оценку программного обеспечения как искусство.

Почему и для кого написана эта книга

Литература, посвященная оценке программных проектов, весьма обширна. Исследователи опубликовали сотни статей, и многие из них достаточно полезны. Однако у среднестатистического практика нет времени следить за десятками статей в малопонятных технических журналах. Ранее было опубликовано несколько книг, посвященных научной оценке. Такие книги содержат по 800–1000 страниц, требуют хорошей математической подготовки и пишутся в основном для профессиональных оценщиков — консультантов или специалистов, которые часто занимаются оценкой крупных проектов.

Я написал эту книгу для разработчиков, руководителей и специалистов по тестированию, для которых построение оценок является лишь одной из многих составляющих повседневной работы. Полагаю, многие практики хотели бы улучшить точность своих оценок, но не располагают свободным временем для получения докторской степени в области оценки программных проектов. Им при-

ходится решать практические вопросы: как обойти политические проблемы, связанные с оценкой; как представить оценку, чтобы она была принята начальством; и что нужно сделать, чтобы посторонние не могли изменить оценку по своему усмотрению. Если вы относитесь к этой категории, значит, книга написана для вас.

Методы, представленные в книге, применимы к разработкам для Интернета и интрасетей, разработке встроенного программного обеспечения, коммерческим системам, новым проектам, наследным системам, крупным проектам, мелким проектам — словом, к оценке практических любых видов программного обеспечения.

Основные преимущества книги

Ориентируясь на неформальное искусство оценки, книга освещает ряд важных вопросов.

- Что такое «оценка»? (Вероятно, вы думаете, что ответ на этот вопрос вам уже известен, однако этот термин часто употребляется в неточном контексте, мешающем эффективной оценке.)
- Факторы, из-за воздействия которых ваши прошлые оценки оказались менее точными, чем могли бы.
- Способы отличить хорошую оценку от плохой.
- Различные приемы, помогающие лично вам создавать хорошие оценки.
- Некоторые приемы, которые помогут другим участникам вашей группы создавать хорошие оценки.
- Пути создания хороших оценок в вашей организации. (Между методиками личными, групповыми и организационными существуют важные различия.)
- Оценочные методы, работающие для проектов на базе гибких (*agile*) технологий, и методы, ориентированные на традиционные, то есть последовательные (плановые) проекты.
- Оценочные методы, предназначенные для мелких проектов, и методы, работающие на крупных проектах.
- Искусство лавирования в бурных политических водах, часто окружающих проблему оценки программных проектов.

Помимо лучшего понимания общих принципов оценки, материал книги поможет вам лучше оценивать конкретные атрибуты программных проектов, в том числе:

- Разработка новых проектов, включая планирование, объем и стоимость работ.
- Планирование, объем и стоимость работ по наследным системам.
- Количество функций, обеспечиваемых той или иной итерацией цикла разработки.
- Объем функциональности, обеспечиваемой для всего проекта при фиксированном временном графике и размере группы.
- Доля других необходимых операций, связанных с разработкой, включая руководящую работу, постановку требований, конструирование, тестирование и исправление недостатков.

- Параметры планирования: соотношение между затратами и временем, оптимальный размер группы, размер страхового буфера, пропорции между численностью разработчиков и специалистов по тестированию и т. д.
- Параметры качества: время, необходимое для исправления дефектов, решение проблем с дефектами, оставшимися в программном продукте во время выпуска, и другие факторы.
- И вообще практически все, что вы захотите оценивать.

Во многих случаях описанные в книге методы можно немедленно применять на практике.

Многим практическим специалистам даже не потребуется выходить за рамки концепций, представленных в книге. Тем не менее понимание изложенных принципов заложит основу для последующего перехода к другим методам, базирующимся на математических расчетах.

Чего нет в книге

В книге не рассматриваются вопросы оценки очень крупных проектов (более одного миллиона строк программного кода, или более 100 человеко-лет). Очень крупные проекты должны оцениваться профессионалами, прочитавшими десятки статей в малопонятных технических журналах, изучившими книги по 800–1000 страниц, знакомыми с коммерческими оценочными программами, поднаторевшими как в научных, так и в неформальных методах оценки.

С чего начать

Выбор отправной точки зависит от того, что вы хотите получить от этой книги.

- *Если вы купили книгу, потому что вам нужно составить оценку прямо сейчас...* Начните с главы 1, а затем переходите к главам 7 и 8. После этого кратко пройдитесь по рекомендациям в главах 10–20 и найдите те методы, которые принесут вам максимальную непосредственную пользу. Кстати, приводимые в тексте рекомендации выделены и пронумерованы, и все они (118) также собраны в приложении В, «Рекомендации по оценке программных проектов».
- *Если вы хотите повысить личную квалификацию в области оценки, улучшить свою репутацию в организации или стремитесь лучше разобраться в области оценки программных проектов в целом...* Прочтите всю книгу. Если вы хотите понять общие принципы перед тем, как погружаться в детали, читайте все подряд. Если же вы предпочитаете сначала ознакомиться с деталями, а затем выводить из них общие заключения, начните с главы 1, прочтите главы 7–23, а потом вернитесь к пропущенным главам.

Бельвио, Вашингтон
Новый год 2006

Kiander), Мехмет Кебек Каннитак (Mehmet Kebek Kanıttağı), Георгий Кацхаби
 (Georgi Katschabi), Молли Джаксон (Molly Jackson), Стив Маттингли (Steve
 Mattingly), Джо Николс (Joe Nichols), Гари О'Донован (Garry O'Donovan)
 (David O'Doherty), Мэрион Нобчин (Maryanne Nobchin), Гарри Пинкотт (Harry
 Pinkett), Дениэл Рид (Deniel Reid), Гарри Соккер (Harry Sooker), Джон
 Танн (John Tann), Ген Трумэн (Gen Truman) (Andy Warhol).
 Мне хотелиось бы поблагодарить всех, кто помогал в написании книги.

Благодарности

Не перестаю поражаться тем возможностям, которые открывает Интернет для повышения качества работы. Почти все рецензенты рукописи моей первой книги жили не далее чем в пятидесяти милях от меня. С рукописью этой книги работали рецензенты из Аргентины, Австралии, Канады, Дании, Великобритании, Германии, Исландии, Нидерландов, Северной Ирландии, Японии, Шотландии, Испании и Соединенных Штатов. Эти рецензии оказали огромное положительное влияние на качество книги.

Прежде всего я благодарен людям, предоставившим свои комментарии по значительному объему материала: Фернандо Берзаль (Fernando Berzal), Стивен Блэк (Steven Black), Дэвид Э. Берджесс (David E. Burgess), Стелла М. Бернс (Stella M. Burns), Гевин Берроуз (Gavin Burrows), Дейл Кэмпбелл (Dale Campbell), Роберт А. Клинкенберд (Robert A. Clinkenbeard), Боб Коррик (Bob Corrick), Брайан Дональдсон (Brian Donaldson), Джейсон Хиллз (Jason Hills), Уильям Хорн (William Horn), Карл Дж. Кристофчик (Carl J. Krzystofczyk), Джейфри Д. Мозер (Jeffrey D. Moser), Томас Освальд (Thomas Oswald), Аллан М. Пиндер (Alan M. Pinder), Джон Прайс (Jon Price), Кэти Род (Kathy Rhode), Саймон Робби (Simon Robbie), Эдмунд Швеппе (Edmund Schweppe), Джеральд Саймон (Gerald Simon), Крейг Р. Смит (Creig R. Smith), Линда Тейлор (Linda Taylor) и Бернд Вифхьюз (Bernd Viefhues).

Также спасибо тем, кто рецензировал отдельные части книги: Лиза М. Адамс (Lisa M. Adams), Хакон Агустссон (Hakon Agustsson), Брайон Бейкер (Bryon Baker), Тина Колман (Tina Coleman), Крис Кроуфорд (Chris Crawford), Доминик Кронин (Dominic Cronin), Джерри Девилль (Jerry Deville), Конрадо Эстол (Conrado Estol), Эрик Фриман (Eric Freeman), Хидэо Фукумори (Hideo Fukumori), С. Дейл Хильдебрандт (C. Dale Hildebrandt), Барбара Хитчингс (Barbara Hitchings), Джим Холмс (Jim Holmes), Рик Хауэр (Rick Hower), Кевин Хатчинсон (Kevin Hutchinson), Фуннур Храфн Джонссон (Funnur Hrafn Jonsson), Аарон Киандер (Aaron

Kiander), Мехмет Керем Кизилтунк (Mehmet Kerem Kiziltunc), Селимир Кустудик (Selimir Kustudic), Молли Дж. Махай (Molly J. Mahai), Стив Маттингли (Steve Mattingly), Джо Николас (Joe Nicholas), Эл Ноэл (Al Noel), Дэвид О'Донохью (David O'Donoohue), Шелдон Порсина (Sheldon Porcina), Дэвид Дж. Престон (David J. Preston), Дэниел Рид (Daniel Read), Дэвид Спокейн (David Spokane), Джанко Танис (Janco Tanis), Бен Тилли (Ben Tilly) и Венди Вильгельм (Wendy Wilhelm).

Мне хотелось бы выразить особую признательность преподавателям семинаров Construx по вопросам оценки программных проектов. После многолетних дискуссий часто бывает невозможно сказать, какие идеи принадлежат мне, а какие – им. Спасибо Эрлу Биду (Earl Beed), Гретту Боэру (Gregg Boer), Мэтту Пелоquinу (Matt Peloquin), Памеле Перро (Pamela Perrott) и Стиву Токи (Steve Tockey).

Книга посвящена неформальным методам оценки, но введенные в ней упрощенные методы стали возможны благодаря исследователям, в течение десятилетий изучавших научные способы оценки. Я искренне благодарен трем гигантам в области научной оценки: Барри Бему (Barry Boehm), Кейперсу Джонсу (Capers Jones) и Лоуренсу Патнэмю (Lawrence Putnam).

Мне снова выпала привилегия работать с Девоном Масгрейвом (Devon Musgrave), руководителем проекта этой книги. Спасибо, Девон! Бека Маккей (Becka McKay), заместитель редактора, также во многих отношениях способствовала улучшению исходной рукописи. Я также благодарен остальным работникам Microsoft Press: это Патрисия Бредбери (Patricia Bradbury), Карл Дилц (Carl Diltz), Трейси Фрил (Tracey Freil), Джесси Гуд (Jessie Good), Патрисия Массерман (Patricia Masserman), Джэл Панчо (Joel Panchot) и Сэнди Резник (Sandy Resnick). Также спасибо составителю алфавитного указателя Сету Мейслину (Seth Maislin).

Остается лишь поблагодарить мою жену Эшли – лучшего (по моей оценке) спутника жизни, на которого только я мог надеяться.

Об авторе

БОЯРДЕН ТО
ПРИНЦИПЫ ОЦЕНКИ

Глава 1. Что такое «оценка»?

Стив Маккоинел работает ведущим программистом в компании Construx Software. Он является руководителем направления конструкторской информации в проекте Software Engineering Body of Knowledge (SWEOK). Стив работал над программными проектами в Microsoft, Boeing и других компаниях, расположенных в Сиэттле. Он был ведущим разработчиком Construx Estimate и SPC Estimate Professional, лауреатом премии журнала «Software Development».

Стив является автором книг «Rapid Development» (1996), «Software Project Survival Guide» (1998), «Professional Software Development» (2004) и «Code Complete, 2nd Edition» (2004). Его книги дважды завоевывали ежегодную премию журнала «Software Development» за самые выдающиеся публикации в области разработки программного обеспечения. Стив также был ведущим разработчиком программы SPC Estimate Professional, удостоенной премии за производительность в разработке программного обеспечения. В 1998 году читатели журнала «Software Development» назвали Стива одним из трех самых влиятельных людей в отрасли разработки программного обеспечения наряду с Биллом Гейтсом и Линусом Торвальдсом.

Стив получил степень бакалавра в колледже Уитмена и степень магистра в области программотехники в университете Сиэттла. Он живет в Бельвию, штат Вашингтон.

Кандер), Мехмет Керем Кизилтунук (Mehmet Kerem Kiziltunc), Селим Кустудик (Selimir Kustudic), Молли Дж. Махай (Molly J. Mahai), Стив Мэттингли (Steve Mattingly), Джо Николас (Joe Nicholas), Эл Ноэл (Al Noel), Дэни О'Донохью (David O'Donoghue), Шелдон Порсина (Sheldon Porsina), Дэвид Дж. Престон (David J. Preston), Даниэл Рид (Daniel Read), Дэвид Спокейн (David Spokane), Джанко Тилье (Janko Tihelcă), Бен Тилли (Ben Tilly) и Венди Уильямс (Wendy Williams).
Следует отметить, что в книге не приведены имена некоторых участников проекта, включая тех, кто участвовал в его ранних этапах. Важно отметить, что в книге не указаны имена тех, кто участвовал в его ранних этапах. Важно отметить, что в книге не указаны имена тех, кто участвовал в его ранних этапах.

Ваши замечания, предложения, вопросы отправляйте по адресу электронной почты comp@piter.com (издательство «Питер», компьютерная редакция).

На веб-сайте издательства <http://www.piter.com> вы найдете подробную информацию о наших книгах.

У фирм имеются веские причины для постановки целей, не зависящие от конкретных проектов. Но тот факт, что цель является желательной и даже абсолютно необходимой, еще не означает, что она достижима.

Основные принципы оценки

Основные принципы оценки



Глава 1. Что такое «оценка»?

Глава 2. Какой из вас оценщик?

Глава 3. О точности оценки

Глава 5. Факторы, влияющие на оценку

твятся много глыб вспомогательных бойцов, однозначно подтверждая этот результат. Мы намеренно (и целесообразно) избрали классическую способность кинематики (одного из основных элементов этого метода) — это, конечно, движение вправо-влево, движение вперед-назад, движение вверх-вниз. И это движение было настолько ярко выражено, что оно неизменно привлекало внимание зрителя.

Оценки формируют основу для дальнейшего анализа, хранятся в базе данных и становятся основой для отыскания определенных явлений в исходном массиве. Несмотря на то что оценки не являются первичными данными, они являются важнейшим инструментом для дальнейшего анализа. Оценки упрощают структуру исходных данных и позволяют выделить из них наиболее важные характеристики. Оценки могут быть получены различными методами, включая статистические, геометрические и другие. Оценки могут быть получены различными методами, включая статистические, геометрические и другие.

Примеры факторов, влияющих на стоимость бизнеса в Германии:

Citroen logoa, erakundearen logoa, gizarteko logoa, eta teknologikoaren logoa, hiru logotipoa ditu.

1

Что такое «оценка»?

Очень трудно составить энергичное, правдоподобное обоснование для оценки, которая не базируется на количественных методах, практически не поддерживается данными и продвигается в основном усилиями начальства.

Фред Брукс

Вероятно, вы полагаете, что вам не нужно объяснять, что такое «оценка». К концу этой главы я постараюсь убедить вас, что смысл термина «оценка» отличается от того, что в него вкладывает большинство людей. А хорошая оценка отличается от общепринятого понимания еще сильнее.

Вот определение слова «оценка» из словаря: 1. приблизительный прогноз или вычисление. 2. предварительный расчет стоимости проекта. 3. суждение, основанное на впечатлениях; личное мнение.

Насколько это похоже на то, что вам нужно сделать при оценке программного проекта? Можно ли назвать такой результат *приблизительным* или *предварительным* — иначе говоря, предполагается ли, что вы сможете изменить свою оценку позднее?

Вероятно, нет. Когда начальство требует у вас «оценки», оно часто подразумевает некое обязательство по соблюдению поставленной цели. Различия между оценками, целями и обязательствами чрезвычайно важны для понимания того, чем является оценка, чем она не является и как повысить качество ваших оценок.

1.1. Оценки, цели и обязательства

Строго говоря, словарное определение термина «оценка» верно: оценкой называется прогноз относительно того, сколько времени или денег потребуется для реализации проекта. Однако в контексте программных проектов оценка увязывается с деловыми целями, обязательствами и контролем.

Целью называется формулировка деловой задачи. Примеры целей.

- Версия 2.1 должна быть готова к демонстрации на выставке в мае.
- Мы должны получить стабильную версию продукта до наступления предпраздничных продаж.

- Затраты на следующую версию ограничиваются двумя миллионами — это максимальный бюджет, который нам удастся получить.

У фирм имеются веские причины для постановки целей, не зависящих от оценки программных проектов. Но тот факт, что цель является желательной или даже абсолютно необходимой, еще не означает, что она достижима.

Если цель может рассматриваться как описание деловой задачи, *обязательство* является обещанием предоставить некоторую функциональность на согласованном уровне качества к конкретной дате. Обязательство может совпадать с оценкой, может быть более агрессивным или более консервативным. Другими словами, не ставьте знак равенства между обязательством и оценкой; это разные понятия.

СОВЕТ № 1

Не путайте оценки, цели и обязательства.

1.2. Связь между оценками и планами

Оценка и планирование — взаимосвязанные темы, но оценка не является планированием, а планирование — оценкой. Оценка должна рассматриваться как объективный, аналитический процесс; планирование должно рассматриваться как процесс изначально субъективный, целенаправленный. В контексте оценки было бы рискованно изначально рассчитывать на получение некоторого конкретного ответа. Целью является точность прогноза, а не достижение конкретного результата. С другой стороны, под целью планирования понимается конкретный результат. Мы намеренно (и целесообразно) воздействуем на свои планы для достижения желаемого исхода. Мы планируем конкретные средства, необходимые для достижения конкретной цели.

Оценки формируют основу для планов, но планы не всегда совпадают с оценками. Если оценки радикально отличаются от целей, планы проектов должны распознать этот разрыв и заложить более высокую степень риска. Если оценка близка к цели, планы также становятся менее рискованными.

И оценка, и планирование важны, но фундаментальные различия между ними означают, что попытки их объединения обычно приводят к плохим оценкам и плохим планам. Наличие сильной цели планирования может привести к тому, что цель подменит оценку, полученную аналитическим путем; участники проекта даже могут называть цель «оценкой», придавая ей ореол объективности, которого она в действительности не заслуживает.

Примеры факторов планирования, частично зависящих от точности оценки:

- построение подробного графика;
- идентификация критического пути проекта;
- создание полной декомпозиции работ¹;
- выбор приоритетной функциональности для сдачи на промежуточных этапах;
- разбиение проекта на итерации.

¹ Work breakdown structure. — Примеч. перев.

Точные оценки повышают качество выполнения работы по каждой из этих областей (в главе 21 приводятся более подробные сведения по всем перечисленным вопросам).

1.3. Общение по вопросам оценок, целей и обязательств

Одним из последствий тесных, а иногда и сбивающих с толку связей между оценками и планированием являются недоразумения, возникающие при общении между критическими фигурами проекта. Пример типичного недоразумения.

Директор: Как вы думаете, сколько времени займет проект? Программа должна быть готова через 3 месяца, к выставке. Я не могу дать больше людей в группу, поэтому вам придется обойтись текущим персоналом. Вот список тех функций, которые нам нужны.

Руководитель проекта: Ладно, я немного посчитаю, а потом снова зайду к вам. Позднее...

Руководитель проекта: По моей оценке, проект займет 5 месяцев.

Директор: Пять месяцев?! Вы что, меня не слушаете? Я же сказал, что программа нужна нам через 3 месяца, к выставке?

Вероятно, после этого разговора руководитель проекта будет считать директора ненормальным, потому что он требует втиснуть 5-месячный объем работы в 3 месяца. Директор же сочтет, что руководитель проекта не понимает деловых реалий и не сознает, как важно, чтобы программа была готова к выставке через 3 месяца.

Обратите внимание: в этом примере директор в действительности не просит оценки, а хочет получить от руководителя проекта *план для достижения поставленной цели*. Большинство начальников не обладает техническим образованием, которое бы позволило им различать оценки, цели, обязательства и планы. Таким образом, технический руководитель обязан сам преобразовать запрос начальства в конкретные технические термины.

Вот более продуктивный сценарий такого общения.

Директор: Как вы думаете, сколько времени займет проект? Программа должна быть готова через 3 месяца, к выставке. Я не могу дать больше людей в группу, поэтому вам придется обойтись текущим персоналом. Вот список тех функций, которые нам нужны.

Руководитель проекта: Давайте убедимся в том, что я вас правильно понял. Что для нас важнее – реализовать все возможности на 100 % или иметь хоть что-нибудь готовое к выставке?

Директор: Иметь что-нибудь готовое к выставке. Ну и хотелось бы реализовать все возможности на 100 %, если получится.

Руководитель проекта: А что делать, если окажется, что программа не готова на 100 % к выставке? Приготовиться показать то, что есть, или перенести дату выхода на более поздний момент?

Директор: Мы обязательно должны что-нибудь показать на выставке. Если время будет поджимать, отправим то, что есть, даже если продукт не будет готов на 100 %.

Руководитель проекта: Хорошо. Я подготовлю план, который позволит нам реализовать как можно большую долю функциональности за следующие три месяца.

СОВЕТ № 2

Когда вас просят предоставить оценку, определите, что именно нужно спрашивающему — оценка или план достижения цели.

1.4. Оценка как вероятностное утверждение

Как известно, около трех четвертей программных проектов превышают свои оценки. Вероятность того, что любой отдельный проект завершится к поставленному сроку и без нарушения бюджета, отлична от 100 %. Но стоит нам признать, что вероятность своевременного завершения отлична от 100 %, возникает очевидный вопрос: «Если не 100 %, то сколько?» Это один из центральных вопросов в области оценки программных проектов.

Обычно оценки программных проектов представляются в виде обычных чисел, например: «Этот проект займет 14 недель». Подобные упрощенные точечные оценки бессмысленны, потому что они не включают никакой информации о вероятности, связанной с точечной оценкой. Подразумевается ситуация, представленная на рис. 1.1, — возможен единственный исход, которым является заданная точка.

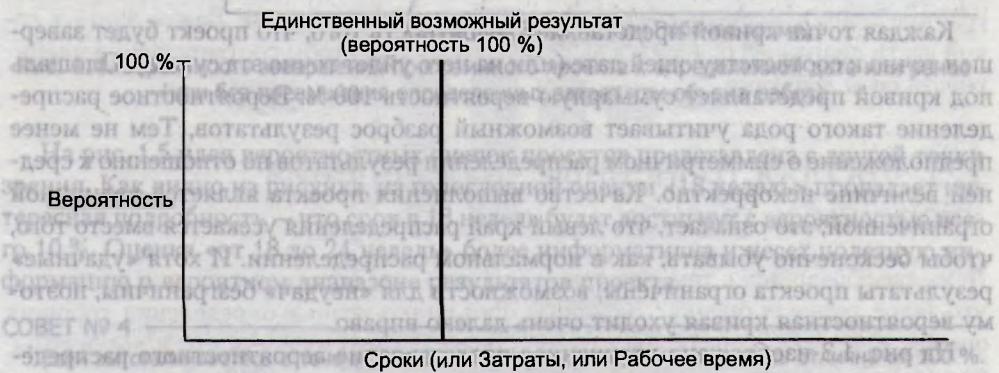


Рис. 1.1. Точечная оценка подразумевает стопроцентную вероятность совпадения фактического исхода с запланированной оценкой. В жизни так не бывает

Точечная оценка на поверку обычно оказывается целью, замаскированной под оценку. Иногда она возникает в результате удаления осмысленной вероятностной информации из более сложной и содержательной оценки.

26 Глава 1 • Что такое «оценка»?

СОВЕТ № 3

Сталкиваясь с точечной «оценкой», спросите себя, действительно ли это число является оценкой или на самом деле перед вами цель.

Точные оценки учитывают, что программные проекты подвергаются влиянию множества факторов неопределенности. В совокупности эти факторы неопределенности означают, что результат проекта подчиняется вероятностному распределению — одни результаты более вероятны, другие менее вероятны, а наиболее вероятная группа результатов сосредоточена где-то в середине распределения. Можно было бы предположить, что распределение результатов проекта будет иметь вид кривой нормального распределения (рис. 1.2).



Рис. 1.2. Часто предполагается, что результаты программного проекта распределяются по нормальному закону. Такое предположение неверно, потому что эффективность выполнения любого заданного объема работы группой, работающей над проектом,

является ограниченной величиной. Обратите внимание, что если руководитель проекта, имея точечную оценку, а хочет получить от руководителя группы точку для доставления поставленных задач, то он не может ее получить. Каждая точка кривой представляет вероятность того, что проект будет завершен точно к соответствующей дате (или на него уйдет точно эта сумма). Площадь под кривой представляет суммарную вероятность 100 %. Вероятностное распределение такого рода учитывает возможный разброс результатов. Тем не менее предположение о симметричном распределении результатов по отношению к средней величине некорректно. Качество выполнения проекта является величиной ограниченной; это означает, что левый край распределения усекается вместо того, чтобы бесконечно убывать, как в нормальном распределении. И хотя «удачные» результаты проекта ограничены, возможности для «неудач» безграничны, поэтому вероятностная кривая уходит очень далеко вправо.

На рис. 1.3 изображено уточненное представление вероятностного распределения результатов программного проекта.

Вертикальная пунктирная линия изображает «номинальный» результат, или результат 50/50 — существует 50%-я вероятность того, что проект завершится лучше, и 50%-я вероятность того, что он завершится хуже. В статистике такой результат называется *медианой*.

На рис. 1.4 изображен другой способ представления подобных вероятностных распределений. Если график на рис. 1.3 показывает вероятность завершения

проекта к конкретной дате, на рис. 1.5 показана вероятность завершения к конкретной дате или ранее.



Рис. 1.3. Уточненное представление возможных результатов программного проекта.

Возможности удачного выполнения проекта ограничены, но количество потенциальных проблем бесконечно

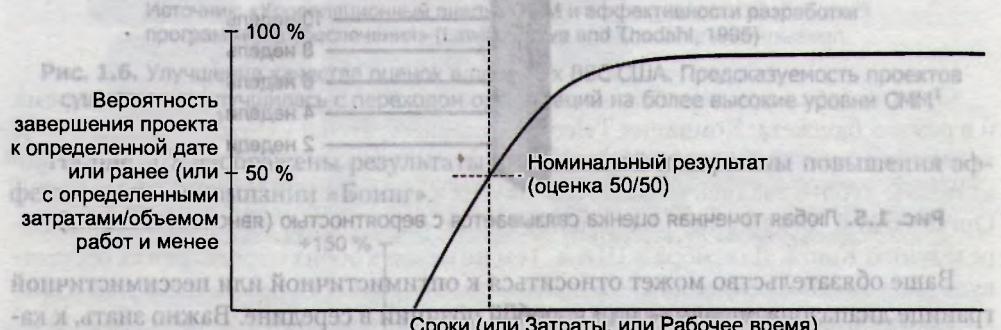


Рис. 1.4. Вероятность завершения программного проекта к определенной дате или ранее (или без превышения определенных затрат или объема работ)

На рис. 1.5 идея вероятностных оценок проектов представлена с другой точки зрения. Как видно из рисунка, из голосовой оценки «18 недель» пропадает интересная подробность — что срок в 18 недель будет достигнут с вероятностью всего 10 %. Оценка «от 18 до 24 недель» более информативна и несет полезную информацию о вероятном диапазоне результатов проекта.

СОВЕТ № 4

Если вы сталкиваетесь с точечной оценкой, скорее всего, ее вероятность отлична от 100 %. Спросите себя, какова вероятность получения этого числа.

Существуют различные способы выражения вероятностей, ассоциируемых с оценками. Можно использовать процентное выражение: «Мы на 90 % уверены в соблюдении 24-недельного срока». Можно описывать оценку в виде диапазона лучший/худший случай, подразумевающем вероятность: «В лучшем случае проект будет завершен за 18 недель, а в худшем — за 24 недели». Наконец, можно

просто сформулировать предполагаемый результат в виде диапазона вместо точечной оценки: «По нашей оценке, на проект уйдет от 18 до 24 недель». Принципиально здесь то, что любая оценка включает вероятность (явно или косвенно). Явное указание вероятности является одним из признаков хорошей оценки.



Рис. 1.5. Любая точечная оценка связывается с вероятностью (явно или косвенно)

Ваше обязательство может относиться к оптимистичной или пессимистичной границе диапазона оценки — или к любой позиции в середине. Важно знать, к какой части диапазона относится ваше обязательство, чтобы соответственно планировать свои действия.

1.5. Распространенные определения «хороших» оценок

Даже разобравшись с тем, что такое «оценка», мы остаемся с другим вопросом: что такое *хорошая* оценка? Эксперты предлагают разные определения хороших оценок. Кейперс Джонс (Capers Jones) утверждал, что точность $\pm 10\%$ достижима, но только в проектах с высокой степенью контроля (Jones 1998). Хаотичные проекты слишком непредсказуемы для достижения такой степени точности.

В 1986 году профессоры Конт, Дансмор и Шен предположили, что хорошая оценка должна в 75 % случаев отличаться от фактического результата не более чем на 25 % (Conte, Dunsmore, Shen 1986). Этот стандарт до настоящего времени является самым распространенным для выражения точности оценок (Stutzke 2005).

Различные компании сообщали о результатах, близких к критерию точности, предложенному Контом, Дансмором и Шеном. На рис. 1.6 показаны оценки и фактические результаты для ряда проектов BBC США.



Источник: «Корреляционный анализ CMM и эффективности разработки программного обеспечения» (Lawlis, Flowe and Thodahl, 1995).

Рис. 1.6. Улучшение качества оценок в проектах BBC США. Предсказуемость проектов существенно улучшилась с переходом организаций на более высокие уровни CMM¹

На рис. 1.7 изображены результаты аналогичной программы повышения эффективности в компании «Боинг».



Рис. 1.7. Повышение качества оценок в компании «Боинг». Как и в случае с BBC США, предсказуемость проектов радикально улучшается на более высоких уровнях CMM

Последний аналогичный пример на рис. 1.8 был получен при улучшении результатов оценки в Schlumberger.

¹ СММ (Capability Maturity Model) — система, определенная Институтом по разработке программного обеспечения (Software Engineering Institute) для оценки эффективности организаций, работающих в области разработки программного обеспечения.

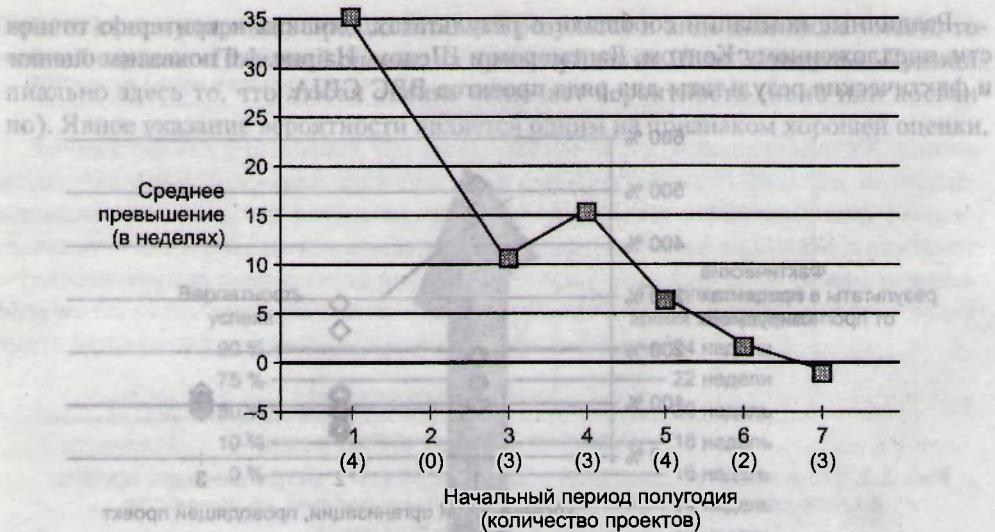


Рис. 1.8. Schlumberger повысила точность своих оценок от среднего превышения в 35 недель до среднего отставания в одну неделю

Одна из моих компаний-клиентов завершает 97 % своих проектов вовремя и в рамках бюджета. Компания Telcordia сообщает, что ей удается завершать 98 % проектов в срок и без нарушения бюджета (Pitterman 2000). Аналогичные результаты публиковались множеством других компаний (Pitnam and Myers 2003). Организации создают хорошие оценки как по определению Джонса, так и по определению Конта, Дансмора и Шена. Тем не менее в обоих определениях отсутствует одна важная концепция — а именно то, что точность результатов оценки не достигается за счет одних лишь методик оценки. Она также должна поддерживаться эффективным управлением проектом.

1.6. Оценки и управление проектом

Иногда при обсуждении оценки программных проектов оценка рассматривается исключительно как прогноз. Все выглядит так, словно оценка производится беспристрастным оракулом, находящимся где-то в открытом космосе, никак не связанным с планированием проекта и назначениями приоритетов.

В действительности оценка программных проектов не является столь отвлеченным делом. Если кому-то захочется увидеть пример принципа неопределенности Гейзенберга в области программного обеспечения — достаточно взглянуть на оценку проектов. (Принцип неопределенности Гейзенберга состоит в том, что сам факт наблюдения за явлением приводит к его изменению, и наблюдатель никогда не может быть полностью уверен в том, как повел бы себя объект при отсутствии наблюдения.) Сначала мы формируем оценку, а затем на базе этой оценки берем на себя обязательство обеспечить определенную функциональность и качество к конкретной дате; после этого мы *управляем* проектом для достижения поставленной

цели. Типичные операции по управлению проектом включают удаление некритических требований, переопределение требований, замену менее опытного персонала более опытным и т. д. Схема управления проектом представлена на рис. 1.9.

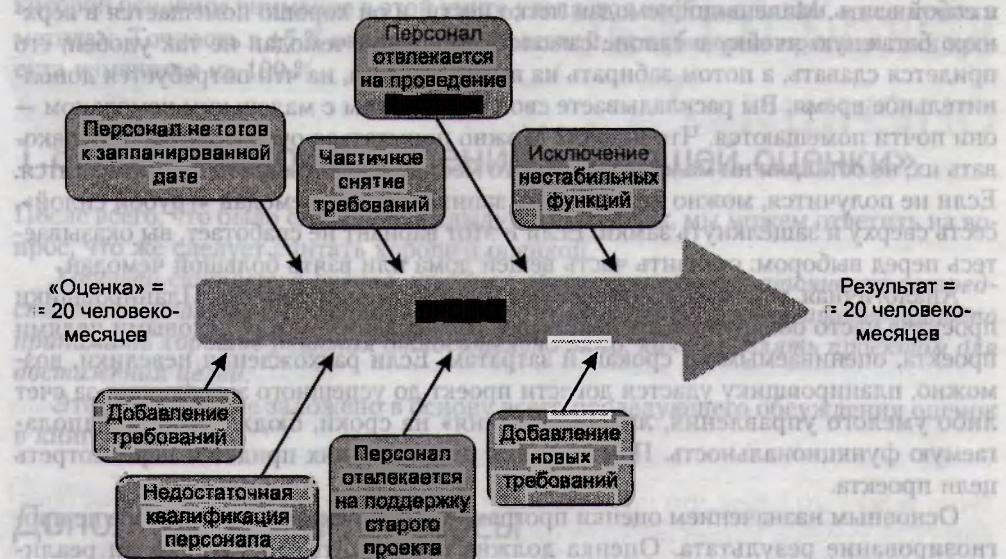


Рис. 1.9. От отправной точки до завершения проект претерпевает значительные изменения. Обычно эти изменения оказываются настолько значительными, что готовый проект отличается от того, который был заложен в основу оценки. Тем не менее, если результат сведен с прогнозом, можно сказать, что проект соответствует своей оценке

Кроме управляющих операций, проекты часто находятся под воздействием непредвиденных внешних событий. Иногда группе разработчиков приходится создавать промежуточную версию продукта для передачи важному клиенту; персоналу приходится отвлекаться на поддержку старого проекта и т. д.

События, происходящие во время работы над проектом, почти всегда берут верх над предположениями, которые использовались при первоначальной оценке проекта. Изменяется предполагаемая функциональность, изменяются предположения по поводу доступного персонала, изменяются приоритеты. В итоге становится невозможно вынести четкое аналитическое суждение относительно точности оценки проекта, потому что программный проект, полученный в конечном итоге, не идентичен изначально запланированному проекту.

На практике, если проект реализует всю предполагаемую функциональность, более или менее ограничивается запланированными ресурсами и завершается более или менее к назначенному сроку, можно сказать, что «оценка проекта оказалась верной», несмотря на все аналитические неточности, кроющиеся в этой формулировке.

Таким образом, критерий «хорошей» оценки должен базироваться не на качестве прогнозирования, а на способности оценки обеспечить успех проекта. Так мы вплотную подошли к следующей теме — настоящему значению оценки.

1.7. Настоящее значение оценки

Предположим, вы собираетесь отправиться в поездку и решаете, какой чемодан с собой взять. Маленький чемодан легко нести, и он хорошо помещается в верхнюю багажную ячейку в салоне самолета. Большой чемодан не так удобен; его придется сдавать, а потом забирать на выдаче багажа, на что потребуется дополнительное время. Вы раскладываете свои вещи рядом с маленьким чемоданом — они почти помещаются. Что делать? Можно попытаться очень тщательно упаковать их, не оставляя ни малейшего пустого места, и надеяться, что они поместятся. Если не получится, можно попробовать запихнуть их в чемодан «грубой силой», сесть сверху и защелкнуть замки. Если и этот вариант не сработает, вы оказываетесь перед выбором: оставить часть вещей дома или взять большой чемодан.

Аналогичная дилемма возникает и в программных проектах. Планировщики проектов часто обнаруживают некоторые расхождения между деловыми целями проекта, оцениваемым по срокам и затратам. Если расхождения невелики, возможно, планировщику удастся довести проект до успешного завершения за счет либо умелого управления, либо «давления» на сроки, бюджет или предполагаемую функциональность. При больших расхождениях придется пересмотреть цели проекта.

Основным назначением оценки программного проекта является вовсе не прогнозирование результата. Оценка должна определить, достаточно ли реалистичны цели проекта, чтобы их можно было достичь за счет управления проектом. Поместятся ли вещи в маленький чемодан или же придется брать большой? А может, удастся взять маленький чемодан, если внести небольшие изменения? Начальство хочет услышать от вас подобные ответы. Обычно ему не нужна точная оценка, которая скажет, что вещи в чемодан не поместятся. Вместо этого начальству нужен план, который позволит взять с собой как можно больше вещей.

Проблемы начинаются тогда, когда разрыв между деловыми целями и сроками/объемом работ, необходимым для достижением этих целей, становится слишком большим. По своему опыту могу сказать, что если исходная цель и исходная оценка расходятся в пределах 20 % в ту или иную сторону, у руководителя проекта остается достаточное пространство для маневра, чтобы за счет управления функциональностью, сроком, численностью группы и другими параметрами добиться поставленных целей; другие эксперты с этим согласны (Boehm 1981, Stutzke 2005). Если разрыв между целью и реальными потребностями окажется слишком большим, руководитель не сможет довести проект до успешного завершения, внося незначительные изменения в его параметры. Сколько бы вы ни перекладывали свои вещи и не сидели на крышке чемодана, большая гора вещей все равно не поместится в маленький чемодан, и вам придется брать большой или оставлять часть вещей. Прежде чем руководитель начнет управлять проектом для достижения поставленных целей, эти цели необходимо привести в соответствие с реальностью.

Оценки должны быть не столько идеально точными, сколько полезными. Комбинация из точных оценок, грамотно выбранных целей и качественного планирования управления позволит привести проект к результату, близкому к «оценке»

(как вы, вероятно, догадались, слово «оценка» взято в кавычки, потому что оцениваемый проект не идентичен завершенному).

Динамика изменения предпосылок проекта является главной причиной, по которой основное внимание в этой книге уделяется неформальным, а не научным методам. Точность в $\pm 5\%$ окажется бесполезной, если базовые предпосылки проекта изменятся на 100 %.

1.8. Рабочее определение «хорошей оценки»

После всего, что было сказано в предыдущих разделах, мы можем ответить на вопрос, что же следует считать хорошей оценкой.

Хорошей считается оценка, которая обеспечивает достаточно ясное представление реального состояния проекта и позволяет руководителю проекта принимать хорошие решения относительно того, как управлять проектом для достижения целей.

Это определение заложено в основу всего последующего обсуждения оценок в книге.

Дополнительные ресурсы

Conte, S.D., H.E. Dunsmore, and V.Y. Shen. «Software Engineering Metrics and Models». Menlo Park, CA: Benjamin/Cummings, 1986. В книге Конта, Дансмора и Шена приводится авторитетное обсуждение моделей оценки. В частности, в ней рассмотрен критерий «25 % отклонений в 75 % случаев», а также другие критерии оценки.

DeMarco, Tom. «Controlling Software Products». New York: NY: Yourdon Press, 1982. ДеМарко рассматривает вероятностную природу программных проектов.

Stutzke, Richard D. «Estimating Software-Intensive Systems». Upper Saddle River, NJ: Addison-Wesley, 2005. В приложении С содержится перечень мер, повышающих точность оценки.

Что такое оценка?

Какой из вас оценщик?



Ваша задача — выдать оценку, а не точное значение¹.

Филип Армор

Итак, теперь мы знаем, что такое хорошая оценка. А вот хороший ли из вас оценщик? Следующий раздел поможет ответить на этот вопрос.

2.1. Простой тест по оценке

В табл. 2.1 содержится краткий вопросник, предназначенный для проверки ваших навыков оценки. Пожалуйста, прочитайте его и выполните следующие рекомендации.

Для каждого вопроса запишите верхнюю и нижнюю границу, которые, на ваш взгляд, обеспечивают 90 % вероятность включения правильного значения. Постарайтесь избегать чрезмерного расширения или сужения диапазонов. Сделайте их достаточно широкими для того, чтобы по вашему здравому смыслу диапазоны с 90%-й достоверностью включали правильный ответ. Пожалуйста, не ищите ответы в справочниках — тест должен проверить ваши навыки оценки, а не умение работать с литературой. Вы должны ввести ответ на каждый вопрос; пропущенный ответ считается неправильным. На выполнение теста отводится 10 минут.

(Также можно скопировать лист перед заполнением, чтобы следующий читатель книги тоже мог ответить на вопросы.)

Правильные ответы на упражнение (например, географическая широта Шанхая) приводятся в приложении Б в конце книги. За каждый диапазон, включающий правильный ответ, вам начисляется одно очко.

¹ В английском языке — игра слов: estimation (оценка) — exactimation (точное значение). — Примеч. перев.

Таблица 2.1. Какой из вас оценщик?

Нижняя оценка — Верхняя оценка	Описание
[] — []	Температура поверхности Солнца
[] — []	Широта Шанхая
[] — []	Площадь континента Азия
[] — []	Год рождения Александра Македонского
[] — []	Общая стоимость американской валюты, находившейся в обращении в 2004 г.
[] — []	Мы склонны полагать, что ошибки в широких диапазонах, точнее, в очень широких диапазонах, тоже ошибки.
[] — []	Общий объем Великих Озер
[] — []	Мировые кассовые сборы фильма «Титаник»
[] — []	Общая длина линии побережья Тихого океана
[] — []	Количество названий книг, опубликованных в США с 1776 г.
[] — []	Максимальный зафиксированный вес голубого кита

Источник: навеяно аналогичным тестом из книги «*Programming Pearls*, Second Edition (Bentley, 2000).

Тест из книги «*Software Estimation*» Стива Макконнела (Microsoft Press, 2006) © 2006 Steve McConnell. Авторские права защищены. Копирование разрешается только при условии включения настоящего уведомления об авторских правах.

Что у вас получилось? (Не огорчайтесь. Большинство людей показывает не лучший результат!) Пожалуйста, запишите свой результат здесь: _____.

2.2. Обсуждение результатов теста

Зачем здесь приводится этот тест? Вовсе не для того, чтобы проверить, знаете ли вы дату рождения Александра Македонского или широту Шанхая. Он проверяет, насколько хорошо вы понимаете свои собственные способности к оценке.

Насколько достоверна «90-процентная достоверность»?

В описании теста четко сказано, что целью теста должна быть оценка с достоверностью 90 %. Тест состоит из 10 вопросов; если вы действительно оцениваете с достоверностью 90 %, вы должны были получить 9 правильных ответов¹.

Возможно, вы были осторожны, определили слишком широкие диапазоны и получили 10 правильных ответов из 10. Если вы поторопились и определили

¹ Математический смысл «90-процентной достоверности» чуть более сложен. Если бы оценка действительно производилась с 90-процентной достоверностью, то с вероятностью 34,9 % вы бы получили 10 правильных ответов, с вероятностью 38,7 % — 9 правильных ответов, 19,4 % — 8 правильных ответов. Другими словами, вероятность получения 8 и более правильных ответов равна 93 %.

диапазоны более узкие, чем следовало, возможно, вам удалось правильно получить 7 или 8 ответов из 10. Я давал этот тест сотням оценщиков. На рис. 2.1 показаны результаты для последних 600 человек, проходивших этот тест.

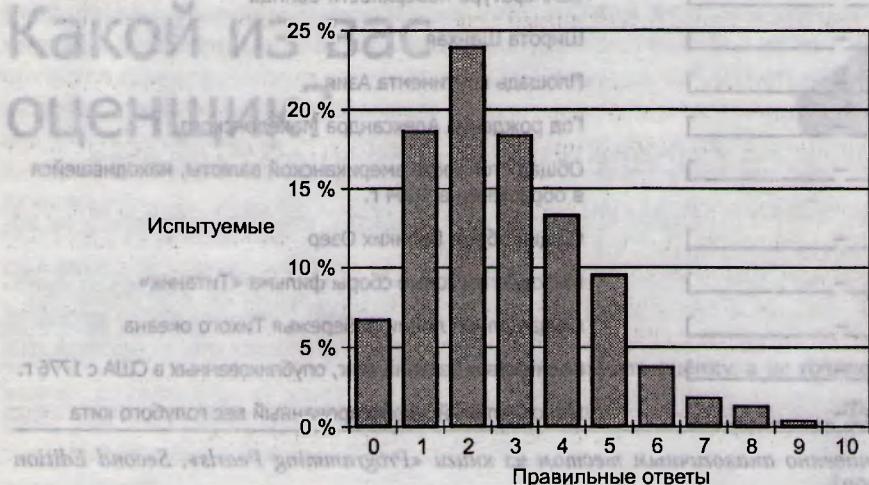


Рис. 2.1. Результаты теста «Какой из вас оценщик?» Большинство испытуемых получает 1–3 правильных ответа

Для группы, результаты которой показаны на рисунке, среднее количество правильных ответов было равно 2,8. Всего два процента испытуемых дали 5 и более правильных ответов. Никто и никогда не дал 10 правильных ответов. Я сделал вывод, что интуитивные представления большинства людей о «90 % достоверности» в действительности ближе к «30 % достоверности». Другие исследования также подтвердили мое базовое открытие (Zultner 1999, Jørgensen 2002).

Я видел множество рабочих групп, представлявших графики, «достоверные на 90 %». Потом эти графики нередко нарушались — собственно, чаще нарушались, чем соблюдались. Если бы график действительно составлялся с 90%-й достоверностью, то по статистике превышение встречалось бы лишь в одном случае из 10.

Отсюда следует, что значения достоверности в процентах (например, 90 %) имеют смысл только в том случае, если они подтверждаются неким статистическим анализом. В противном случае это не более чем подмена действительного желаемым. Как добиться *настоящей* 90 % достоверности, мы разберемся позднее в этой книге.

СОВЕТ № 5

Не предоставляйте процентные оценки достоверности (особенно «достоверные на 90 %»), если только они не подтверждаются количественными методами.

Если вы не прошли тест, приведенный ранее в этой главе, сейчас самое время вернуться и пройти его. Вас удивит, как мало будет правильных ответов даже после того, как вы прочли мои объяснения.

Как выбирать ширину диапазона?

Столкнувшись с человеком, получившим 7 или 8 правильных ответов, я спрашиваю: «Как вам это удалось?» Типичный ответ: «Я выбрал слишком широкие диапазоны».

Мой ответ: «Ничего подобного! Ваши диапазоны были слишком узкими!» Даже если вы получили 7 или 8 правильных ответов, значит, диапазоны все равно оказались недостаточно широкими для получения желаемой доли правильных ответов.

Мы склонны полагать, что оценки, выраженные в виде узких диапазонов, точнее оценок с широкими диапазонами. Нам кажется, будто широкие диапазоны свидетельствуют о невежестве или некомпетентности оценщика. Обычно все наоборот (конечно, узкие диапазоны предпочтительны в тех случаях, когда они поддерживаются фактическими данными).

СОВЕТ № 6

Избегайте искусственного сужения диапазонов. Следите за тем, чтобы диапазоны, используемые в оценках, не искали вашего представления о достоверности оценки.

Откуда возникает стремление к сужению диапазонов?

Когда вы отвечали на вопросы теста, хотелось ли вам расширить диапазоны? Или вы чувствовали, что их лучше сузить? Как правило, испытуемые говорят, что им хотелось сделать диапазоны как можно уже. Но если вы вернетесь к инструкциям и внимательно прочтете их, то обнаружите, что я не просил сужать диапазоны. Действительно, в них говорится, что диапазоны не должны быть ни слишком широкими, ни слишком узкими — достаточно широкими для того, чтобы вы были на 90 % уверены в том, что они включают правильный ответ.

После обсуждения этого вопроса с сотнями разработчиков и руководителей я пришел к выводу, что стремление к сужению диапазонов возникает самопроизвольно внутри самого оценщика. Отчасти оно обусловлено чувством профессиональной гордости — люди полагают, что узкий диапазон свидетельствует о более качественной оценке, хотя это совсем не так. Также не стоит сбрасывать со счетов опыт общения с начальством и клиентами, настаивающими на использовании чрезмерно зауженных диапазонов.

Аналогичное самопроизвольное стремление обнаруживается и в общении между клиентами и оценщиками. Йоргенсен и Сьеберг сообщают, что информация об ожиданиях клиентов оказывает сильное влияние на оценку, причем оценщики обычно не сознают, в какой степени она отражается на их результатах (Jørgensen and Sjøberg 2002).

СОВЕТ № 7

Если вы испытываете воздействие, направленное на сужение диапазона, убедитесь в том, что оно идет извне, а не рождается внутри вас.

В главах 22 и 23 обсуждаются различные способы решения проблемы для тех случаев, когда давление действительно идет из внешнего источника.

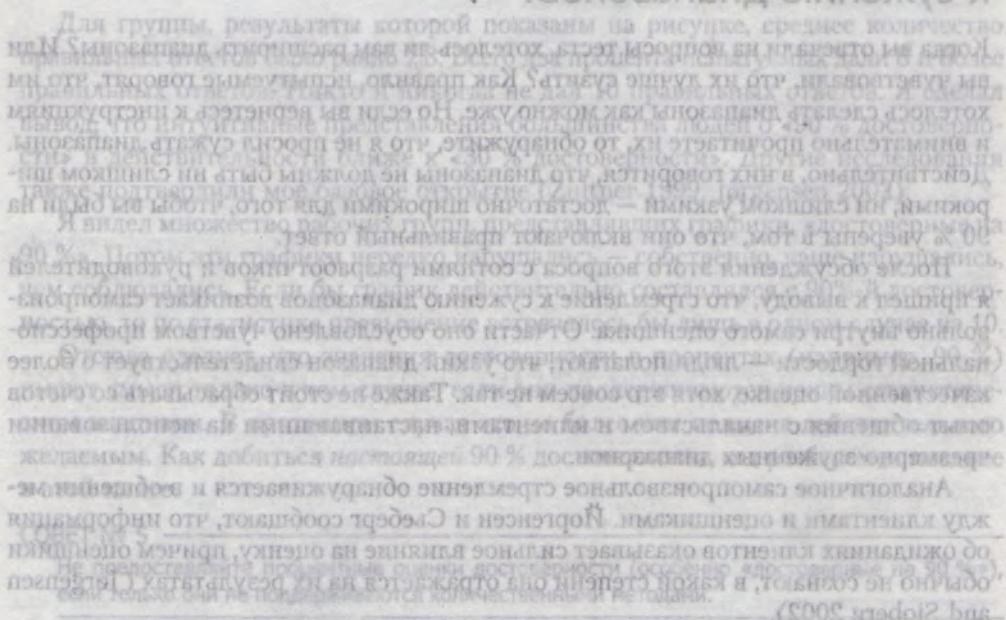
Насколько показателен тест для реальной оценки программных проектов?

В области программного обеспечения нам нечасто приходится оценивать объем воды Великих Озер или температуру поверхности Солнца. И можно ли ожидать, что вам удастся оценить общую сумму находящейся в обращении валюты США или количество книг, опубликованных в США, — особенно если вы живете в другой стране?

Разработчикам программного обеспечения часто приходится оценивать проекты из незнакомых областей — проекты, в реализации которых задействованы новые технологии, новые средства программирования, незнакомый персонал и т. д. Оценка в условиях неопределенности является вполне обычным делом. В оставшейся части этой книги я постараюсь объяснить, как добиться успеха в подобной ситуации.

Предыдущие статьи

Рис. 2.1. Результаты исследования оценщиков программных проектов



Если вы не прошли тест, приведенный ранее в этой главе, сейчас самое время пройти его заново. Итак, сколько бы времени вы потратили на него, по- сле того, как вы прочли мои объяснения?

3

О точности оценки

[Одно из распространенных определений оценки — ...] «...наиболее оптимистичный прогноз, вероятность исполнения которого отлична от нуля». ...Руководствуясь этим определением, мы неизбежно приходим к методу оценки, который можно назвать «поиском самой ранней даты, когда вы еще не можете точно сказать, что проект не будет закончен вовремя».

Том Демарко (*Tom DeMarco*)

Неточность оценок программных проектов, замутненных нереалистичными целями и недостижимыми обязательствами, оставалась проблемой в течение многих лет. Еще в 1970-х годах Фред Брукс указывал, что «нехватка календарного времени загубила больше проектов, чем все остальные причины вместе взятые» (Brooks 1975). Десять лет спустя Скотт Костелло заметил, что «давление предельных сроков является величайшим врагом разработки программного обеспечения» (Costello 1984). В 1990-х годах Кейперс Джонс указал, что «слишком подробные или иррациональные графики, вероятно, оказывают наиболее разрушительное влияние на судьбу всех программных проектов».

Том Демарко сформулировал свое общее определение в 1982 году. Несмотря на успехи, о которых я упоминал в первой главе, за прошедшие годы изменилось не так уж много. Вероятно, вы уже согласны с тем, что точные оценки ценнее неточных. В этой главе описаны преимущества точных оценок и тех данных, на которых они базируются.

3.1. Что лучше — переоценка или недооценка?

На интуитивном уровне ясно, что точная оценка закладывает идеальную основу для планирования проекта. Наличие точных оценок позволяет эффективно координировать работу между несколькими разработчиками. Результаты, выдаваемые одно группой для другой группы, планируются с точностью до дня, часа или

минуты. Но мы знаем, что точные оценки встречаются редко, и раз уж нам все равно суждено ошибиться — в какую сторону это лучше сделать? В сторону завышения или занижения?

Аргументы против переоценки

Начальство и другие заинтересованные стороны иногда опасаются, что при переоценке проекта вступит в силу закон Паркинсона — принцип, согласно которому любая работа заполняет все отведенное для нее время. Если дать разработчику 5 дней на решение задачи, которую можно решить за 4 дня, разработчик придумает, чем заняться в лишний день. Если дать группе 6 месяцев на завершение проекта, который можно завершить за 4 месяца, группа найдет способ использовать лишние два месяца. В результате некоторые начальники сознательно поджимают оценки, пытаясь избежать действия закона Паркинсона.

Другая потенциальная проблема — «студенческий синдром» Голдрэтта (Goldratt 1997). Если выделить разработчикам слишком много времени, они работают спустя рукава. Когда проект близится к концу, начинается аврал, и скорее всего, разработчики не успеют сдать проект к сроку.

Недооценка также часто применяется еще по одной похожей причине — из-за желания внушить группе разработчиков чувство срочности проекта. Обоснование выглядит примерно так.

Разработчики говорят, что проект займет 6 месяцев. Наверняка, в их оценках есть какие-нибудь допуски и излишества, которые можно отжать. Кроме того, в проект нужно заложить плановую срочность, чтобы обеспечить его приоритетное выполнение. Значит, нужно настаивать на 3-месячном сроке. Конечно, я не верю, что проект можно завершить за 3 месяца, но разработчикам назову именно этот срок. Если я прав, разработчики все сделают за 4 или 5 месяцев. В худшем случае потребуются те 6 месяцев, которые были названы в изначальной оценке.

Насколько убедительны эти аргументы? Чтобы выяснить это, необходимо изучить аргументы в пользу смещения в другую сторону, то есть переоценки.

Аргументы против недооценки

Недооценка создает множество проблем — как очевидных, так и скрытых.

- **Снижение эффективности планирования.** Заниженные оценки подрывают эффективность планирования и закладывают неверные предположения в планы выполнения некоторых операций. Они могут привести к ошибкам планирования в численности группы — скажем, заложенная в план численность группы окажется меньше необходимой. Также могут быть подорваны возможности координации между группами — если группа не успевает завершить работу к положенному сроку, другие группы не смогут использовать ее результаты.

- **Если ошибка оценки приводит к нарушению плана всего на 5 % или 10 %, она не вызывает серьезных проблем.** Однако многочисленные исследования показали, что оценки программных проектов часто оказываются неточными на 100 % и более (Lawlis, Flowe, and Thordahl 1995; Jones 1998; Standish Group 2004; ISBSG 2005). Если предпосылки планирования неверны до такой степени, планы среднего проекта настолько отрываются от реальности, что становятся практически бесполезными.
- **Статистическое снижение вероятности своевременного завершения.** Разработчики обычно склонны оценивать объем работы на 20–30 % ниже реального (van Genuchten 1991). Даже если просто воспользоваться их обычными оценками, планы проекта будут слишком оптимистичными. Дальнейшее сокращение оценок делает своевременное завершение еще менее вероятным.
- **Плохая техническая база ухудшает результат по сравнению с номиналом.** Занизенная оценка может привести к тому, что на предварительные операции (такие, как постановка требований и проектирование) будет потрачено слишком мало времени. Если же требованиям и проектированию уделено недостаточно внимания, вам придется переделывать и то и другое на более поздних стадиях проекта, причем с большими затратами, чем при своевременном выполнении этих операций (Boehm and Turner 2004, McConnell 2004a). В конечном итоге работа над проектом займет больше времени, чем при точной оценке.
- **Деструктивная динамика поздней стадии работы над проектом ухудшает результат по сравнению с номиналом.** При переходе проекта в состояние «опоздания» рабочим группам приходится тратить время на действия, не нужные для «своевременных» проектов. Вот лишь несколько примеров:
 - Дополнительные встречи с начальством с обсуждением хода работ и мер, призванных вернуть проект на правильный путь.
 - Частые переоценки для определения уточненной даты завершения проекта.
 - Общение к важным клиентам по поводу нарушения срока поставки (в том числе и посещение собраний с участием этих клиентов).
 - Подготовка промежуточных версий продукта для выставок, демонстраций и т. д. Если бы проект был готов вовремя, можно было бы использовать саму программу вместо промежуточных версий.
 - Дополнительные дискуссии по поводу того, какие требования абсолютно необходимо включить в задание из-за задержки проекта.
 - Решение проблем с наспех сделанными обходными решениями, которые пришлось реализовать ранее из-за поджимающих сроков.

У всех перечисленных действий есть одна важная особенность: они *совершенно* не нужны, если работа над проектом идет по графику. Дополнительные действия отвлекают от продуктивной работы и задерживают сдачу проекта по сравнению с точной оценкой и планированием.

Сравнивая аргументы

«Студенческий синдром» Голдрэтта может оказывать влияние на программные проекты, но я обнаружил, что самым эффективным способом его устранения является активное отслеживание задач и буферное управление (то есть управление проектом), по аналогии с предложениями Голдрэтта, а не смещение оценок.

Как видно из рис. 3.1, лучшие результаты проекта достигаются при самых точных оценках (Саймонс 1991). Если оценка занижена, неэффективность планирования ведет к повышению затрат и затягиванию проекта. Если оценка завышена, начинает действовать закон Паркинсона.



Рис. 3.1. Потери от недооценки превышают потери от переоценки, поэтому если точная оценка невозможна, старайтесь ошибаться в сторону завышения, нежели в сторону занижения

Я считаю, что закон Паркинсона применим к программным проектам; объем работы растет, заполняя все отведенное время. Однако намеренная недооценка проекта из-за закона Паркинсона оправдана лишь в том случае, если потери из-за переоценки превышают потери из-за недооценки. В области программного обеспечения потери от переоценки являются *линейными и ограниченными* — работа заполняет все отведенное время, но не более того. С другой стороны, потери от недооценки *нелинейны и неограничены* — ошибки планирования, недостаточные предварительные операции и создание новых дефектов приносят больший ущерб, чем переоценка, причем предсказать размер этого ущерба заранее почти невозможно.

СОВЕТ № 8

Избегайте намеренной недооценки. Потери от недооценки превышают потери от переоценки. Если вас беспокоит возможная переоценка, решайте проблемы посредством планирования и управления, а не за счет смещения оценки.

3.2. Достижения в области оценки программных проектов

История достижений в области оценки программных проектов помогает сделать ряд интересных выводов относительно природы возникающих проблем. В последнее время группа The Standish Group каждые два года публикует отчет «The

«Chaos Report» с описанием результатов программных проектов. В 2004 году 54 % проектов завершались с опозданием, 18 % полностью проваливались, а 28 % завершались в положенный срок и в рамках бюджета. На рис. 3.2 изображены результаты за 10 лет, с 1994 по 2004 год.

Обратите внимание: в данных The Standish Group даже не существует категории для досрочного завершения проектов! Лучшим из возможных результатов было соответствие ожиданиям «вовремя/в рамках бюджета» — а все остальные результаты ему уступали.

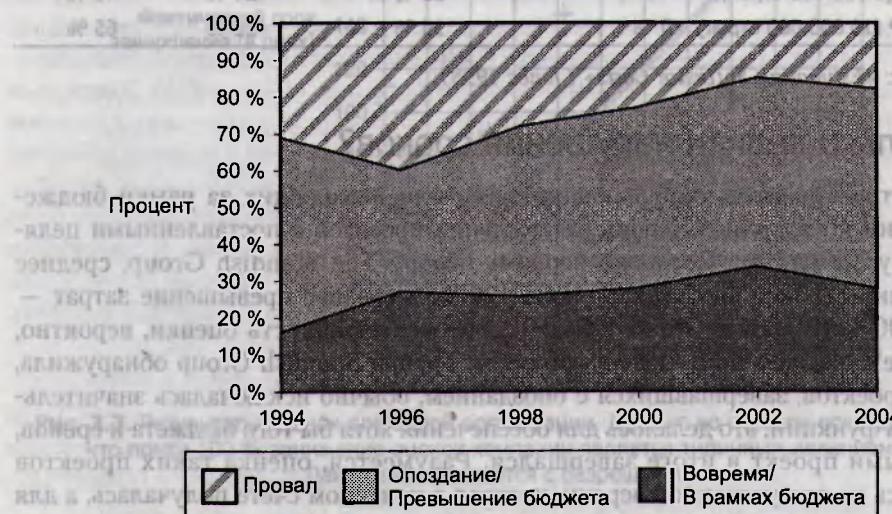


Рис. 3.2. Результаты проектов по данным отчета The Standish Group «Chaos Report» изменяются от года к году. Около 3/4 всех программных проектов сдаются с опозданием или проваливаются

Кейперс Джонс дает другое представление статистики о результатах проектов. На основании многолетних наблюдений Джонс заметил, что успех проекта зависит от его размера. Иначе говоря, в крупных проектах возникает больше проблем, чем в мелких. Таблица 3.1 доказывает это утверждение.

Как видно из данных Джонса, чем крупнее проект, тем ниже вероятность его своевременного завершения и тем выше вероятность полного провала.

В ходе многих исследований были получены результаты, соответствующие результатам The Standish Group и Джонса, — примерно четверть всех проектов завершается своевременно; еще четверть отменяется; и около половины проектов сдается с опозданием и/или превышением бюджета (Lederer and Prasad 1992; Jones 1998; ISBSG 2001; Krasner 2003; Putnam and Myers 2003; Heemstra, Siskens and van der Stelt 2003; Standish Group 2004).

Существует множество причин, по которым проекты нарушают поставленные цели. Плохая оценка — лишь одна из них, но отнюдь не единственная. Эта тема подробнее рассматривается в главе 4.

Таблица 3.1. Зависимость результата проекта от его размера

Размер в функциональных пунктах (и примерное количество строк программного кода)	Преждевременное завершение	Своевременное завершение	Опоздание	Провал (отмена)
10 FP (1000 строк)	11 %	81 %	6 %	2 %
100 FP (10 000 строк)	6 %	75 %	12 %	7 %
1000 FP (100 000 строк)	1 %	61 %	18 %	20 %
10 000 FP (1 000 000 строк)	<1 %	28 %	24 %	48 %
100 000 FP (10 000 000 строк)	0 %	14 %	21 %	65 %

Источник: «Estimating Software Costs» (Jones 1998).

Насколько велики нарушения сроков?

Количество проектов, нарушающих сроки или выходящих за рамки бюджета, — один показатель. Степень расхождения проектов с поставленными целями — другой фактор. Согласно первому обзору The Standish Group, среднее превышение сроков составляло около 120 %, а среднее превышение затрат — около 100 % (Standish Group 1994). Тем не менее точность оценки, вероятно, была еще хуже, чем показывают эти числа. Группа Standish Group обнаружила, что из проектов, завершившихся с опозданием, обычно исключалась значительная часть функций; это делалось для обеспечения хотя бы того бюджета и сроков, с которыми проект в итоге завершался. Разумеется, оценка таких проектов строилась не для усеченной версии, которая в конечном счете получалась, а для полноценной исходной версии. Если бы в опаздывающих проектах реализовывалась вся запланированная функциональность, нарушение планов было бы еще большим.

По данным одной компании

Данные о результатах проектов для одной конкретной компании, полученные от одного из моих клиентов, показаны на рис. 3.3.

Точки, сгруппированные на линии 0 в левой части графика, представляют проекты, о которых разработчики сообщили как о завершенных; тем не менее при интеграции результатов с другими группами оказалось, что проекты требуют доработки.

Диагональная линия представляет идеальную точность планирования. В идеале график должен состоять из точек данных, сгруппированных вблизи от диагональной линии. Вместо этого почти все 80 показанных точек данных находятся над линией и представляют нарушения сроков. Одна точка расположена под линией, и еще несколько — на линии. Линия демонстрирует стандартное определение «оценки» по Демарко — самая ранняя дата, к которой проект может быть завершен.

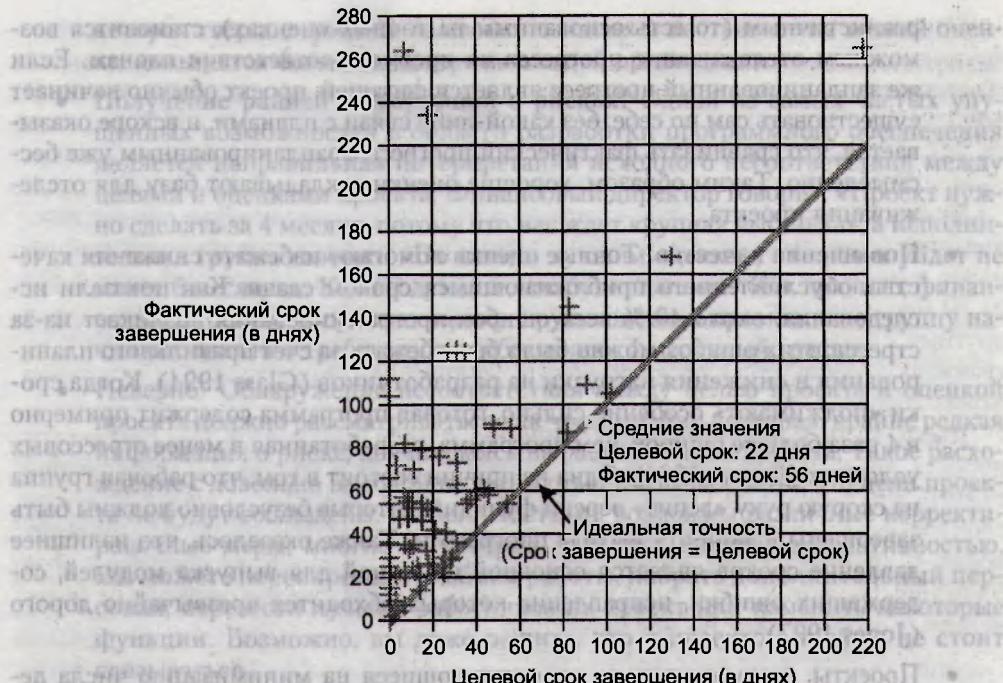


Рис. 3.3. Результаты оценок для одной организации. Данные по отрасли показывают, что почти 100 % занижение оценок компаний является типичным явлением.

Данные используются с разрешения

Общесистемная проблема программной отрасли

Мы часто говорим о проблеме оценки в отрасли разработки программного обеспечения нейтрально — вроде бы иногда проекты переоцениваются, иногда недооцениваются, нам просто не удается получить правильную оценку.

Но в отрасли разработки программного обеспечения не существует проблемы нейтральной оценки. Статистика ясно показывает, что в отрасли разработки программного обеспечения существует проблема недооценки. Прежде чем заниматься повышением точности оценок, необходимо начать с их *увеличения*. Это становится изрядной проблемой для многих организаций.

3.3. Преимущества точных оценок

Ваши оценки стали достаточно точными, и вы перестаете беспокоиться о больших ошибках оценки как в одну, так и в другую сторону. Действительно точные оценки дают немало дополнительных преимуществ.

- **Возможность отслеживания состояния проекта.** Один из лучших способов отслеживания состояния основан на сравнении запланированного прогресса с фактическим. Если запланированный прогресс был достаточно

реалистичным (то есть основанным на точных оценках), становится возможным отслеживание прогресса на предмет соответствия планам. Если же запланированный прогресс является фикцией, проект обычно начинает существовать сам по себе, без какой-либо связи с планами, и вскоре оказывается, что сравнивать фактический прогресс с запланированным уже бесполезно. Таким образом, хорошие оценки закладывают базу для отслеживания проекта.

- **Повышение качества.** Точные оценки помогают избежать снижения качества, обусловленного приближающимся сроком сдачи. Как показали исследования, около 40 % всех ошибок программирования возникает из-за стресса; этих ошибок можно было бы избежать за счет правильного планирования и снижения нагрузки на разработчиков (Glass 1994). Когда сроки «поджимают» особенно сильно, готовая программа содержит примерно в 4 раза больше ошибок, чем программа, разработанная в менее стрессовых условиях (Джонс 1994). Одна из причин состоит в том, что рабочая группа на скорую руку «лепит» версии функций, которые безусловно должны быть завершены к моменту выхода программы. Также оказалось, что излишнее давление сроков является основной причиной для выпуска модулей, содержащих ошибки, исправление которых обходится чрезвычайно дорого (Jones 1997).
- Проекты, с самого начала ориентирующиеся на минимизацию числа дефектов, обычно также имеют самое короткий срок сдачи (Jones 2000). Но если в результате давления руководства создаются нереалистичные оценки и страдает качество, ничем хорошим это не кончится: бюджет и график также пострадают.
- **Улучшение координации с функциями, не связанными с программированием.** Программные проекты обычно координируются с другими видами деятельности: тестированием, написанием документации, маркетинговыми кампаниями, обучением торгового персонала, финансовыми прогнозами, обучением службы поддержки и т. д. Ненадежный график сдачи проекта способен привести к сбоям взаимосвязанных функций, а это может нарушить весь график проекта. Хорошая оценка программного проекта предусматривает более тесную координацию всего проекта, включая как программные, так и прочие виды деятельности.
- **Повышение качества бюджета.** Как бы очевидно это ни прозвучало, точная оценка способствует выработке точного бюджета. Организация, не обеспечивающая точных оценок, подрывает свои возможности по прогнозированию стоимости проектов.
- **Повышение доверия к группе разработчиков.** Ирония судьбы: после того как группа, работающая над проектом, создает оценку, начальники, отделы маркетинга и продаж превращают эту оценку в оптимистичную цель — не взирая на все возражения разработчиков. Затем разработчики нарушают оптимистичную цель, и тогда начальники, отделы маркетинга и продаж обвиняют их в том, что они не умеют оценивать! Исполнительная группа,

которая твердо держится на своих позициях и настаивает на точной оценке, пользуется большим доверием в своей организации.

- **Получение ранней информации о рисках.** Одной из самых частых упущеных возможностей в области разработки программного обеспечения является неправильная интерпретация исходного несоответствия между целями и оценками проекта. Финансовый директор говорит: «Проект нужно сделать за 4 месяца, потому что нас ждет крупная выставка», а исполнительная группа говорит: «По нашим лучшим оценкам, на проект уйдет не менее 6 месяцев». Как вы думаете, что будет дальше? Как правило, финансовый директор обсуждает *оценку* с руководством проекта, а на группу начинают давить с требованиями обеспечить 4-месячный график.
- Неверно! Обнаружение несоответствия между целью проекта и оценкой проекта должно рассматриваться как чрезвычайно полезная, крайне редкая информация о риске, появившаяся на ранней стадии проекта. Такое расхождение с довольно высокой вероятностью указывает на то, что цели проекта не будут соблюдены. На ранней стадии возможны различные корректировочные меры, многие из которых обладают высокой эффективностью. Вы можете переопределить объем работы, набрать дополнительный персонал, перевести лучших работников на проект или изменить некоторые функции. Возможно, вы даже решите, что с проектом вообще не стоит связываться.
- Но если оставить несоответствие без внимания, возможностей корректировки на более поздней стадии будет гораздо меньше, и они будут обладать куда меньшей эффективностью. Как правило, приходится выбирать между двумя вариантами: нарушением сроков и бюджета и отказом от важных функций.

СОВЕТ № 9

Несоответствие между деловыми целями и оценкой проекта следует рассматривать как ценную информацию о возможной неудаче проекта. Принимайте меры на ранней стадии, когда еще можно что-то сделать.

3.4. Значение предсказуемости по сравнению с другими желательными атрибутами проекта

Организации, занимающиеся разработкой программного обеспечения, и люди, ведущие собственные проекты, пытаются достичь ряда целей. Вот лишь некоторые из них.

- **Срок.** Кратчайший возможный срок для реализации заданной функциональности с заданным уровнем качества.
- **Бюджет.** Минимальные затраты на реализацию заданной функциональности за заданное время.
- **Функциональность.** Максимальный набор возможностей при доступном времени и затратах.

Относительные приоритеты этих общих целей (а также других, более специфических) зависят от проекта. Разработки на базе динамических методологий обычно отдают предпочтение динамичности, повторяемости, надежности, устойчивости и наглядности (Cockburn 2001, McConnell 2002). Модель СММ института SEI обычно фокусируется на целях эффективности, улучшаемости, предсказуемости, повторяемости и наглядности.

Во время дискуссий с руководством я часто спрашиваю: «Что для вас важнее: возможность изменять решения относительно функциональности или заранее известные затраты, сроки и функциональность?» По крайней мере 8 раз из 10 мне отвечают: «заранее известные затраты, сроки и функциональность» — другими словами, *предсказуемость*. Другие эксперты в области разработки программного обеспечения также отмечали это явление (Moseman 2002, Putnam and Myers 2003).

После этого я часто говорю: «Допустим, я могу предложить вам результаты проекта, аналогичные варианту № 1 или варианту № 2 на рис. 3.4. Также будем считать, что вариант № 1 означает, что проект будет завершен за предполагаемый срок в 4 месяца, но он может быть сдан как на один месяц раньше, так и на 4 месяца позже. С другой стороны, вариант № 2 означает, что проект будет завершен в срок 5 месяцев (вместо 4), но я могу гарантировать, что отклонение составит не более недели. Какой вариант вы предпочтете?»



Рис. 3.4. При выборе между коротким средним сроком при большой неустойчивости и более длинным средним сроком при малой неустойчивости большинство коммерческих организаций выбирает второй вариант

По собственному опыту могу сказать, что почти все руководители выбирают вариант № 2. Сокращенные сроки, предполагаемые вариантом № 1, не принесут никакой пользы, потому что организация не может твердо рассчитывать на них. Так как превышение срока может достигать 4 месяцев, приходится планировать 8-месячный график вместо 4-месячного или вообще отказаться от какого-либо планирования вплоть до фактического завершения проекта. Гарантированный 5-месячный срок варианта № 2 выглядит гораздо лучше.

За прошедшие годы в отрасли разработки программного обеспечения важнейшими показателями считались время до внедрения продукта на рынок, затраты и гибкость. Каждая из этих целей важна, но руководители высшего звена более всего ценят предсказуемость. Организации должны брать на себя обязательства перед клиентами, инвесторами, поставщиками, рынком и т. д. В основе всех этих обязательств лежит предсказуемость.

Сказанное вовсе не означает, что предсказуемость должна обладать наивысшим приоритетом в ваших проектах. Скорее, речь идет о том, что вам не следует делать априорные предположения о приоритетах вашей фирмы.

СОВЕТ № 10

Многие организации ценят предсказуемость выше, чем срок разработки, затраты или гибкость. Обязательно выясните, какие показатели считаются приоритетными в вашем случае.

3.5. Недостатки распространенных методов оценки

Поскольку неудачи при оценках программных проектов встречаются сплошь и рядом, можно сделать вывод, что методы, используемые для построения оценок, малоэффективны. Их следует тщательно проанализировать... и выбросить!

Альберт Ледерер (Albert Lederer) и Джайеш Прасад (Jayesh Prasad) обнаружили, что самым распространенным методом оценки является сравнение нового проекта с похожим проектом из прошлого, причем на основании исключительно личных воспоминаний. Как выяснилось, эта методика имеет мало общего с точной оценкой. Стандартные методы, основанные на «интуиции» и «предположениях», связываются с превышением затрат и сроков (Lederer and Prasad 1992). Другие исследователи также выяснили, что догадки, интуиция, неструктурированные экспертные оценки, неформальные аналогии и т. д. являются преобладающими стратегиями, используемыми в 60–85 % всех оценок (Hihn and Habib-Agahi 1991, Heemstra and Kusters 1991, Paynter 1996, Jørgensen 2002, Kitchenham et al. 2002).

В главе 5 приводится более подробный анализ источников ошибок оценки, а в остальных главах книги представлены альтернативы для этих общих методов.

Дополнительные ресурсы

Goldratt, Eliyahu M. «Critical Chain». Great Barrington, MA: The North River Press, 1997. Голдрэтт описывает меры борьбы со «студенческим синдромом», а также подход к буферному управлению с учетом действия закона Паркинсона.

Putnam, Lawrence H. and Ware Myers. «Five Core Metrics». New York, NY: Dorset House, 2003. В главе 4 превосходно проанализирована важность предсказуемости по сравнению с другими целями проекта.

4

Откуда берутся ошибки оценки?

После этого я часто говорю: «Допустим, я могу предложить вам включить в проект аналогичные за-

Бессмысленно требовать точных формулировок, если вы считать, что вариант № 1

Джон фон Нейман

У одного из проектов факультета информационных технологий Вашингтонского университета возникли серьезные трудности с оценкой. Проект запаздывал на несколько месяцев, а превышение бюджета составило 20,5 миллиона долларов. Спектр причин был широким, от недостатков проектирования и недоразумений до вносимых в последнюю минуту изменений и многочисленных ошибок. Университет ссылался на некачественное планирование проекта. Тем не менее его нельзя было отнести к обычным программным проектам. Более того, он вообще не являлся программным проектом; речь шла о строительстве нового университетского корпуса информационных технологий и проектирования (Sanchez 1998).

Оценка программного обеспечения сопряжена с проблемами, потому что сам процесс оценки сопряжен с проблемами. В 1995 году строительство нового бейсбольного стадиона в Сиэттле было оценено в 250 миллионов долларов. Стадион был построен в 1999 году за 517 миллионов долларов — ошибка оценки составила более 100 % (Withers 1999). Самым выдающимся случаем превышения затрат за последнее время, вероятно, был проект строительства скоростного шоссе в Бостоне. Затраты, изначально оцениваемые в 2,6 миллиарда долларов, в конечном счете превысили 15 миллиардов — ошибка оценки составила более 400 % (Associated Press 2003).

Конечно, в мире программного обеспечения найдутся свои выдающиеся примеры. Ирландская система PPARC (Personnel, Payroll and Related Systems) была отменена после того, как она превысила свой бюджет в 8,8 миллиона евро на 140 миллионов евро (The Irish Times 2005)! Проект ФБР VCF (Virtual Case File) был свернут в марте 2005 года; при вложениях в 170 миллионов долларов была реализована всего 1/10 запланированных возможностей (Arnone 2005). Подрядчик жаловался, что ФБР сменило 5 руководителей информационной службы и 10 руководителей проектов, не говоря уже о 36 изменениях контракта (Knott 2005).

Подобный хаос довольно часто встречается в проектах, испытывающих проблемы с качеством оценки.

Главу, посвященную ошибкам оценки, с таким же успехом можно было назвать «Классические ошибки при оценке программного обеспечения». Даже если вы просто постараетесь избежать проблем, описанных в этой главе, половина пути к созданию точных оценок будет пройдена.

Ошибки, закрадывающиеся в оценки, идут из четырех общих источников.

- Неточная информация об оцениваемом проекте.
- Неточная информация о возможностях организации, которой будет поручено выполнение проекта.
- Чрезмерное усердие, направленное на получение точной оценки (то есть попытки оценивать изменяющиеся цели).
- Неточности, обусловленные самим процессом оценки.

В этой главе подробно рассматривается каждый источник ошибок оценки.

4.1. Источники неопределенности в оценках

Сколько стоит построить новый дом? Это зависит от дома. Сколько будет стоить веб-сайт? Это зависит от сайта. Пока все специфические особенности не будут понятны во всех подробностях, невозможно точно оценить стоимость программного проекта. Нельзя оценить объем работы, необходимый для построения чего-то нового, пока это «что-то» еще не было определено.

Разработка программного обеспечения представляет собой процесс постепенного уточнения. Вы начинаете с общей концепции продукта (своих представлений о программе, которую собираетесь построить) и уточняете ее, руководствуясь целями продукта и проекта. Иногда требуется определить бюджет и сроки, необходимые для реализации заданного объема функциональности. В других случаях нужно понять, сколько функциональности удастся реализовать в заранее определенный срок при фиксированном бюджете. Многие проекты существуют в условиях «золотой середины», то есть некоторой гибкости в выборе бюджета, срока и функциональности. В таких ситуациях различные пути, по которым может пойти программа, порождают сильно различающиеся комбинации затрат, сроков и функциональности.

Допустим, вы разрабатываете систему ввода заказов, но еще не смогли выработать требования для ввода телефонных номеров. Среди факторов неопределенности, способных повлиять на оценку программы, можно выделить следующие.

- Желает ли клиент, чтобы введенный телефонный номер проверялся на действительность?
- Если клиент хочет иметь систему проверки телефонных номеров, какую версию он предпочтет — дешевую или дорогую? (Обычно каждая конкретная функция существуют в нескольких версиях, рассчитанных на 2 часа, 2 дня или 2 недели — например, только для внутренних телефонных номеров США или для международных номеров).

- Если реализовать дешевую версию проверки телефонных номеров, не захотят ли клиент позднее переключиться на дорогую?
- Нельзя ли воспользоваться готовой системой проверки телефонных номеров или вследствие каких-то проектных ограничений необходимо разработать свою собственную?
- Как будет спроектирована система проверки? (Обычно разные проектные версии одной функции различаются по сложности как минимум на порядок.)
- Сколько времени потребуется на программирование системы проверки телефонных номеров? (Время, необходимое разным разработчикам на программирование одной возможности, также может различаться на порядок и более.)
- Должна ли система проверки телефонных номеров взаимодействовать с системой проверки адресов? Сколько времени потребуется на интеграцию двух систем?
- Каким должен быть уровень качества системы проверки телефонных номеров? (В зависимости от уровня принятых мер предосторожности количество дефектов в исходной реализации может различаться на порядок.)
- Сколько времени потребуется на отладку и исправление ошибок в реализации системы проверки телефонных номеров? (Эффективность отладки и исправления одних и тех же ошибок программистами одного уровня может различаться на порядок.)

Как видно даже из этого короткого списка, потенциальные различия в определении, проектировании и реализации одних и тех же возможностей могут накапливаться и увеличить время их реализации в сотни и более раз. А если объединить их в сотнях и тысячах функций большого проекта, вы получаете весьма значительную неопределенность в оценке самого проекта.

4.2. Конус неопределенности

Разработка программного обеспечения состоит из буквально тысяч решений относительно всех вопросов функциональности, описанных в предыдущем разделе. Неопределенность в оценках программного обеспечения обусловлена разрешением неопределенности при принятии решений. С принятием все большей доли этих решений сокращается общая неопределенность оценки.

Исследователи обнаружили, что оценкам проектов на разных стадиях присущи прогнозируемые уровни неопределенности. Конус неопределенности на рис. 4.1 показывает, что оценки становятся более точными по мере продвижения работы над проектом. (В последующем объяснении для простоты рассматривается последовательная методология разработки. В конце раздела объясняется, как применить эти концепции к итеративным проектам.)

По горизонтальной оси отложены основные ключевые этапы проекта — исходная концепция, согласованное определение проекта, завершение постановки требований и т. д. Может показаться, что терминология ориентирована на конкретные программные продукты, но это объясняется лишь ее происхождением. Скажем, «определение продукта» просто обозначает согласованное представление о программе, или *концепцию программы*, и в равной степени относится к веб-службам, внутренним системам организаций и к большинству других разновидностей программных проектов.

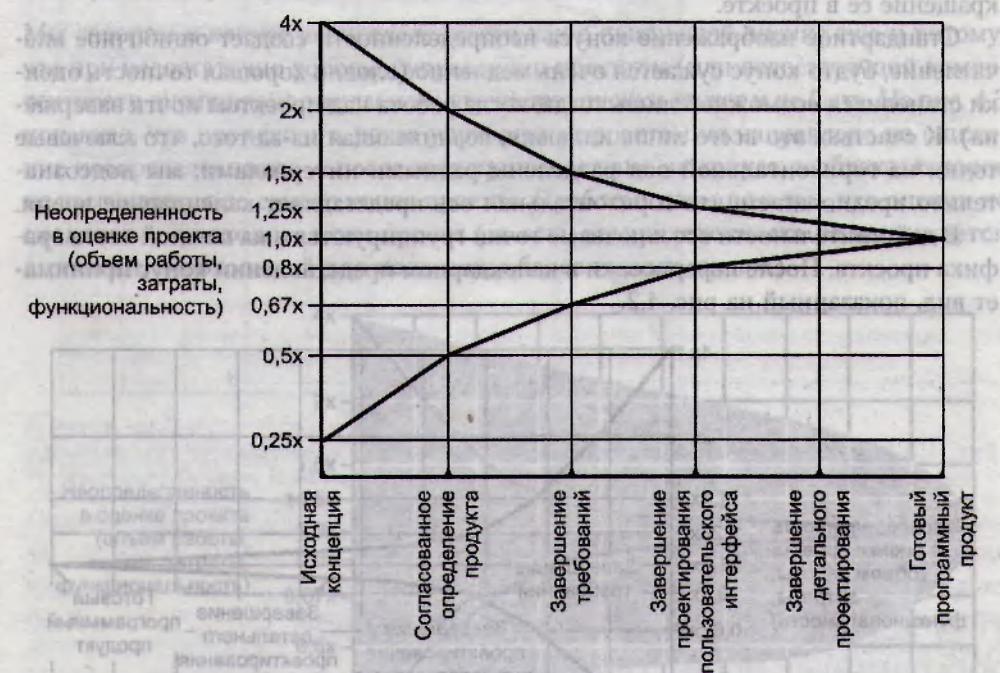


Рис. 4.1. Конус неопределенности для основных ключевых этапов проекта

По вертикальной оси отсчитывается относительная величина ошибки в проектах, создаваемых опытными оценщиками на разных стадиях работы над проектом. Оценка может относиться к затратам или объему работы на реализацию определенного набора функций, количеству функций для заданного объема работы или срока и т. д. В книге общим термином *объем* обозначается размер проекта в рабочем времени, затратах, функциональности или некоторой их комбинации.

Как видно из графика, оценки, созданные на очень ранней стадии проекта, подвержены высокой степени ошибок. Оценки, созданные на стадии исходной концепции, могут отличаться в большую или меньшую сторону до 4 раз (что также выражается в виде $0,25x$, то есть $1/4$). Полный диапазон от верхней оценки до нижней составляет $4x/0,25x$, то есть $16x$!

Руководство и клиенты часто задают вопрос: «Если дать вам еще неделю на работу над оценкой, сможете ли вы уточнить ее так, чтобы снизить степень неопределенности?» Вопрос вполне логичный, но, к сожалению, ответить на него положительно невозможно. Исследования Луиса Ларанхейра показывают, что точность оценки программного проекта зависит от степени уточнения определения программы (Laranjeira 1990). Чем точнее определение, тем точнее оценка. Оценка изменчива, прежде всего, потому, что неопределенность заложена в самом проекте. Единственным способом сокращения неопределенности в оценке является сокращение ее в проекте.

Стандартное изображение конуса неопределенности создает ошибочное впечатление, будто конус сужается очень медленно (словно хорошая точность оценки становится возможной лишь тогда, когда работа над проектом почти завершена). К счастью, это всего лишь иллюзия, возникающая из-за того, что ключевые точки на горизонтальной оси разделены равными интервалами; мы подсознательно предполагаем, что горизонтальная ось представляет календарное время.

В действительности все ключевые точки группируются в начальной части графика проекта. После перерисовки в календарном представлении конус принимает вид, показанный на рис. 4.2.



Рис. 4.2. Конус неопределенности в календарном представлении

Как видно из этого рисунка, точность оценки быстро возрастает в течение первых 30 % проекта и улучшается с $\pm 4x$ до $\pm 1,25x$.

Можно ли победить конус неопределенности?

Говоря о конусе неопределенности, необходимо учитывать одну важную (и сложную) концепцию: конус неопределенности представляет *лучшую точность*, которая может быть обеспечена в различных ключевых точках проекта. Конус представляет

ошибку оценки, разработанную опытными специалистами. Результат вполне может оказаться и хуже. Получить более точную оценку невозможно; разве что вам может больше повезти.

СОВЕТ № 11

Проанализируйте воздействие конуса неопределенности на точность вашей оценки. Ваша оценка не может быть более точной, чем это возможно на текущей позиции проекта внутри конуса.

Конус не сужается автоматически

Мы говорим о конусе неопределенности как о наилучшей оценке еще и потому, что при недостаточно хорошем управлении проектом (или недостаточной компетентности оценщиков) ожидаемого уточнения оценки может и не быть. На рис. 4.3 показано, что происходит, когда управление проектом не направлено на снижение неопределенности — последняя принимает вид не конуса, а облака, которое не рассеивается до самого конца проекта. Проблема даже не в том, что оценки не сходятся — не сходится сам проект (то есть неопределенность не вытесняется в степени, необходимой для поддержки более точных оценок).

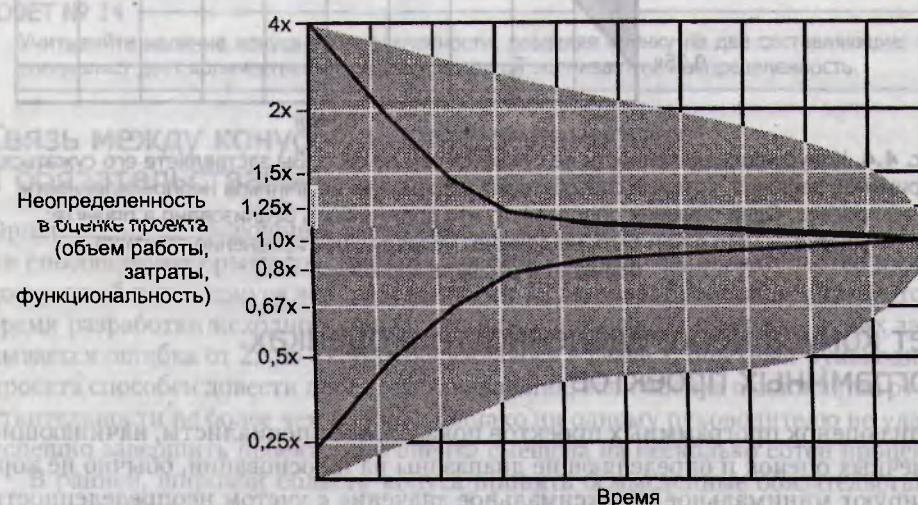


Рис. 4.3. При недостаточно качественной оценке или управлении проектом возникает «облако неопределенности», представляющее еще большую ошибку оценки по сравнению с той, что заложена в конусе

Конус сужается только при принятии решений, направленных на устранение неопределенности. Как видно из рис. 4.4, определение представлений о продукте (включая представления о том, что он *не должен* делать) способствует снижению уровня неопределенности. Определение требований (также включая требования к тому, что он не будет делать) продолжает снижать неопределенность. Проектирование пользовательского интерфейса способствует снижению риска неопределенности, возникающего из-за неверного понимания требований. Конечно, если

56 Глава 4 • Откуда берутся ошибки оценки?

продукт не был толком определен или определение продукта было позднее изменено, конус расширяется, а точность оценки ухудшается.

СОВЕТ № 12

Не надейтесь, что конус неопределенности будет сужаться сам собой. Вы должны заставить его сужаться, устранив источники неопределенности из проекта.

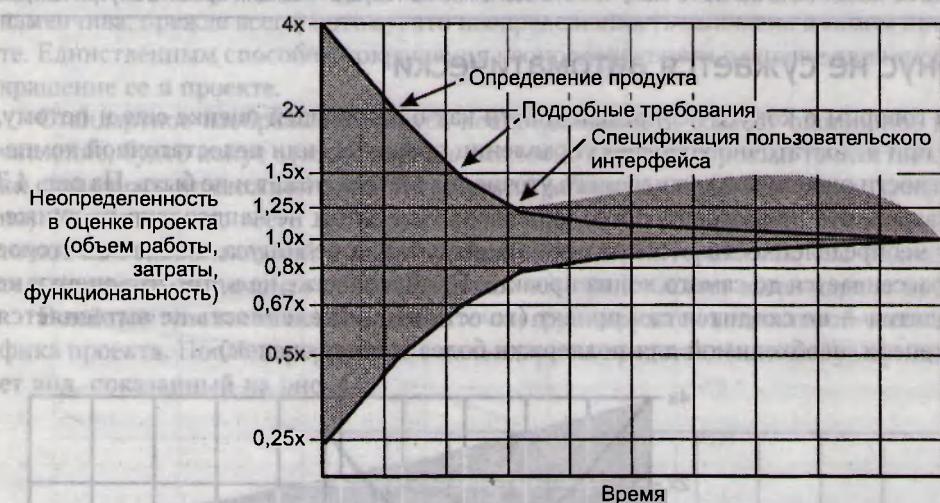


Рис. 4.4. Конус неопределенности не сужается сам по себе. Вы заставляете его сужаться, принимая решения, которые способствуют устранению источников неопределенности из проекта. Одни решения определяют, что должно быть реализовано в проекте; другие — чего в нем быть не должно. Последующее изменение решений приведет к расширению конуса

Учет конуса неопределенности в оценках программных проектов

Анализ оценок программных проектов показал, что специалисты, начинающие с точечных оценок и определяющие диапазоны на их основании, обычно не корректируют минимальное и максимальное значение с учетом неопределенности в оценке, особенно в ситуациях высокой неопределенности (Jørgensen 2002). Тенденция к использованию зауженных диапазонов преодолевается двумя способами. Во-первых, можно начать с «наиболее вероятной» оценки, а затем вычислить диапазоны с использованием заранее определенных множителей, как показано в табл. 4.1.

При использовании оценок из этой таблицы необходимо понимать, что в момент создания оценки вы еще не знаете, в какую сторону окажется смещенным фактический результат проекта — к началу или к концу диапазона.

СОВЕТ № 13

Учитывайте наличие конуса неопределенности, закладывая в своих оценках заранее определенную амплитуду неопределенности.

Таблица 4.1. Ошибка оценки в ключевых точках работы над проектом

Фаза	Ошибка
	Возможная ошибка в меньшую сторону
Исходная концепция	0,25x (-75 %)
Согласованное определение продукта	0,50x (-50 %)
Завершение проектирования пользовательского интерфейса	0,67x (-33 %)
Завершение детального проектирования	0,90 % (-10 %)
	Возможная ошибка в большую сторону
	Диапазон
	4,0x (+300 %)
	2,0x (+100 %)
	1,5x (+50 %)
	1,10x (+10 %)
	16x
	4x
	2,25x
	1,2x

Источник: По материалам «Software Estimation with Cocomo II» (Boehm et al. 2000).

Второй способ основан на отделении «оценки того, что мы знаем» от «оценки неопределенности». Один специалист дает оценки наилучшего и наихудшего случая (то есть концов диапазона), а другой оценивает вероятность того, что фактический результат войдет в этот диапазон (Jørgensen 2002).

СОВЕТ № 14

Учитывайте наличие конуса неопределенности, разделяя оценку на две составляющие: один специалист дает количественную оценку, а другой оценивает ее неопределенность.

Связь между конусом неопределенности и обязательствами

Организации, занимающиеся разработкой программного обеспечения, нередко сами способствуют срыву собственных проектов, принимая обязательства в слишком ранней точке конуса неопределенности. Если обязательства принимаются во время разработки исходной концепции или определения продукта, в них закладывается ошибка от 2x до 4x. Как обсуждалось в главе 1, опытный руководитель проекта способен довести проект до завершения, если оценка отклоняется от действительности не более чем на 20 %. Однако ни одному руководителю не удастся успешно завершить проект, если оценка смешена на несколько сотен процентов.

В ранней, широкой области конуса принять осмысленные обязательства невозможно. Эффективно работающие организации откладывают принятие обязательств до того момента, когда выполненная работа приведет к сужению конуса. В более зрелой фазе проекта (примерно 30 %) осмысленные обязательства возможны и уместны.

Конус неопределенности и итеративная разработка

Учесть воздействие конуса неопределенности в итеративных проектах несколько сложнее, чем при традиционном последовательном подходе.

Если вы работаете над проектом, который на каждой итерации проходит полный цикл разработки (то есть от постановки требований до выхода готовой версии), то на каждой итерации возникает свой миниатюрный конус неопределенности.

Перед постановкой требований для текущей итерации проект находится в точке согласованного определения продукта и подвержен 4-кратной амплитуде неопределенности в оценках. При коротких итерациях (меньше месяца) переход от согласованного определения проекта к фазам определения требований и завершения проектирования пользовательского интерфейса происходит за несколько дней, а неопределенность снижается с 4x до 1,6x. При фиксированном графике неопределенность 1,6x будет относиться к функциональности, готовой в отведенное время, а не к объему работ или срокам. Некоторые преимущества в оценке, возникающие при использовании коротких итераций, обсуждаются в разделе 8.4.

С другой стороны, при выборе подходов, при которых требования остаются неопределенными до начала каждой итерации, утрачивается возможность долгосрочного прогнозирования комбинаций затрат, сроков и функциональности (то есть их прогнозирования на несколько итераций спустя). Как рассказано в главе 3, в вашей организации приоритетным показателем может считаться именно гибкость, а возможно, руководство выберет предсказуемость проектов.

Альтернативой *полностью* итеративного цикла разработки вовсе не является *отсутствие* итераций; этот подход уже доказал свою полную неэффективность. Скорее, речь идет о *сокращении* количества итераций или выборе *других* итераций.

Многие группы разработчиков выбирают промежуточные методики, когда большинство требований определяется в начальной стадии работы над проектом, а проектирование, конструирование, тестирование и выпуск производятся короткими итерациями. Другими словами, проект движется последовательно от точки завершения проектирования пользовательского интерфейса (около 30 % календарного времени проекта), а затем переходит на более итеративный путь. Неопределенность, обусловленная воздействием конуса, снижается до ±25 %; это позволяет добиться поставленных целей при качественном управлении проектом, не утрачивая основных преимуществ итеративной разработки. Рабочие группы могут оставить часть запланированного времени на требования, которые еще предстоит определить, в конце проекта. В функциональность проекта вводится небольшая неопределенность, которая в данном случае играет скорее положительную роль — ведь данная возможность используется только в том случае, если в ходе работы над проектом будут выявлены новые желательные возможности. Промежуточный подход обеспечивает долгосрочную прогнозируемость затрат и сроков в сочетании с умеренной гибкостью в требованиях.

4.3. Хаотические процессы разработки

Конус представляет неопределенность, присущую даже в хорошо управляемых проектах. Плохое управление проектом создает дополнительную неопределенность — «факторы хаоса», которых можно было бы избежать.

Типичные примеры хаотических факторов:

- поверхностный анализ исходных требований;
- отсутствие участия конечного пользователя в постановке требований;

- плохое проектирование, порождающее многочисленные ошибки в программном коде;
- плохая методология программирования, требующая продолжительного исправления ошибок;
- недостаточная квалификация персонала;
- неполное или неумелое планирование проекта;
- присутствие «звезд» в группах;
- отказ от планирования из-за давления;
- отсутствие автоматизированной системы контроля исходных кодов.

Я привел лишь частичный список возможных хаотических факторов. За более полной информацией обращайтесь к главе 3 моей книги «*Rapid Development*» (McConnell 1996) и к статье по адресу www.stevemcconnell.com/rdenum.htm.

Источники хаоса обладают двумя общими признаками. Во-первых, каждый из них порождает неопределенность, затрудняющую оценку. Во-вторых, возникающие под их воздействием проблемы лучше всего решать на уровне управления проектом, а не на уровне оценки.

СОВЕТ № 15

Не рассчитывайте, что совершенствование методики оценки само по себе обеспечит более точную оценку в хаотических проектах. Невозможно точно оценивать процесс, который вами не контролируется. На первом шаге важнее избавиться от хаоса, чем совершенствовать оценку.

4.4. Нестабильные требования

Изменения в требованиях часто называют среди стандартных источников неопределенности при оценке (Lederer and Prasad 1992, Jones 1994, Stutzke 2005). Кроме всех типичных проблем общего плана, нестабильные требования создают две специфические проблемы.

Во-первых, нестабильные требования представляют одну из конкретных разновидностей хаотических факторов. Если требования не удастся стабилизировать, конус неопределенности не сужится и неопределенность оценки останется высокой вплоть до завершающей стадии работы над проектом.

Во-вторых, изменения в требованиях часто не отслеживаются, а проект не подвергается переоценке, как это должно быть. В хорошо управляемом проекте исходный набор требований принимается за точку отсчета, на основании которой оцениваются затраты и сроки. По мере добавления новых или пересмотра старых требований оценки затрат и стоимости также должны пересматриваться с учетом этих изменений. На практике руководители проектов часто пренебрегают обновлением оценок стоимости и затрат при изменении требований. Возникает парадоксальная ситуация: оценка исходной функциональности могла быть правильной, но после того, как проект был расширен десятками новых требований (согла-

сованных, но не учтенных), у него не остается ни малейшего шанса выдержать исходную оценку. Все согласны, что добавленные возможности были полезными, — а проект становится опоздавшим.

Конечно, описанные в книге методы помогут *улучшить* оценку в условиях высокой изменчивости требований, однако совершенствование оценки само по себе не решит проблем, возникающих из-за нестабильных требований. Эффективные меры должны приниматься на уровне управления проектом, нежели на уровне оценок. Если рабочая ситуация не позволяет стабилизировать требования, подумайте о применении альтернативных подходов, предназначенных для сред с высокой изменчивостью, — коротких итераций, экстремального программирования, Scrum, DSDM (Dynamic System Development Method) и т. д.

СОВЕТ № 16

В условиях нестабильных требований следует ориентироваться на стратегии управления проектом вместо стратегий оценки (или совместно с ними).

Оценка роста требований

Если потребуется оценить влияние нестабильности требований, один из возможных путей заключается в простом включении допуска на рост и/или изменения требований в оценки. На рис. 4.5 показан видоизмененный конус неопределенности, учитывающий приблизительно 50%-й рост требований в ходе работы над проектом. (Рисунок приведен только для наглядности. Конкретные точки данных не поддерживаются данными тех исследований, что и точки исходного конуса.)

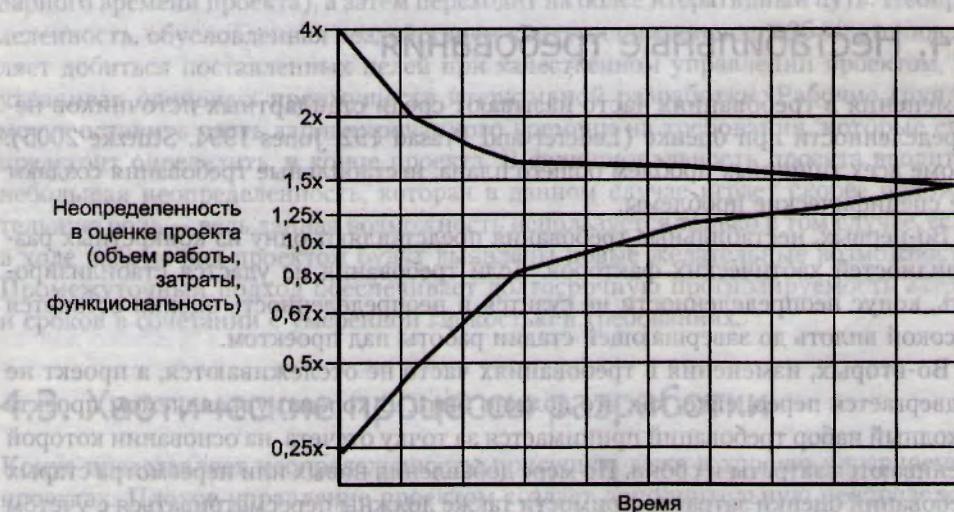


Рис. 4.5. Конус неопределенности с поддержкой изменения требований в ходе проекта
Данная методика используется многими ведущими организациями, в том числе лабораторией разработки программного обеспечения NASA, закладывающей

в свои оценки возможность 40 % роста требований (NASA SEL 1990). Аналогичная концепция присутствует в модели оценки Cocomo II (Boehm et al. 2000).

4.5. Пропущенные операции

В предыдущих разделах были описаны источники ошибок, обусловленные спецификой проекта. В остальных разделах настоящей главы мы займемся ошибками, обусловленными методами оценки.

Одним из самых распространенных источников ошибок оценки — необходимые задачи, которые не были включены в оценку проекта (Lederer and Prasad 1992, Coombs 2003). Аналитики обнаружили, что это явление проявляется как на уровне планирования проектов, так и на уровне отдельных разработчиков. В одном из исследований обнаружилось, что разработчики обычно довольно точно оценивают ту работу, которую ясно представляют себе, но при этом забывают от 20 до 30 % необходимых задач, что приводит к 20–30%-й ошибке в оценке (van Genuchten 1991).

Работа, не учтенная в оценке, делится на три общие категории: отсутствующие требования, отсутствующие действия по разработке программного обеспечения, и отсутствующие действия, не связанные с разработкой.

В табл. 4.2 перечислены требования, часто не учитываемые при оценке проекта.

Таблица 4.2. Функциональные и нефункциональные требования, часто не учитываемые при оценке проекта

Функциональные требования	Нефункциональные требования
Программа установки/настройки конфигурации	Точность
Утилита преобразования данных	Способность к взаимодействию
Связующий код, необходимый для использования программного обеспечения сторонних производителей (или распространяемого с открытыми кодами)	Модифицируемость
Справочная система	Производительность
Режимы развертывания	Портируемость
Интерфейсы с внешними системами	Надежность
	Способность к реагированию
	Возможность многократного использования
	Масштабируемость
	Безопасность
	Живучесть
	Практичность

СОВЕТ № 17

Включайте в оценку время для всех видов требований — заявленных, неявных и нефункциональных. Ничто не создается «из ничего», и ваша оценка не должна подразумевать, будто возможно обратное.

В табл. 4.3 перечислены факторы, относящиеся к разработке программного обеспечения, о которых часто забывают при составлении оценки.

62 Глава 4 • Откуда берутся ошибки оценки?**Таблица 4.3.** Действия по разработке программного обеспечения, часто не учитываемые при оценке проекта

«Время разгона» для новых участников группы	Техническая поддержка существующих систем
Обучение новых участников группы	Работа по сопровождению существующих систем во время проекта
Координация управления/встречи руководства	Работа по исправлению дефектов
Развертывание	Настройка производительности
Преобразование данных	Изучение нового инструментария
Установка	Административная работа, связанная с отслеживанием дефектов
Настройка	Координация с тестовой группой (для разработчиков)
Прояснение требований	Координация с разработчиками (для тестовой группы)
Сопровождение системы управления пересмотром проектных решений (Revision Control System)	Ответ на вопросы группы контроля качества
Поддержка сборки	Поставка материалов для написания пользовательской документации и ее рецензирование
Сопровождение сценариев, необходимых для выполнения ежедневной сборки	Рецензирование технической документации
Сопровождение автоматизированных тестов, используемых в сочетании с ежедневной сборкой	Демонстрация программы клиентам или пользователям
Установка тестовых сборок на оборудовании пользователя	Демонстрация программы на выставках
Создание тестовых данных	Демонстрация программы или прототипа высшему руководству, клиентам и пользователям
Управление программой бета-тестирования	Общение с клиентами и конечными пользователями; поддержка установки бета-версий на оборудовании клиента
Участие в техническом рецензировании	Рецензирование планов, оценок, архитектуры, детальных проектов, поэтапных планов, кода, тестовых случаев и т. д.
Работа по интеграции	
Обработка запросов на внесение изменений	
Присутствие на собраниях, посвященных управлению изменениями/назначению приоритетов	
Координация с субподрядчиками	

СОВЕТ № 18

Учитывайте в оценке все необходимые операции, связанные с разработкой программного обеспечения, не только программирование и тестирование.

В табл. 4.4 перечислены факторы, не относящиеся к разработке программного обеспечения, но также часто не учитываемые при составлении оценки.

Таблица 4.4. Факторы, не связанные с разработкой программного обеспечения, часто не учитываемые при оценке проекта

Отпуска	Собрания уровня компании
Праздники	Собрания отделов
Болезни	Настройка новых рабочих станций
Обучение	Установка новых версий инструментария на рабочих станциях
Выходные	Диагностика аппаратных и программных проблем

Некоторые проекты намеренно планируются с исключением многих факторов, перечисленных в табл. 4.4. Такой подход может сработать в течение короткого времени, но все эти операции все равно займут свое место в проектах, продолжительность которых превышает несколько недель.

СОВЕТ № 19

В проектах, продолжительность которых превышает несколько недель, следует предусматривать допуск для дополнительных факторов: праздников, отпусков по болезни, времени обучения, собраний.

Кроме использования этих таблиц для идентификации частей программного продукта или операций, которые должны быть включены в оценку, также стоит рассмотреть возможность применения методики WBS (Work Breakdown Structure) для всех стандартных факторов, учитываемых в оценке. В разделе 10.3 обсуждается оценка с применением WBS и рассматривается обобщенный пример WBS.

4.6. Необоснованный оптимизм

Оптимизм атакует программные проекты из всех источников. Что касается оценки проекта со стороны разработчиков, вице-президент Microsoft Крис Питерс (Chris Peters) заметил: «Никогда не следует опасаться того, что оценка, созданная разработчиком, окажется излишне пессимистичной, потому что разработчики всегда назначают слишком оптимистичные сроки» (Cusumano and Selby 1995). Изучив 300 программных проектов, Майкл ван Генахтен заметил, что в оценках разработчиков обычно закладывается допуск на оптимизм, составляющий от 20 до 30 % (van Genuchten 1991). Хотя руководители иногда жалуются на обратное, разработчики не склонны к завышению при оценках проектов — наоборот, их оценки обычно оказываются заниженными!

СОВЕТ № 20

Не уменьшайте оценки, полученные от разработчиков, — скорее всего, они и без того излишне оптимистичны.

Оптимизм также проявляется и в высшем звене. При изучении 100 оценок проектов Министерства обороны США обнаружился устойчивый «коэффициент фантазии», равный примерно 1,33 (Boehm 1981). Возможно, руководители проектов и начальство и не *предполагают*, что проекты могут быть сделаны на 30 % быстрее или дешевле, чем в действительности, но они *хотят*, чтобы это произошло. Это само по себе является проявлением оптимизма.

Несколько стандартных вариаций на тему оптимизма.

- Над этим проектом мы будем работать более эффективно, чем над предыдущим проектом.
- В последнем проекте все шло наперекосяк. В этом проекте проблем будет меньше.

- Проект начинался медленно, и мы прошли трудный начальный период. Тем не менее мы многому научились на своих ошибках, и ближе к концу мы будем работать гораздо эффективнее, чем в начале.

Учитывая, что оптимизм является почти неотъемлемой частью человеческой натуры, оценки программных проектов иногда нарушаются из-за явления, которое я называю «сговором оптимистов». Разработчики предоставляют оптимистичные оценки. Высшему руководству нравятся оптимистичные оценки, потому что они подразумевают достижимость желательных деловых целей. Менеджерам нравятся эти оценки, потому что они подразумевают возможность выполнения указаний начальства. Программный проект набирает силу, и никому даже в голову не приходит критически проанализировать обоснованность самих оценок.

4.7. Субъективность и пристрастие

Субъективность проникает в оценки в форме оптимизма, в форме сознательного пристрастия и в форме бессознательного пристрастия. Я различаю *пристрастие* в оценке, предполагающее намерение сместить оценку в том или ином направлении, и *субъективность*, то есть простое признание того факта, что на человеческие суждения оказывает влияние житейский опыт (как сознательно, так и бессознательно).

Что касается пристрастия, когда клиенты и руководство обнаруживают, что оценка не соответствует деловой цели, иногда они пытаются оказать больше давления на оценку, на проект и на группу. Избыточное давление, направленное на сокращение сроков работ, встречается в 75–100 % крупных проектов (Jones 1994).

Что касается субъективности, при рассмотрении различных методик оценки мы от природы склонны полагать: чем больше у нас «регуляторов», то есть мест для подгонки оценки под наш конкретный проект, тем точнее будет оценка.

В действительности все наоборот. Чем больше «регуляторов» у оценки, тем больше возможностей для проникновения субъективности. Проблема даже не в том, что оценщики намеренно смещают свои оценки. Скорее, каждый субъективный фактор слегка размывает оценку в ту или иную сторону. Если методика оценки основана на большом количестве субъективных входных факторов, их суммарное воздействие может быть весьма значительным.

Я встречал один из примеров такого рода, когда учил несколько сотен специалистов использовать модель оценки Сосомо II. Модель включает 17 множителей и 5 масштабных коэффициентов. Чтобы создать оценку Сосомо II, оценщик должен решить, какая регулировка необходима для каждого из этих 22 факторов. Факторы определяют квалификацию группы (выше или ниже среднего), относительную сложность программы (выше или ниже средней) и т. д. Теоретически эти 22 «регулятора» должны обеспечивать тонкую подстройку практически любой оценки. Похоже, на практике они лишь открывают 22 пути для проникновения ошибок в оценку.

На рис. 4.6 показаны диапазоны результатов примерно 100 групп оценщиков, определявших 17 множителей Сосомо II для одной и той же проблемы оценки.

Нижний край каждой полосы представляет самую низкую, а верхний край — верхнюю оценку группы в сеансе. Общая высота полосы соответствует изменчивости оценок.

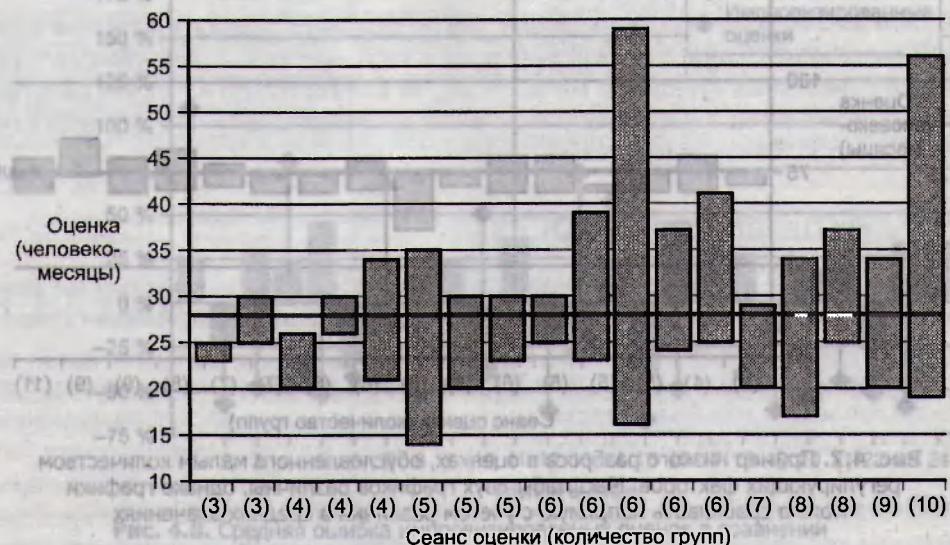


Рис. 4.6. Пример неоднозначности оценок при наличии многочисленных регулирующих факторов. Чем больше регулирующих факторов предоставляет метод оценки, тем больше возможностей для проникновения в оценку субъективности

Если бы методика оценки давала последовательный результат, мы бы увидели тесную группировку результатов вдоль горизонтальной синей линии (средняя оценка). Но как видите, разброс между оценками огромен. Наибольшая оценка превышала наименьшую в 4 раза! Среднее отклонение верхней группы от нижней в рамках одного сеанса было равно 1,7.

Важная особенность данных этого конкретного примера заключается в том, что они были освобождены от внешних смещений. Все происходило на учебном семинаре, в котором первоочередное внимание уделялось точности. Единственным смещением, влиявшим на оценки, было пристрастие, изначально присущее оценщикам. Вероятно, в реальной ситуации разброс был бы еще больше из-за возрастаания внешних факторов, влияющих на оценки.

Напротив, на рис. 4.7 показан диапазон результатов оценки для методики, включающей единственную возможность введения субъективного мнения в оценку, то есть обладающей одним «регулятором» (в данном случае этот регулятор никак не связан с множителями модели Сосото II).

Вывод «больше регулирующих факторов — не значит лучше» выходит за рамки оценки программных проектов. По выражению эксперта по прогнозированию Дж. Скотта Армстронга (J. Scott Armstrong), «один из самых неизменных и полезных выводов из исследований в области прогнозирования состоит в том, что простые методы в общем случае не уступают в точности сложным методам» (Armstrong 2001).

66 Глава 4 • Откуда берутся ошибки оценки?

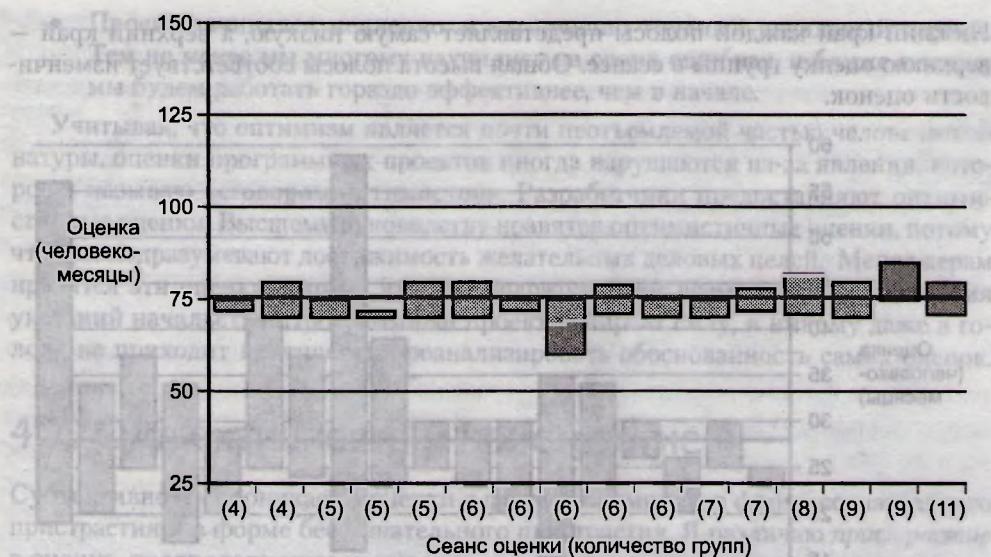


Рис. 4.7. Пример низкого разброса в оценках, обусловленного малым количеством регулирующих факторов. Масштабы двух графиков различны, однако графики можно сравнивать напрямую с учетом различий в средних значениях

СОВЕТ № 21

Избегайте включения «регуляторов» в свои оценки. Хотя может показаться, будто регуляторы улучшают точность, на самом деле они вводят в оценку субъективность и снижают реальную точность.

4.8. Импровизированные оценки

Рабочие группы иногда попадают в ловушку импровизированных оценок. Допустим, ваш начальник спрашивает: «Сколько времени потребуется для реализации предварительного просмотра на веб-сайте Gigacorp?» Вы отвечаете: «Не знаю. Наверное, около недели. Надо подумать». Вы идете к своему столу, анализируете архитектуру и код той программы, о которой спрашивал начальник, замечаете несколько обстоятельств, о которых забыли раньше, подводите итог и решаете, что потребуется около пяти недель. Вы торопитесь зайти в кабинет начальника, чтобы обновить свою первую оценку, но начальник на собрании. На следующий день вы встречаетесь с ним, но не успеваете и рта открыть, как он говорит: «Кажется, проект небольшой, так что я сходу предложил одобрить функцию предварительного просмотра на бюджетном совещании. Комитет по бюджету заинтересовался и хочет увидеть готовый результат на следующей неделе. Вы можете приступить немедленно?»

Я обнаружил, что самая безопасная политика — не давать импровизированных оценок. Ледерер и Прасад выяснили, что интуиция и догадки при оценке программных проектов коррелируются с превышениями затрат и сроков (Lederer and Prasad 1992). Я собрал данные импровизированных оценок по 24 группам. На рис. 4.8 показаны средние ошибки импровизированной оценки в этих 24 группах в сравнении с оценками, прошедшими процесс обсуждения в группах.

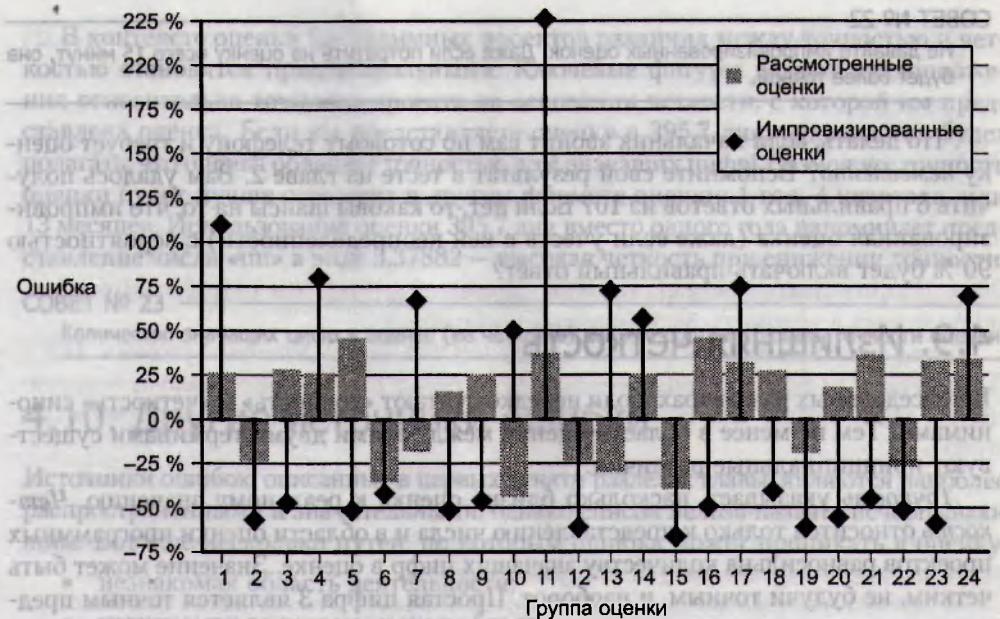


Рис. 4.8. Средняя ошибка импровизированных оценок в сравнении с рассмотренными оценками

Средняя импровизированная оценка обладала средней относительной ошибкой 67 %, тогда как у средней рассмотренной оценки ошибка составляла всего 30 % — менее половины. (Пример взят не из области оценки программного обеспечения, поэтому ошибки в процентах не следует применять буквально к оценкам программных проектов.)

Одна из ошибок, часто допускаемых при составлении оценок только на базе собственных воспоминаний, заключается в том, что новый проект сравнивается с воспоминаниями о том, сколько времени ушло на работу над прошлым проектом. К сожалению, люди часто запоминают свои *оценки* прошлых проектов, а не *фактические результаты*. Если прошлая оценка берется за основу для новой оценки, а в результате прошлого проекта эта оценка была превышена — чем это кончится? Оценщик изначально закладывает нарушение сроков в оценку нового проекта.

Хотя Ледерер и Прасад обнаружили, что догадки и интуиция положительно коррелируются с превышением оценок, также выяснилось, что использование «документированных фактов» *отрицательно* коррелируется с превышением оценок. Другими словами, существует огромная разница между импровизацией и ответом: «Не могу сказать сразу — я должен подойти к столу и проверить кое-какие заметки. Я подойду через 15 минут, если вы не против».

Это может показаться тривиальным, однако импровизированные оценки составляют одну из самых частых причин ошибок, допускаемых при работе над проектами (Lederer and Prasad 1992, Jørgensen 1997, Kitchenham et al. 2002). Отказ от импровизированных оценок — один из самых важных моментов в книге.

СОВЕТ № 22

Не давайте импровизированных оценок. Даже если потратить на оценку всего 15 минут, она будет более точной.

Что делать, если начальник звонит вам по сотовому телефону и требует оценку *немедленно*? Вспомните свой результат в тесте из главы 2. Вам удалось получить 8 правильных ответов из 10? Если нет, то каковы шансы на то, что импровизированная оценка (даже если учесть в ней неопределенности) с вероятностью 90 % будет включать правильный ответ?

4.9. Излишняя четкость

В повседневных разговорах люди нередко считают «точность» и «четкость» синонимами. Тем не менее в области оценки между этими двумя терминами существуют принципиальные различия.

Точность указывает, насколько близка оценка к реальному значению. *Четкость* относится только к представлению числа и в области оценки программных проектов равносильна количеству значащих цифр в оценке. Значение может быть четким, не будучи точным, и наоборот. Простая цифра 3 является точным представлением числа «пи» с точностью до одной цифры, но четким такое представление не является. С другой стороны, число 3,37882 обладает большей четкостью, но точность у него меньше.

Расписания полетов обладают четкостью до минут, но они не очень точны. Измерение роста людей в целых метрах может давать точный результат, но о четкости не может быть и речи.

В табл. 4.5 приводятся примеры чисел, которые являются точными и/или четкими.

Таблица 4.5. Примеры точности и четкости

Пример	Комментарий
$\pi = 3$	Точность до одной цифры, но без четкости
$\pi = 3,37882$	Четкость до 6 значащих цифр, но точность только до одной значащей цифры
$\pi = 3,14159$	И точность, и четкость до 6 значащих цифр
Мой рост = 2 метра	Точность до одной значащей цифры, но не очень четко
Расписания полетов	Четкость до минуты, но не очень точно
«На выполнение проекта потребуется 395,7 дня ± 2 месяца»	Высокая четкость, но оценка неточна для заявленной четкости
«На выполнение проекта потребуется один год»	Не очень четко, но, может быть точно
«На выполнение проекта потребуется 7,214 человека-часов»	Высокая четкость, но, вероятно, оценка неточна для заявленной четкости
«На выполнение проекта потребуется 4 человека-года»	Не очень четко, но может быть точно

В контексте оценки программных проектов различия между точностью и четкостью становятся принципиальными. Ключевые фигуры делают предположения относительно точности проекта на основании четкости, с которой им представлена оценка. Если вы представляете оценку в 395,7 дня, руководство будет полагать, что оценка обладает точностью до 4 значащих цифр! Возможно, точность оценки будет лучше отражена в другом формате оценки: 1 год, 4 квартала, или 13 месяцев. Использование оценки 395,7 дня вместо одного года напоминает представление числа «пи» в виде 3,37882 – высокая четкость при снижении точности.

СОВЕТ № 23

Количество значащих цифр в оценке (ее четкость) должно соответствовать точности оценки.

4.10. Другие источники ошибки

Источники ошибок, описанные в первых девяти разделах главы, являются наиболее распространенными и значительными, однако список нельзя назвать исчерпывающим. Вот еще несколько путей, по которым ошибка может проникать в оценку:

- незнакомая область деятельности;
- незнакомая технологическая область;
- неверное преобразование оцениваемого времени в проектное (например, предположение о том, что группа будет работать над проектом по 8 часов в день, 5 дней в неделю);
- неверное понимание статистических концепций (особенно суммирование набора оценок для лучших и худших случаев);
- бюджетные процессы, подрывающие эффективную оценку (особенно требующие согласования итогового бюджета в широкой части конуса неопределенности);
- наличие точной оценки масштаба проекта, с внесением ошибок при ее преобразовании в оценку объема работы;
- наличие точных оценок масштаба и объема работы, с внесением ошибок при их преобразовании в расписание проекта;
- завышенные ожидания от применения новых средств или методов разработки;
- упрощение оценки при ее передаче на верхние уровни управления, при формировании бюджета и т. д.

Эти темы подробно рассматриваются в следующих главах.

Дополнительные ресурсы

Armstrong, J. Scott, ed. «Principles of forecasting: A handbook for researchers and practitioners». Boston, MA: Kluwer Academic Publishers, 2001. Армстронг – один из ведущих исследователей в области рыночного прогнозирования. Многие

СОВЕТ № 22

результаты наблюдений, приведенные в книге, относятся к оценке программных проектов. Армстронг также является ведущим критиком излишне сложных моделей оценки.

Boehm, Barry, et al. «Software Cost Estimation with Cocomo II». Reading, MA: Addison-Wesley, 2000. Бем первым популяризировал идею конуса неопределенности (он называет его «воронкой»). В книге приведено его самое новое описание этого явления.

Cockburn, Alistair. «Agile Software Development». Boston, MA: Addison-Wesley, 2001. В книге Кокберна представлены динамические методы разработки, особенно полезные в условиях нестабильных требований.

Laranjeira, Luiz. «Software Size Estimation of Object-Oriented Systems», IEEE Transactions of Software Engineering, May 1990. В статье представлено теоретическое обоснование для конуса неопределенности, выявленного эмпирическим путем.

Tockey, Steve. «Return on Software». Boston, MA: Addison-Wesley, 2005. В главах 21–23 обсуждаются основные принципы и общие методы, а также рассказано, как учесть неточность в оценке. В книге Токи приведено подробное описание построения собственного конуса неопределенности.

Wieggers, Karl. «More About Software Requirements: Thorny Issues and Practical Advice». Redmond, WA: Microsoft Press, 2006.

Wieggers, Karl. «Software Requirements, Second Edition». Redmond, WA: Microsoft Press, 2003. Методы, описанные Вигерсом в этих двух книгах, помогают качественно сформулировать исходные требования, что способствует существенному снижению неустойчивости требований на последующих стадиях проекта.

Таблица 4.5. Примеры точности и четкости

Точность	Четкость
Точность до одной цифры, но не более четырех	Точность до трех цифр, но не более четырех
«Мой рост = 2 метра»	«На выполнение проекта потребуется 395,7 дня ± 2 месяца»
«На выполнение проекта потребуется 7,214 человеко-часов»	«На выполнение проекта потребуется 7,214 человеко-часов»

5

Факторы, влияющие на оценку

Сколько будет $68 + 73$?

Инженер: «141». Коротко и мило.

Математик: « $68 + 73 = 73 + 68$ по свойству коммутативности сложения». Верно, но не слишком полезно.

Бухгалтер: «Обычно 141, но в каких целях вы собираетесь использовать результат?»

Барри В. Бем и Ричард Э. Фарли

Факторы, оказывающие влияние на программный проект, следует подвергнуть разностороннему анализу. Хорошее знание этих факторов повышает точность оценки и улучшает понимание общей динамики программного проекта.

Разумеется, размер проекта является самым значительным фактором, определяющим объем работ, затраты и сроки. На втором месте стоит тип разрабатываемой программы, а на третьем — личные факторы. Язык программирования и рабочая среда не оказывают первостепенного влияния на исход проекта, но они оказывают первостепенное влияние на оценку. В этой главе представлены первостепенные факторы в порядке снижения значимости, а в конце приводится сводка второстепенных факторов.

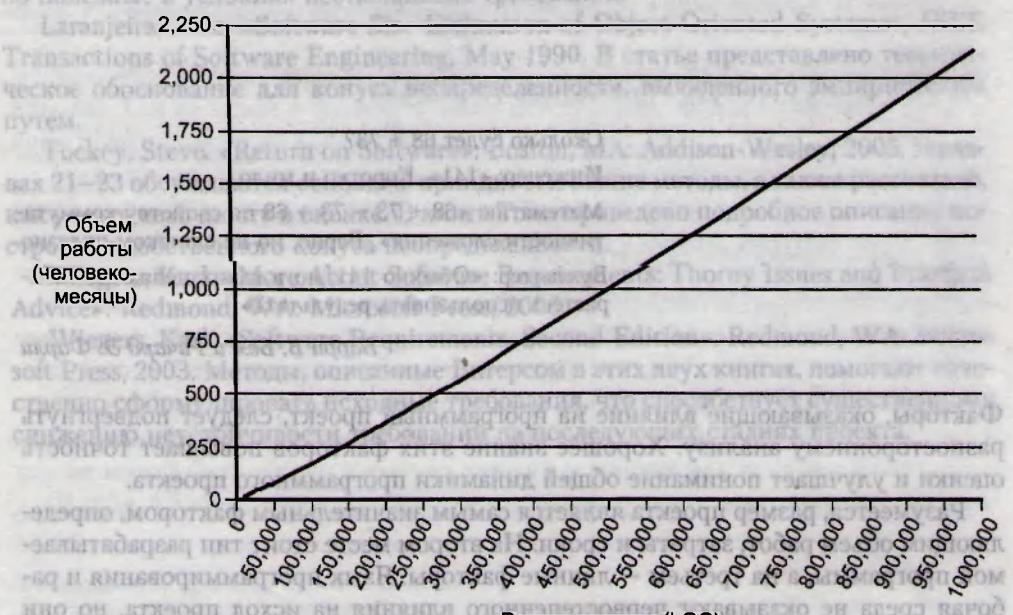
5.1. Размер проекта

Важнейшим фактором влияния в оценке программного обеспечения является размер разрабатываемой программы, потому что он подвержен наибольшему разнообразию по сравнению со всеми остальными факторами. На рис. 5.1 показана зависимость роста объема работ в среднем проекте бизнес-системы при увеличении размера проекта с 25 000 до 1 000 000 строк кода. Размер проекта на рисунке выражается в строках программного кода (LOC), но динамика останется одной и той же независимо от того, в чем измеряется размер — в функциональных пунктах, длине списка требований, количестве веб-страниц или любых других показателях, выраждающих те же диапазоны.

Как видно из диаграммы, система, состоящая из 1 000 000 строк кода, потребует гораздо большего объема работы, чем система, состоящая из 100 000 строк.

Утверждение о том, что размер проекта является основным фактором, определяющим затраты, покажется кому-то тривиальным, однако многие организации часто нарушают этот основополагающий принцип в двух отношениях:

- затраты, объем работы и сроки оцениваются без информации о размере проекта;
- затраты, объем работы и сроки не регулируются при сознательном увеличении размера проекта (то есть в ответ на запросы о внесении изменений).



Источник: По данным модели оценки Cocomo II при номинальных издержках масштаба (Boehm, et al 2000)

Рис. 5.1. Рост объема работ в типичном проекте бизнес-системы. Конкретные числовые показатели имеют смысл только для средних бизнес-систем, но общая динамика применима к программным проектам всех типов

СОВЕТ № 24

Приложите соответствующие усилия для оценки размера программы, над которой вы работаете. Размер вносит наибольший вклад в определение объема работы и сроков.

Почему в книге размер задается в количестве строк программного кода?

Люди, не имеющие опыта оценки, иногда спрашивают, действительно ли количество строк программного кода является содержательным показателем для измерения размера программы. Во-первых, многие современные среды программирования

в меньшей мере ориентированы на строки кода, чем старые среды. Во-вторых, не-малая часть процесса разработки программного обеспечения — постановка требований, проектирование и тестирование — не производят программный код, измеряе-мый в строках. Если вас интересует, как эти факторы отражаются на полезности измерения размера программы в строках кода, обратитесь к разделу 18.1.

Издержки масштаба

Мы часто полагаем, что на построение системы, в 10 раз большей другой, потребу-ется в 10 раз больше усилий. Однако объем работы для системы в 1 000 000 строк превышает 10-кратный объем работы для системы в 100 000 строк, а последний более чем в 10 раз превышает объем работы для системы в 10 000 строк.

Основная проблема заключается в том, что крупные проекты требуют коорди-нации между большим количеством групп, которым приходится больше общаться между собой (Brooks 1995). С ростом размера проекта число коммуникационных путей связи между людьми растет в *квадратичной* зависимости от количества участников проекта¹. Динамика роста показана на рис. 5.2.

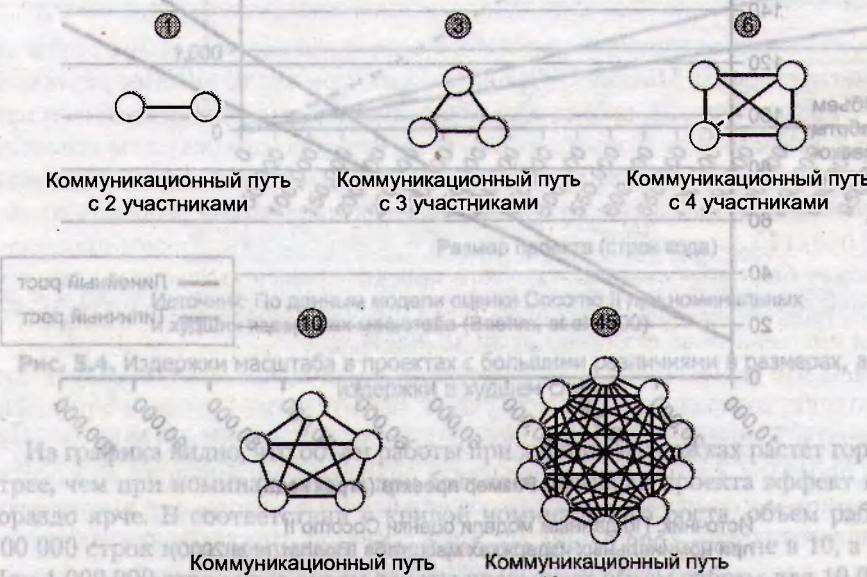


Рис. 5.2. Количество коммуникационных путей в проекте возрастает

Следствием экспоненциального роста количества коммуникационных ка-налов (наряду с другими факторами) является экспоненциальный рост трудоем-кости с увеличением размера проекта. Данное явление называется *издержками масштаба* (diseconomy of scale).

¹ Количество каналов равно $n \times (n-1)$, то есть функция порядка n^2 .

Вне области программного обеспечения мы обычно говорим об **экономии масштаба**, а не об издержках масштаба. Экономия масштаба означает примерно следующее: «Если построить больший обрабатывающий завод, мы сможем сократить стоимость единицы товара». Другими словами, чем больше объем, тем ниже стоимость.

С издержками масштаба дело обстоит наоборот — с увеличением масштаба системы стоимость каждой единицы повышается. Если бы в области программного обеспечения проявлялся эффект экономии масштаба, то стоимость системы в 100 000 строк была бы меньше 10-кратной стоимости системы из 10 000 строк. На практике почти всегда наблюдается обратное явление.

На рис. 5.3 продемонстрированы типичные издержки масштаба в области программного обеспечения по сравнению с увеличением объема работы, ассоциируемого с линейным ростом.

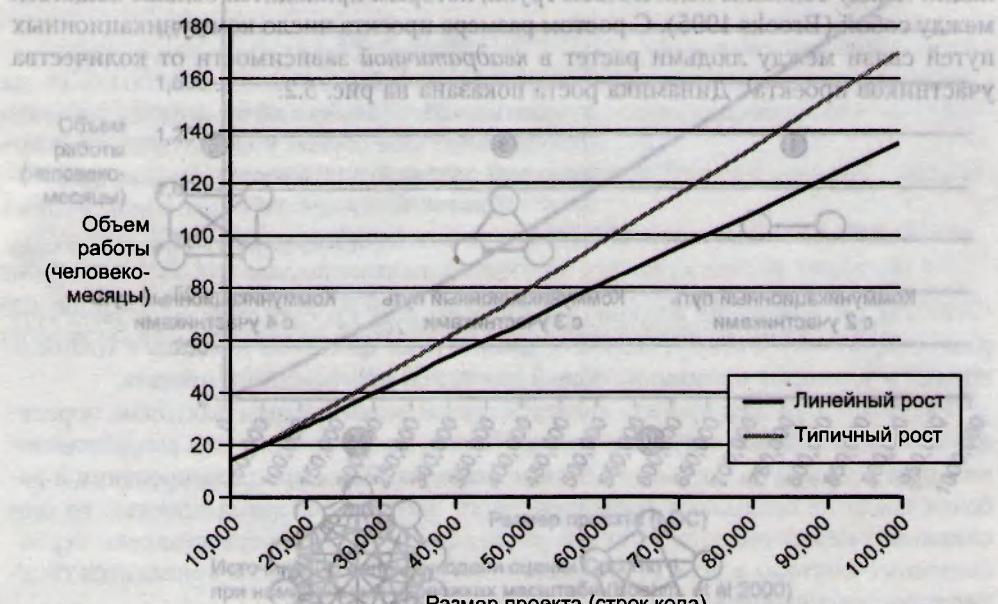
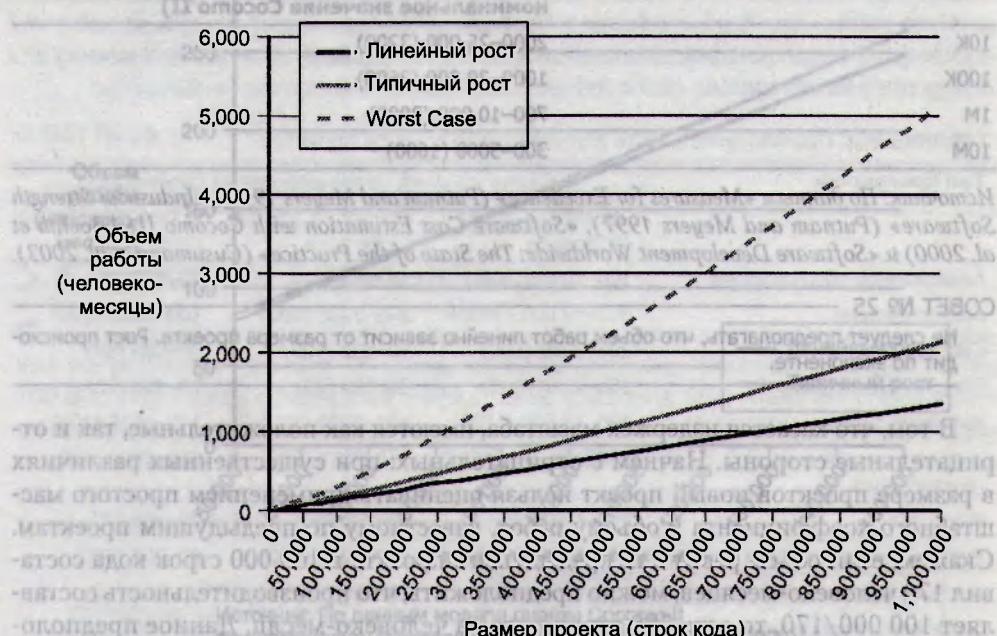


Рис. 5.3. Издержки масштаба в типичных проектах бизнес-систем от 10 000 до 100 000 строк кода

Как видно из графика, в данном примере на систему в 10 000 строк кода потребуется 13,5 человека-месяцев. Если бы объем работ возрастал линейно, то система в 100 000 строк потребовала бы 135 человеко-месяцев, но в действительности она требует 170 человеко-месяцев.

На рис. 5.3 эффект издержек масштаба не выглядит особенно впечатляющим. Действительно, в диапазоне от 10 000 до 100 000 строк он не так уж силен. Тем не менее два фактора способны сделать эффект издержек масштаба более ради-

кальным. Один фактор — это большие различия в размерах проекта, а другой — условия проекта, снижающие производительность при росте размера проекта. На рис. 5.4 изображен диапазон результатов для проектов в интервале от 10 000 до 1 000 000 строк. Кроме номинальных издержек масштаба, на графике также показаны издержки в наихудшем случае.



Источник: По данным модели оценки Cocomo II при номинальных и худших издержках масштаба (Boehm, et al 2000)

Рис. 5.4. Издержки масштаба в проектах с большими различиями в размерах, а также издержки в худшем случае

Из графика видно, что объем работы при худших издержках растет гораздо быстрее, чем при номинальных, а при больших размерах проекта эффект выражен гораздо ярче. В соответствии с кривой номинального роста, объем работы для 100 000 строк кода превышает объем работы для 10 000 строк не в 10, а в 13 раз. При 1 000 000 строк кода объем работы превышает объем работы для 10 000 строк уже не в 100, а в 160 раз.

С худшими издержками дело обстоит еще хуже. Объем работы для худшего случая при 100 000 строк кода в 17 раз превышает объем для 10 000 строк, а при 1 000 000 строк он больше не в 100, а в целых 300 раз!

В табл. 5.1 продемонстрирована общая связь между размером проекта и производительностью.

Цифры, приведенные в таблице, действительны только для сравнения между диапазонами размеров. Тем не менее представленная ими общая тенденция весьма важна. Производительность в мелких проектах в 2–3 раза превышает

76 Глава 5 • Факторы, влияющие на оценку

производительность в крупных проектах, а между тем самый мелкий проект может превосходить самый крупный по производительности в 5–10 раз.

Таблица 5.1. Связь между размером проекта и производительностью

Размер проекта (в строках кода)	Строк кода на человека-год (в скобках — номинальное значение Cocomo II)
10K	2000–25 000 (3200)
100K	1000–20 000 (2600)
1M	700–10 000 (2000)
10M	300–5000 (1600)

Источник: По данным «Measures for Excellence» (Putnam and Meyers 1992), «Industrial Strength Software» (Putnam and Meyers 1997), «Software Cost Estimation with Cocomo II» (Boehm et al. 2000) и «Software Development Worldwide: The State of the Practice» (Cusumano et al. 2003).

СОВЕТ № 25

Не следует предполагать, что объем работ линейно зависит от размера проекта. Рост происходит по экспоненте.

В том, что касается издержек масштаба, имеются как положительные, так и отрицательные стороны. Начнем с отрицательных: при существенных различиях в размере проектов новый проект нельзя оценивать применением простого масштабного коэффициента к объему работ, известному по предыдущим проектам. Скажем, если объем работ для предыдущего проекта в 100 000 строк кода составил 170 человеко-месяцев, можно предположить, что производительность составляет $100\ 000 / 170$, то есть 588 строк кода на человека-месяц. Данное предположение может быть разумным для другого проекта примерно такого же размера, но если новый проект в 10 раз больше, такая оценка производительности может оказаться смещенной на величину от 30 до 200 %.

Впрочем, это еще не все: в области неформальной оценки не существует простой методики, которая бы позволяла учесть значительные различия в размерах двух проектов. Если вы оцениваете проект, по размеру значительно отличающийся от тех проектов, с которыми ранее имела дело ваша организация, вам потребуется оценочная программа, использующая научные методы для вычисления оценки нового проекта по результатам старых проектов. Моя компания предлагает бесплатную программу Construx® Estimate™, выполняющую подобные оценки. Программу можно загрузить по адресу www.construx.com/estimate.

СОВЕТ № 26

Используйте специализированные программы для вычисления влияния издержек масштаба.

Когда можно смело игнорировать издержки масштаба

После всех плохих новостей настал черед хороших. Как правило, проекты, которыми занимается организация, имеют сходные размеры. Если новый проект по размеру мало отличается от прошлых проектов, обычно при его оценке можно

использовать простой масштабный коэффициент — скажем, количество строк кода на человеко-месяц. На рис. 5.5 показаны относительно малые различия при линейном и экспоненциальном росте в конкретном диапазоне размеров.

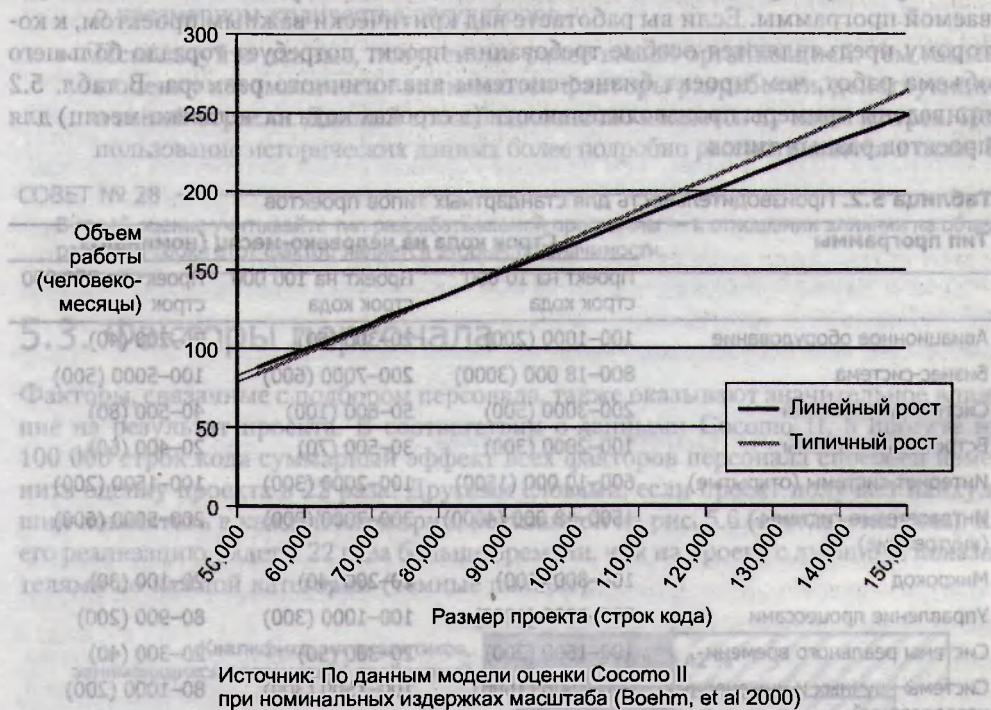


Рис. 5.5. Различия между оценками, полученными с применением простых коэффициентов, и оценками с учетом издержек масштаба, в одном диапазоне размеров будут более или менее минимальными

СОВЕТ № 27

Если ранее вы уже завершали предыдущие проекты, размер которых незначительно отличается от размера оцениваемого проекта (в диапазоне, в котором самый большой проект превосходит самый мелкий не более чем в 3 раза), для оценки нового проекта можно смело использовать масштабный коэффициент (скажем, количество строк кода на человеко-месяц).

Важность издержек масштаба при оценке программных проектов

В мире оценки программных проектов на точное определение влияния издержек масштаба направляются значительные усилия. Несмотря на важность этого фактора, следует помнить, что главным определяющим фактором в оценке является простой размер проекта. Эффект издержек масштаба является фактором второго порядка, поэтому основные усилия следует направить на получение хорошей оценки размера. О том, как создаются оценки размера, будет рассказано в главе 18.

5.2. Тип программы

После размера проекта наибольшее влияние на оценку оказывает тип создаваемой программы. Если вы работаете над критически важным проектом, к которому предъявляются особые требования, проект потребует гораздо большего объема работ, чем проект бизнес-системы аналогичного размера. В табл. 5.2 приведены примеры производительности (в строках кода на человека-месяц) для проектов разных типов.

Таблица 5.2. Производительность для стандартных типов проектов

Тип программы	Строк кода на человека-месяц (номинал)		
	Проект на 10 000 строк кода	Проект на 100 000 строк кода	Проект на 250 000 строк кода
Авиационное оборудование	100–1000 (200)	20–300 (50)	20–200 (40)
Бизнес-система	800–18 000 (3000)	200–7000 (600)	100–5000 (500)
Системы управления	200–3000 (500)	50–600 (100)	40–500 (80)
Встроенные системы	100–2000 (300)	30–500 (70)	20–400 (60)
Интернет-системы (открытые)	600–10 000 (1500)	100–2000 (300)	100–1500 (200)
Интррасетевые системы (внутренние)	1500–18 000 (4000)	300–7000 (800)	200–5000 (600)
Микрокод	100–800 (200)	20–200 (40)	20–100 (30)
Управление процессами	500–5000 (1000)	100–1000 (300)	80–900 (200)
Системы реального времени	100–1500 (200)	20–300 (50)	20–300 (40)
Системы научных и инженерных исследований	500–7500 (1000)	100–1500 (300)	80–1000 (200)
Коммерческие пакеты	400–5000 (1000)	100–1000 (200)	70–800 (200)
Системные программы/драйверы	200–5000 (600)	50–1000 (100)	40–800 (90)
Телекоммуникации	200–3000 (600)	50–600 (100)	40–500 (90)

Источник: По данным «Measures for Excellence» (Putnam and Meyers 1992), «Industrial Strength Software» (Putnam and Meyers 1997) и «Five Core Metrics» (Putnam and Meyers 2003).

Как видно из таблицы, группа, разрабатывающая интрасетевую систему для внутреннего использования, может генерировать код в 10–20 раз быстрее, чем группа, работающая над проектом управления авиационным оборудованием, системой реального времени или встроенной системой. Таблица также в очередной раз демонстрирует издержки масштаба: в проектах на 100 000 строк код генерируется гораздо менее эффективно, чем в проектах на 10 000 строк, а в проектах на 250 000 строк эффективность оказывается еще ниже.

Область, для которой создается программное обеспечение, можно учесть следующими способами.

- Используйте результаты из табл. 5.2 в качестве отправной точки. В этом случае обратите внимание на то, что диапазоны в таблице широки — обычно верхняя граница отличается от нижней на порядок.

- Используйте модель оценки (такую, как Сосомо II) и отрегулируйте параметры оценки в соответствии со спецификой разрабатываемой программы. Если вы пойдете по этому пути, вспомните предостережения из главы 4 о чрезмерном количестве регуляторов.
- Используйте данные, полученные ранее вашей организацией; тем самым в оценку автоматически включаются факторы разработки, действующие в вашей отрасли. Данный способ однозначно является лучшим из трех. Использование исторических данных более подробно рассматривается в главе 8.

СОВЕТ № 28

В своей оценке учитывайте тип разрабатываемой программы — в отношении влияния на объем работы и сроки этот фактор является вторым по значимости.

5.3. Факторы персонала

Факторы, связанные с подбором персонала, также оказывают значительное влияние на результат проекта. В соответствии с данными Сосомо II, в проекте на 100 000 строк кода суммарный эффект всех факторов персонала способен изменить оценку проекта в 22 раза! Другими словами, если проект получает наихудший показатель в каждой категории, показанной на рис. 5.6 (светлые полосы), на его реализацию уйдет в 22 раза больше времени, чем на проект с лучшими показателями по каждой категории (темные полосы).

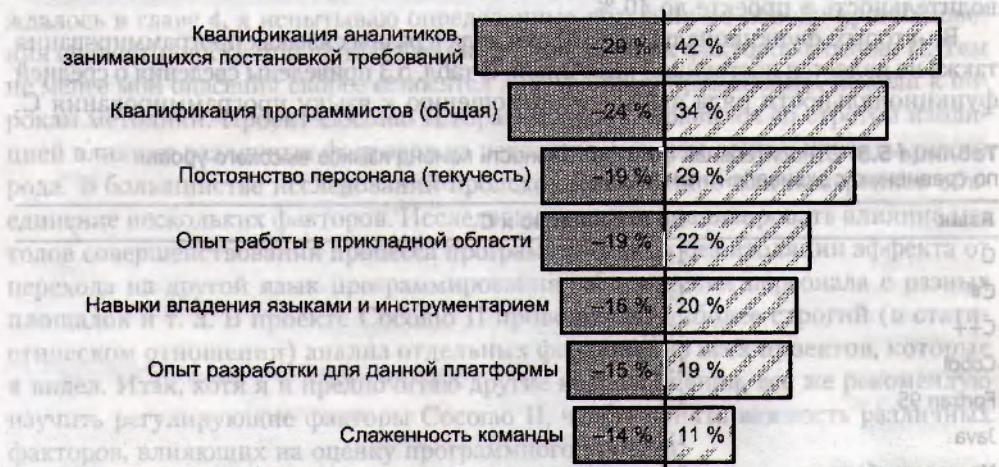


Рис. 5.6. Зависимость объема работы над проектом от факторов персонала. В зависимости от силы или слабости каждого фактора результаты проекта могут изменяться на указанную величину, то есть если выработкой требований будут заниматься худшие аналитики, объем работы возрастает на 42 % по сравнению с номиналом, а при лучших аналитиках он снижается на 29 %. Воздействие этих факторов из модели Сосомо II подтверждается многочисленными исследованиями, начавшимися еще в 1960-х годах и демонстрировавшими

различия от 10:1 до 20:1 в производительности отдельных участников и целых групп (Sackman, Erikson, and Grant 1968; Weinberg and Schulman 1974; Curtis 1981; Mills 1983; Boehm, Gray, and Seewaldt 1984; DeMarco and Lister 1985; Curtis et al 1986; Card 1987; Boehm 1987b; Boehm and Papaccio 1988; Valett and McGarry 1989; Boehm et al. 2000).

Одно из следствий подобных расхождений заключается в том, что точная оценка проекта невозможна без некоторого представления о том, кто будет заниматься его выполнением, потому что производительность работников может отличаться в 10 раз и более. Тем не менее в рамках одной конкретной организации такого разброса, скорее всего, не будет, потому что разработчики как верхнего, так и нижнего уровня обычно склонны выбирать организации, в которых работают другие люди аналогичного уровня (Mills 1983, DeMarco and Lister 1999).

Есть и другое следствие: выбор метода, обеспечивающего наиболее точную оценку, зависит от того, знаете ли вы, кто именно будет выполнять оцениваемую работу. Эта тема обсуждается в главе 16.

5.4. Язык программирования

Язык программирования, используемый в проекте, влияет на оценку по меньшей мере в четырех отношениях.

Во-первых, как видно из рис. 5.6, опыт работы группы с конкретным языком и инструментарием, используемыми в проекте, способен изменить общую производительность в проекте до 40 %.

Во-вторых, функциональность строки кода в разных языках программирования также не является постоянной величиной. В табл. 5.3 приведены сведения о средней функциональности ряда языков по отношению к языку программирования С.

Таблица 5.3. Относительная функциональность команд языков высокого уровня по сравнению с эквивалентным кодом С

Язык	По отношению к С
C	1:1
C#	1:2,5
C++	1:2,5
Cobol	1:1,5
Fortran 95	1:2
Java	1:2,5
Макроассемблер	2:1
Perl	1:6
Smalltalk	1:6
SQL	1:10
Visual Basic	1:4,5

Источник: По данным «Estimating Software Costs» (Jones 1998) и «Software Cost Estimation with Cocomo II» (Boehm 2000).

Если используемый язык программирования заранее известен, этот пункт не относится к вашей оценке. Но если вы обладаете некоторой свободой выбора языка программирования, из таблицы видно, что такие языки, как Java, C# и Microsoft Visual Basic, обычно превосходят по продуктивности C, Cobol или макроассемблер.

Третий фактор, также относящийся к языкам, — широта возможностей инструментария и рабочих сред. Согласно данным Сосомо II, слабый инструментарий и рабочая среда способны увеличить объем работы над проектом примерно на 50 % по сравнению с сильными инструментариями и рабочими средами (Boehm et al. 2000). Также следует понимать, что выбор языка программирования может определить выбор инструментария и рабочей среды.

Последний фактор, связанный с языком программирования, заключается в том, что разработчики, использующие интерпретируемые языки, обычно работают продуктивнее тех, кто применяет компилируемые языки — выигрыш составляет до 2 раз (Jones 1986a, Preschelt 2000).

Концепция объема функциональности на строку кода будет рассматриваться далее в разделе 18.2.

5.5. Другие факторы

Модель оценки Сосомо II уже неоднократно упоминалась в этой главе. Как обсуждалось в главе 4, я испытываю определенные опасения по поводу проникновения субъективности при использовании регулирующих факторов Сосомо II. Тем не менее мои опасения скорее относятся к неудачному применению, нежели к покоркам методики. Проект Сосомо II гораздо лучше справился со строгой изоляцией влияния различных факторов на исход проекта, чем другие проекты такого рода. В большинстве исследований происходит случайное или намеренное объединение нескольких факторов. Исследование может анализировать влияние методов совершенствования процесса программирования без изоляции эффекта от перехода на другой язык программирования, объединения персонала с разных площадок и т. д. В проекте Сосомо II проводится наиболее строгий (в статистическом отношении) анализ отдельных факторов из всех проектов, которые я видел. Итак, хотя я и предпочитаю другие методы оценки, все же рекомендую изучить регулирующие факторы Сосомо II, чтобы понять важность различных факторов, влияющих на оценку программного проекта.

В табл. 5.4 перечислены рейтинги 17 множителей объема работы в модели Сосомо II. В столбце «Очень низкие» представлена величина, на которую следует отрегулировать оценку объема работы для лучшего (или худшего) воздействия данного фактора. Например, если группа обладает крайне низким опытом работы в прикладной области, номинальная оценка объема работы Сосомо II умножается на 1,22. Если же группа очень хорошо разбирается в предметной области, оценка вместо этого умножается на 0,81.

82 Глава 5 • Факторы, влияющие на оценку

Таблица 5.4. Регулирующие факторы Сосото II

Фактор	Рейтинги						
	Очень низкие	Низкие	Номинальные	Высокие	Очень высокие	Чрезвычайно высокие	Влияние
Опыт работы в прикладной области	1,22	1,10	1,00	0,8	0,81		1,51
Размер базы данных		0,90	1,00	1,14	1,28		1,42
Разработка для повторного использования	0,95	1,00		1,07	1,15	1,24	1,31
Объем необходимой документации	0,81	0,91	1,00	1,11	1,23		1,52
Навыки владения языками и инструментарием	1,20	1,09	1,00	0,91	0,84		1,43
Рассредоточенная разработка	1,22	1,09	1,00	0,93	0,86	0,78	1,56
Постоянство персонала (текущесть)	1,29	1,12	1,00	0,90	0,81		1,59
Опыт разработки для платформы	1,19	1,09	1,00	0,91	0,85		1,40
Неустойчивость платформы		0,87	1,00	1,15	1,30		1,49
Сложность продукта	0,73	0,87	1,00	1,17	1,34	1,74	2,38
Квалификация программистов (общая)	1,34	1,15	1,00	0,88	0,76		1,76
Требуемая надежность программного обеспечения	0,82	0,92	1,00	1,10	1,26		1,54
Квалификация аналитиков по требованиям	1,42	1,19	1,00	0,85	0,71		2,00
Ограничения по объему хранимых данных		1,00		1,05	1,17	1,46	1,46
Ограничения по быстродействию		1,00		1,11	1,29	1,63	1,63
Использование программных инструментов	1,17	1,09	1,00	0,90	0,78		1,50

with Сосото II (Voss, 2000).

В крайнем правом столбце «Влияние» показана степень влияния каждого отдельно взятого фактора на общую оценку рабочего времени. Скажем, у фактора квалификации в предметной области в этом столбце стоит значение 1,51; это означает, что проект, выполняемый группой с очень низкой квалификацией в этой области, потребует в 1,51 раза больший объем работы, чем у группы с очень высокой квалификацией (влияние вычисляется делением наибольшего значения на наименьшее — например, $1,51 = 1,22/0,8$).

На рис. 5.7 изображено другое представление факторов Сосомо II; факторы упорядочиваются по убыванию влияния.

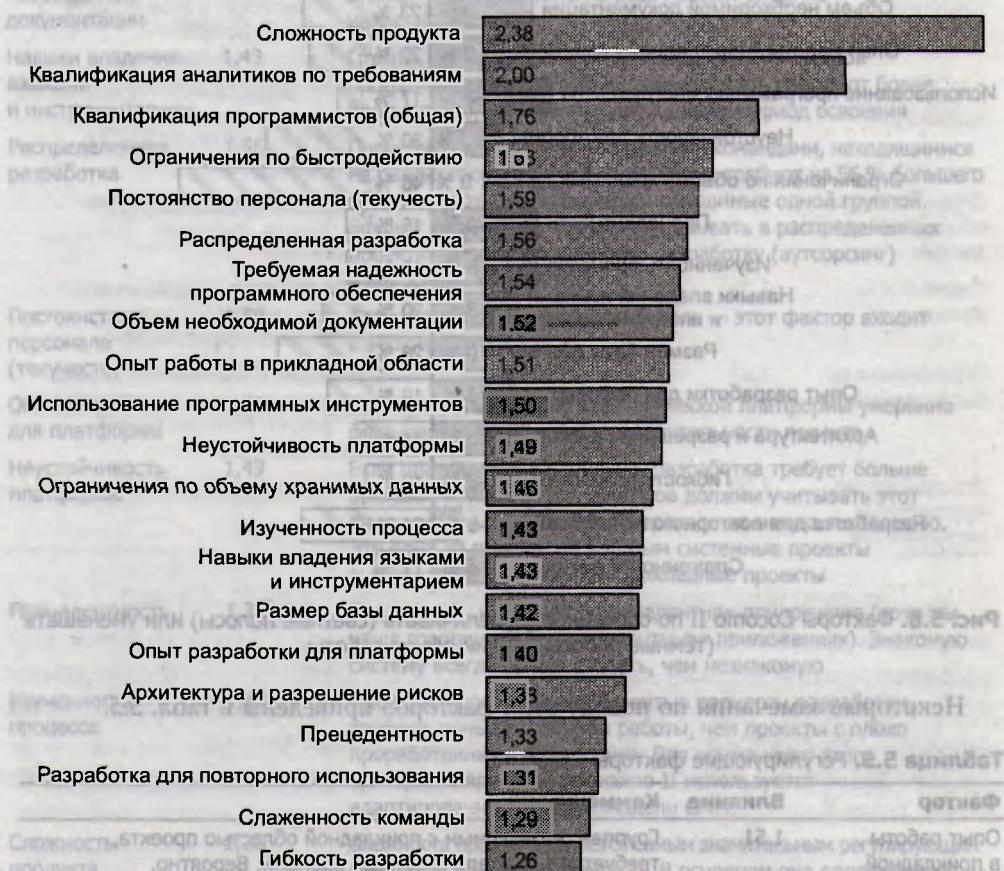


Рис. 5.7. Факторы Сосомо II по убыванию значимости. Относительная длина полос представляет чувствительность оценки к различным факторам

На рис. 5.8 те же факторы представлены в зависимости от своего потенциала к увеличению (светлые полосы) или снижению (темные полосы) общего объема работы.

84 Глава 5 • Факторы, влияющие на оценку

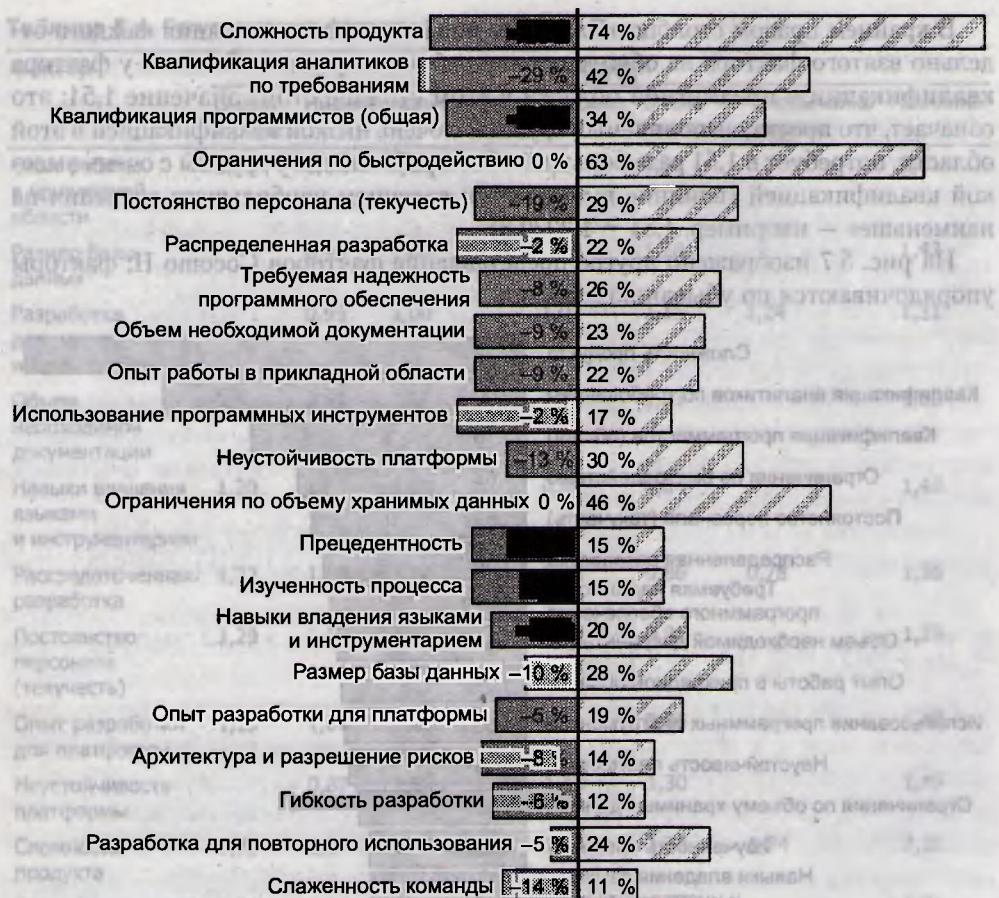


Рис. 5.8. Факторы Сосомо II по способности увеличивать (светлые полосы) или уменьшать (темные полосы) общий объем работы

Некоторые замечания по поводу этих факторов приведены в табл. 5.5.

Таблица 5.5. Регулирующие факторы Сосомо II

Фактор	Влияние	Комментарий
Опыт работы в прикладной области	1,51	Группам, незнакомым с прикладной областью проекта, требуется значительно больше времени. Вероятно, это никого не удивит
Архитектура и разрешение рисков	1,38 ¹	Чем активнее проект расправляется с возможными рисками, тем ниже будут затраты и объем работ. Это один из немногочисленных факторов Сосомо, находящихся под контролем руководителя проекта

¹ Точный эффект зависит от размера проекта. Приведенное значение соответствует размеру проекта около 100 000 строк. Эти факторы рассматриваются в следующем разделе.

Фактор	Влияние	Комментарий
Размер базы данных	1,42	Большие, сложные базы данных требуют больших усилий на уровне проекта. Общее влияние — умеренное
Разработка для повторного использования	1,31	Затраты на программное обеспечение, разрабатываемое с расчетом на повторное использование в будущем, могут увеличиваться до 31 %. Более того, успех инициативы не гарантирован — опыт отрасли показывает, что перспективные проекты повторного использования часто завершаются неудачей
Объем необходимой документации	1,31	Слишком большое количество документации может отрицательно повлиять на весь проект. Влияние — умеренно высокое
Навыки владения языками и инструментарием	1,43	Группы, обладающие опытом использования языков программирования и/или инструментария, работают более продуктивно, чем группы, проходящие период освоения
Распределенная разработка	1,56	Проекты, выполняемые несколькими командами, находящимися на разных географических площадках, требуют на 56 % большего объема работы, чем проекты, проводимые одной группой. Эффект необходимо серьезно учитывать в распределенных проектах, включая удаленную разработку (аутсорсинг) и офшорные проекты
Постоянство персонала (текучесть)	1,59	Текущесть кадров обходится дорого — этот фактор входит в верхнюю треть
Опыт разработки для платформы	1,40	Опыт разработки для технологической платформы умеренно оказывается на общей производительности проекта
Неустойчивость платформы	1,49	Если платформа нестабильна, разработка требует больше времени. Руководители проектов должны учитывать этот фактор в своих решениях о переходе на новую технологию. Это одна из причин, по которым системные проекты выполняются дольше, чем прикладные проекты
Прецедентность	1,33 ¹	Показывает, насколько «прецедентно» приложение (хотя мы чаще говорим о «беспрецедентных» приложениях). Знакомую систему всегда проще создать, чем незнакомую
Изученность процесса	1,43 ¹	Проекты, использующие развитые процессы разработки, требуют меньшего объема работы, чем проекты с плохо проработанными процессами. Для применения этого критерия к проектам в Cocomo II используется адаптированный вариант модели СММ
Сложность продукта	2,38	Сложность продукта является самым значительным регулирующим фактором в модели Cocomo II. В основном она определяется типом создаваемой программы
Квалификация программистов (общая)	1,76	Квалификация программистов способна изменить общие результаты проекта примерно в 2 раза

продолжение ↗

¹ Точный эффект зависит от размера проекта. Приведенное значение соответствует размеру проекта около 100 000 строк. Эти факторы рассматриваются в следующем разделе.

Таблица 5.5 (продолжение)

Фактор	Влияние	Комментарий
Необходимая надежность	1,54	На построение более надежных систем уходит больше времени. Это одна из причин (хотя и не единственная), по которым встроенные и критические системы требуют большего объема работ, чем другие проекты аналогичного размера. В большинстве случаев надежность программного продукта определяется рынком, и у вас нет сколько-нибудь ощущимых возможностей для ее изменения
Квалификация аналитиков по требованиям	2,00	Важнейший из факторов персонала — хорошие навыки в постановке требований — способен изменить объем работы по всему проекту в 2 раза. Компетентность в этой области способна снизить общий объем работы по проекту от номинала более, чем какой-либо другой фактор
Гибкость требований	1,26 ¹	Проекты, представляющие группе разработчиков определенную гибкость в интерпретации требований, требуют меньшего объема работы, чем проекты, настаивающие на жесткой, буквальной интерпретации всех требований
Ограничения по объему хранимых данных	1,46	Работа на платформе, на которой приходится постоянно сталкиваться с ограничениями объема, несколько увеличивает объем работы по проекту
Слаженность команды	1,29 ¹	Группы с высоким уровнем сотрудничества разрабатывают программы более эффективно, чем группы, в которых взаимодействия часто сопровождаются разногласиями
Ограничения по быстродействию	1,63	Снижение времени отклика приводит к увеличению объема работ. Это одна из причин, по которым системные проекты и проекты реального времени требуют больших усилий, чем другие проекты аналогичного размера
Использование программных инструментов	1,50	Выбор современного инструментария способен заметно снизить объем работ

Как я уже намекал ранее, изучение регулирующих факторов Сосомо II для получения представления о сильных и слабых сторонах проекта является деятельности относительно высокого уровня. Для самой оценки гораздо проще и точнее воспользоваться историческими данными из прошлых или текущих проектов, чем возиться с 22 регулирующими факторами Сосомо.

Процесс сбора и использования исторических данных подробно рассматривается в главе 8.

5.6. Снова об издержках масштаба

Регулирующие факторы Сосомо II позволяют взглянуть на издержки масштаба с новой интересной точки зрения. Пять из факторов, изображенных на рис. 5.9, называются *масштабными факторами* — это те факторы, которые вносят свой вклад

¹ Точный эффект зависит от размера проекта. Приведенное значение соответствует размеру проекта около 100 000 строк. Эти факторы рассматриваются в следующем разделе.

в издержки масштаба программного проекта. Они в разной степени влияют на проекты разных размеров. На рис. 5.9 масштабные факторы выделены светлым.

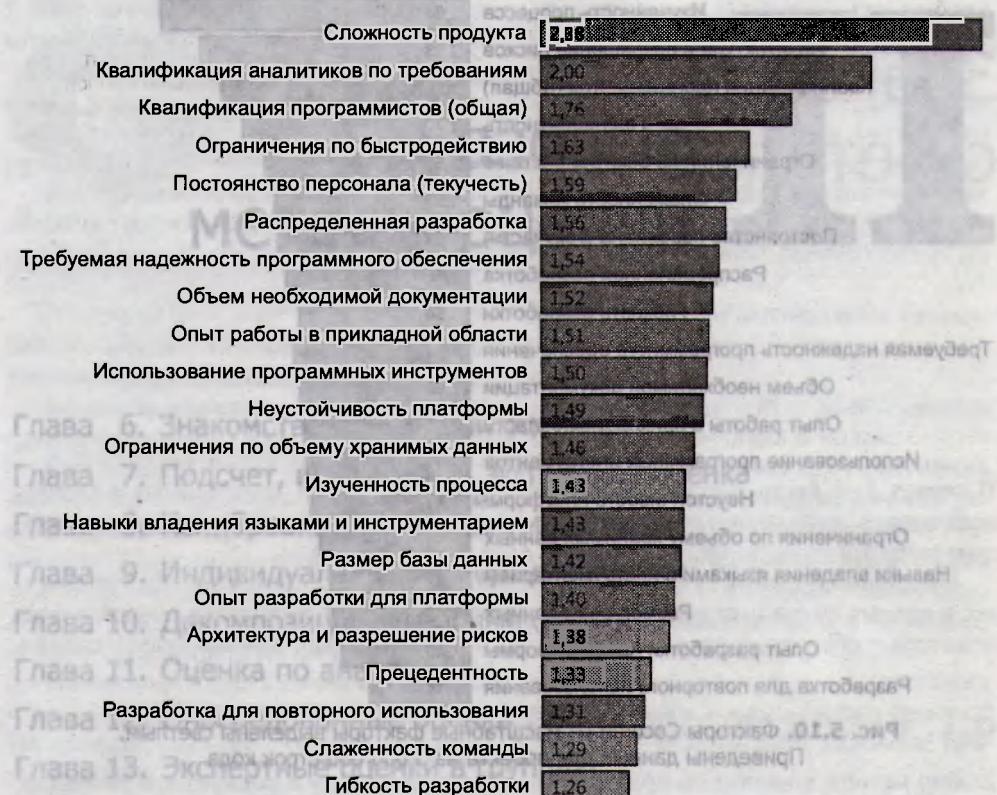


Рис. 5.9. Факторы Сосомо II; масштабные факторы выделены светлым.

Приведены данные для проекта на 100 000 строк кода

Ни один из факторов, вносящих вклад в издержки масштаба проекта, не находится в верхней половине факторов, упорядоченных по значимости. Более того, 4 из 5 наименее значимых факторов являются масштабными. Тем не менее, поскольку масштабные факторы вносят разный вклад в зависимости от размера проекта, диаграмма должна составляться для проекта конкретного размера. Данные на рис. 5.9 приведены для проекта на 100 000 строк кода. На рис. 5.10 показано, что происходит при пересчете факторов для гораздо большего проекта на 5 000 000 строк кода.

С точки зрения оценки это означает, что в проектах разных размеров факторам должны присваиваться разные весовые коэффициенты. С точки зрения планирования и управления проектами это означает, что успех мелких и средних проектов сильно зависит от качества персонала. В крупных проектах сильные личности по-прежнему необходимы, но не менее значительную роль играет и качество управления проектом (особенно в отношении управления рисками, «зрелость» организаций и то, насколько слаженно отдельные личности работают в составе группы).

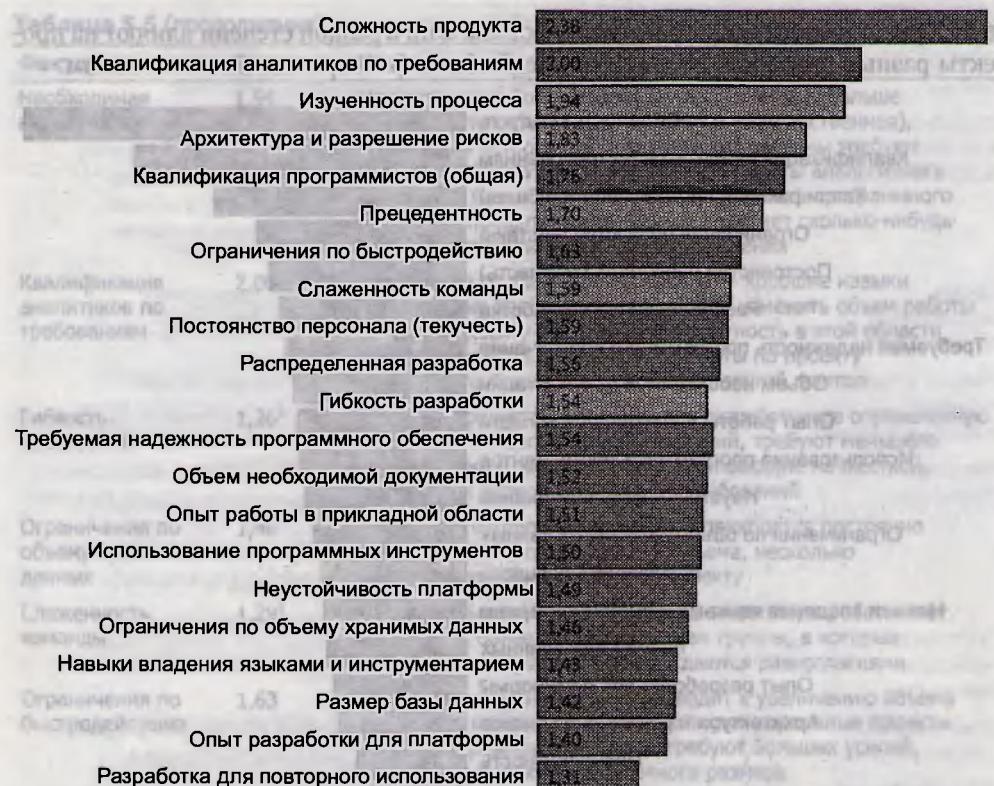


Рис. 5.10. Факторы Cocomo II; масштабные факторы выделены светлым.

Приведены данные для проекта на 5 000 000 строк кода

Дополнительные источники

Boehm, Barry, et al. «Software Cost Estimation with Cocomo II». Reading, MA: Addison-Wesley, 2000. Книга содержит наиболее авторитетное описание Cocomo II. Ее толщина внушает страх, однако базовая модель Сокомо описывается на первых 80 страницах, вместе с подробными определениями множителей объема работы и масштабных факторов и учетом издержек масштаба в Cocomo II. В оставшейся части книги рассматриваются расширения базовой модели.

Putnam, Lawrence H. and Ware Myers. «Measures for Excellence: Reliable Software On Time, Within Budget». Englewood Cliffs, NJ: Yourdon Press, 1992. В книге описан метод оценки Путнэма и способы учета издержек масштаба. Мне нравится модель Путнэма, потому что она содержит малое количество регуляторов и лучше всего работает при калибровке с использованием исторических данных. Материал сильно насыщен математическими формулами, поэтому чтение идет довольно медленно.

Размер проекта

Размер проекта – еще один фактор, который необходимо учитывать при выборе методов оценки.

Самые простые. Я отношу к этой категории проекты, в которых участвует всего один человек, но это весьма полная оценка. Такие методы лучше для больших проектов, в которых оценка продолжительности может быть выполнена с помощью специальных календарных методов.

Как правило, в малых проектах используются простые методы оценки, достаточные для больших проектов, становятся несоставительными.

Лучшими методами оценки для малых проектов обычно оказываются «восходящие» методы, основанные на опытах людей, которые будут непосредственно заниматься выполнением работы.

Базовые методы оценки

- Глава 6. Знакомство с методами оценки
- Глава 7. Подсчет, вычисление, экспертная оценка
- Глава 8. Калибровка и исторические данные
- Глава 9. Индивидуальные экспертные оценки
- Глава 10. Декомпозиция и сводные оценки
- Глава 11. Оценка по аналогии
- Глава 12. Опосредованные оценки
- Глава 13. Экспертные оценки в группах
- Глава 14. Оценочные программы
- Глава 15. Использование альтернативных оценок
- Глава 16. Формирование оценок в правильно оцениваемых проектах
- Глава 17. Стандартизованные процедуры оценки

Знакомство с методами оценки

6

В главах 1–5 были рассмотрены важнейшие концепции, заложенные в основу всех оценок программных проектов. Настало время перейти к подробному описанию методов оценки, применяемых к конкретным задачам.

При использовании этих методов следует учитывать одно важное обстоятельство: в разных ситуациях применяются разные оценки. В этой главе представлены некоторые обстоятельства, которые должны учитываться при выборе методики.

6.1. Выбор методов оценки

Выбор метода, способного принести наибольшую пользу в конкретной ситуации, определяется как стремлением учесть факторы, описанные в главе 5, так и стремлением обойти источники ошибок оценки, описанных в главе 4. Далее описываются основные аспекты, которые следует принять во внимание.

Что именно оценивается

Одни проекты начинаются с определения функциональности, а затем переходят к оценке сроков и объема работы, необходимые для ее реализации. Другие проекты определяют свои бюджеты и временные рамки разработки, после чего выясняется, сколько функций можно реализовать за этот срок.

Многие методы оценки работают независимо от того, что именно оценивается; некоторые методы лучше подходят для оценки объема работ, продолжительности или количества функций.

В этой книге под оценкой *размера* понимается оценка всей области технической работы для реализации заданной функциональности — в таких единицах, как сроки программного кода, функциональные пункты и т. д. Оценкой *функциональности* мы будем называть оценку количества функций, реализуемых в ограничениях срока и бюджета. Эти термины не являются стандартными; я определяю их ради большей ясности.

Размер проекта

Размер проекта — еще один фактор, который необходимо учитывать при выборе методики.

Малые проекты. Я отношу к этой категории проекты с пятью и менее техническими участниками, но это весьма вольная оценка. Статистические методы, подходящие для больших проектов, в малых проектах обычно неприменимы, потому что различия в производительности отдельных участников затмевают другие факторы.

Как правило, в малых проектах используется плоская модель комплектования кадрами (на протяжении всего проекта численность персонала остается неизменной), из-за чего некоторые алгоритмические методы оценки, рассчитанные на большие проекты, становятся несостоятельными.

Лучшими методами оценки для малых проектов обычно оказываются «нисходящие» методы, основанные на оценках людей, которые будут непосредственно заниматься выполнением работы.

Большие проекты. К большим проектам относятся проекты, выполняемые группами около 25 участников, занимающие от 6 до 12 месяцев и более.

Оптимальный выбор методики оценки для большого проекта существенно изменяется в зависимости от состояния проекта. На ранних стадиях лучший результат обычно дают «нисходящие» методы, основанные на алгоритмах и статистике. Они состоятельны на той стадии проекта, когда конкретный состав участников еще не известен — когда планы основываются на группах, состоящих, допустим, из «11 ведущих инженеров, 25 рядовых разработчиков и 8 тестеров», нежели на конкретных личностях.

На средней стадии более точную оценку обеспечивает сочетание нисходящих и восходящих методов, базирующихся на исторических данных самого проекта.

На поздних стадиях крупных проектов восходящие методы дают наиболее точную оценку.

Средние проекты. К этой категории относятся проекты, выполняемые 5–25 участниками, занимающие от 3 до 12 месяцев. К преимуществам средних проектов можно отнести возможность применения практически всех методов оценки, применимых в крупных проектах, а также ряда методик малых проектов.

Стиль разработки

В контексте оценки выделяются два основных стиля разработки: *последовательный* и *итеративный*. Отраслевая терминология, окружающая итеративные, последовательные и динамические проекты, довольно сложна. Для наших целей основным различием между этими типами проектов является процент требований, определяемых на ранней стадии проекта, по сравнению с процентом требований, определяемых в ходе работы.

Далее представлены некоторые подходы к разработке, выделенные по указанным критериям.

Эволюционное макетирование используется в тех случаях, когда требования неизвестны, а одна из главных причин для применения этой методики — содействие в определении требований (McConnell 1996). В контексте оценки эволюционное макетирование относится к итеративному стилю разработки.

Экстремальное программирование намеренно ограничивается определением только тех требований, которые будут реализовываться при следующей итерации, обычно занимающей менее одного месяца (Beck 2004). В контексте оценки экстремальное программирование относится к высокоитеративному стилю.

Эволюционная выдача. В проектах с эволюционной выдачей доля изначально определяемых требований изменяется от «почти отсутствует» до «большинства» (Gilb 1988, McConnell 1996). В зависимости от того, к какому концу шкалы относится конкретный проект, проекты с эволюционной выдачей могут быть как последовательными, так и итеративными. Как правило, проекты с эволюционной выдачей оставляют достаточно большое количество требований неопределенными на момент начала разработки, чтобы разработку можно было отнести к итеративной.

Поэтапная выдача. В проектах с поэтапной выдачей основные требования определяются до начала основной работы над проектом (McConnell 1996). Поэтапная выдача использует итеративный подход к проектированию, конструированию и тестированию и поэтому в некотором смысле является итеративной. Тем не менее в контексте оценки ее следует отнести к последовательному стилю разработки.

Унифицированный процесс Rational. Стадии процесса RUP (Rational Unified Process) называются «итерациями». Тем не менее в типичном проекте RUP около 80 % требований должны определяться до начала разработки (Jacobson, Booch, and Rumbaugh 1999). В контексте оценки RUP относится к последовательному стилю разработки.

Scrum — стиль разработки, при котором рабочая группа проекта выбирает набор возможностей, которые она может реализовать в течение 30-дневного «броска» (Schwaber and Beedle 2002). После того как «бросок» начался, клиенту не разрешается изменять требования. Если рассматривать отдельные броски, в контексте оценки Scrum относится к последовательному стилю. Но поскольку функциональность не распределяется более чем по одному броску, с учетом множественных итераций Scrum относится к итеративному стилю.

Влияние стиля разработки на выбор методов оценки

Как итеративные, так и последовательные проекты обычно начинаются с нисходящих (основанных на статистике) методов и постепенно мигрируют к восходящим методам. Итеративные проекты быстрее переходят к уточнению своих оценок с использованием данных самого проекта.

Стадия разработки

По мере того как группа работает над проектом, накапливается информация, позволяющая создать более точную оценку. Требования постепенно становятся более понятными, архитектура — более подробной, планы — более стабильными, а сам проект выдает данные производительности, которые могут использоваться для оценки оставшейся части проекта.

В книге стадии разработки определяются следующим образом.

Ранняя стадия. В последовательных проектах к ранней стадии относится период от начала построения концепции проекта и до того момента, когда требования

в целом можно считать определенными. В итеративных проектах термином «ранняя стадия разработки» обозначается период исходного планирования.

Средняя стадия. Средней стадией называется период времени между начальным планированием и ранним конструированием. В последовательных проектах средняя стадия продолжается от этапа постановки требований и архитектуры до момента, когда проект будет в достаточной степени проработан для получения данных производительности, на основе которых может быть составлена оценка. В итеративных проектах под средней стадией понимаются первые две-четыре итерации, происходящие до того, как в основу оценок проекта можно будет уверенно заложить его собственные данные производительности.

Поздняя стадия. Термином «поздняя стадия» обычно обозначается время от середины разработки до выпуска.

Некоторые методы лучше всего работают в широкой части конуса неопределенности. Другие обеспечивают лучшие результаты после того, как в ходе проекта будут сгенерированы данные, которые могут использоваться для оценки оставшейся части проекта.

Область применения методов этой главы

Возможная точность

Точность методики отчасти зависит как от самой оценки, так и от того, насколько правильно выбрана методика для конкретной проблемы, и от специфики проекта.

Некоторые методы оценки обеспечивают высокую точность ценой высоких затрат. Другие обеспечивают более низкую точность при меньших затратах. Как правило, желательно использовать наиболее точные методы, однако в зависимости от текущего состояния проекта и точности, возможной в текущей точке конуса неопределенности, метод с низкой точностью при низких затратах может оказаться более уместным.

6.2. Таблицы применимости методов

Большинство остальных глав книги начинается с таблиц, описывающих область применения метода, представленного в данной главе. Пример:

Область применения методов этой главы — ПРИМЕР

	Групповой анализ	Калибровка данными проекта
Что оценивается	Размер, объем работ, сроки, функциональность	Размер, объем работ, сроки, функциональность
Размер проекта	- СБ	МСБ
Стадия разработки	Ранняя–средняя	Средняя–поздняя
Итеративный или последовательный стиль	Оба	Оба
Возможная точность	Средняя–высокая	Высокая

Приведенные в таблице данные основаны на сведениях, приведенных в предыдущем разделе. Смысль содержимого таких таблиц объясняется в табл. 6.1.

Таблица 6.1. Содержимое таблиц «Область применения методов этой главы»

Строка таблицы	Допустимые значения
Что оценивается	Размер, объем работ, сроки, функциональность
Размер проекта	М С Б (малый, средний, большой)
Стадия разработки	Ранняя, средняя, поздняя
Итеративный или последовательный стиль	Итеративный, последовательный, оба
Возможная точность	Низкая, средняя, высокая

СОВЕТ № 29

При выборе метода оценки следует учитывать оцениваемый показатель, размер проекта, стадию разработки, стиль разработки и требуемую точность.

Унифицированный процесс Rational, Стадии процесса КМП (National Computer Process) называются «итеративик». Тем не менее в данном процессе 80–85% требований должны определяться до начала разработки (Gosselin, 1999; и др., 1999). В контексте оценки ВПР это означает, что в начале разработки должны быть определены требования к функциям, а также определены критерии оценки. В дальнейшем разработка может вестись в итеративном режиме, когда в ходе разработки появляются новые требования и корректируются существующие. Важно помнить, что итеративные методы оценки не являются альтернативой классическим методам, а являются их продолжением. Итеративные методы оценки не могут заменить классические методы, но могут дополнить их. Точность оценки не распределяется более чем по одному борю, а между методами оценки, применяемыми на различных итерациях. Самые отдаленные итерации относятся к итеративному стилю.

Влияние стиля разработки на выбор методов оценки

Как итеративные, так и последовательные проекты обычно начинаются с низкоточечной оценкой, которая в дальнейшем уточняется с помощью различных методов. Итеративные методы оценки позволяют уточнение оценки с использованием данных самого проекта.

Стадия разработки	Критерии	Логика оценки
Последовательная	М С Б	Построение модели для каждого блока, в результате чего получается полная модель проекта.
Итеративная	М С Б	Построение модели для каждого блока, в результате чего получается полная модель проекта.

7

Подсчет, вычисление, экспертная оценка

— это общая национальная и региональная система оценки и аудита интегральных пункты — все это признаки обобщения от 1 до 100 единиц, соединенных с итоговым размером системы.

В разных средах наиболее точными показателями размера проекта являются разные величины. В одной среде лучший показатель может быть количество веб-страниц, в другой — количество маркетинговых требований, в третьем — количество языков, четвертом — количество концепций. Поэтому чтобы подсчитать ходящий показатель размера можно использовать следующие формулы:

Область применения методов этой главы

	Счетные методы	Вычислительные методы
Что оценивается	Размер, функциональность	Размер, объем работ, сроки, готовность, функциональность
Размер проекта	МСБ	МСБ
Стадия разработки	Ранняя–поздняя	Ранняя–средняя
Итеративный или последовательный стиль	Оба	Оба
Возможная точность	Высокая	Высокая

Представьте, что вы попали на конференцию лучших оценщиков. Зал заполнен людьми; вы сидите за столом в середине комнаты с тремя другими специалистами. Куда ни глянь, кругом одни оценщики. Внезапно распорядитель подходит к микрофону и говорит: «Нам нужно точно знать, сколько людей находится в зале, чтобы заказать десерт. Кто даст самую точную оценку количества собравшихся?»

За столом немедленно завязывается оживленная дискуссия по поводу того, как лучше всего получить оценку. Билл, ваш сосед справа, говорит: «Оценка размера толпы — мое хобби. На основании своего опыта могу сказать, что в комнате находится 335 людей».

Сосед напротив, Карл, говорит: «Столы расставлены равномерно, 11 в длину и 7 в ширину. Организатор банкета — мой хороший знакомый — сказал, что они собираются усадить по 5 людей за один стол. Кажется, за большинством столов сейчас действительно сидит по 5 человек. Умножаем 11 на 7, затем умножаем на 5 и получаем 385 человек. Пожалуй, это значение можно использовать как оценку».

Сосед слева, Люси, говорит: «При входе я заметила табличку, на которой написано, что зал рассчитан на 485 человек. Сейчас зал заполнен процентов на 70–80. Если умножить проценты на максимальную емкость, мы получаем от 340 до 388 человек. Давайте используем среднее — 364 человека... или 365 для надежности?»

Билл: «Итак, мы имеем оценки 335, 365 и 385. Похоже, истина где-то посередине. Согласен на 365».

«Я тоже», — говорит Карл.

Все смотрят на вас. Вы говорите: «Мне нужно кое-что проверить. Я ненадолго отлучусь, вы не возражаете?» Соседи удивлены, но не возражают.

Вы возвращаетесь через пару минут. «Помните, при входе наши приглашения обрабатывались на сканере? Я заметил, что на сканере есть счетчик, поэтому подошел к контролеру у входа. Контролер говорит, что было отсканировано 407 билетов, и из зала еще никто не выходил. Полагаю, в качестве оценки нужно использовать число 407. Что скажете?»

Возможная точность

Низкая, средняя, высокая

7.1. Сначала подсчет

Как вы думаете, каков правильный ответ? 335, как предложил Билл, эксперт по оценке размера толпы? 385, как вычислил Карл по нескольким разумным предположениям? 365, как вычислила Люси по другим разумным предположениям? Или правильным было число 407, отображающееся на сканере приглашений? Есть ли какие-нибудь сомнения относительно того, что 407 является наиболее точным ответом? Кстати, вся история кончилась тем, что ваш стол предложил число 407, которое оказалось правильным, и первым получил десерт.

Один из секретов этой книги состоит в том, что вы должны по возможности избегать того, что мы обычно называем оценкой! Если ответ можно просто *посчитать*, действуйте именно так. В нашей истории этот способ дал наиболее точный ответ.

Если прямой подсчет невозможен, нужно посчитать что-нибудь другое и *вычислить* ответ по вспомогательным (калибровочным) данным. Скажем, Карлу было известно, что на банкете за каждым столом должно было сидеть 5 человек. Он *подсчитал* количество столов и вычислил ответ по этому значению.

Люси действовала сходным образом: ее оценка основывалась на документированной вместительности зала. На основании своего *экспертного суждения* она прикинула, что зал заполнен на 70–80 % процентов.

Наименее точная оценка поступила от Билла, который руководствовался только своим *экспертным суждением* для получения ответа.

СОВЕТ № 30

Считайте везде, где это возможно. Если счет невозможен — вычисляйте. Используйте оценки, полученные на основании одного лишь экспертного суждения, только в крайнем случае.

7.2. Что считать

Программные проекты порождают многочисленные показатели, которые могут использоваться в подсчетах. На ранней стадии цикла разработки можно подсчитывать маркетинговые требования, функции, сценарии использования и т. д.

В середине работы над проектом подсчет может вестись с большей глубиной детализации. Вот лишь несколько примеров: инженерные требования, функциональные пункты, запросы на внесения изменений, веб-страницы, отчеты, диалоговые окна, экраны, таблицы базы данных и т. д.

На более поздней стадии проекта детализация становится еще большей: объем уже написанного кода, количество выявленных дефектов, классы и задачи, а также уточнения всех показателей, которые подчтывались ранее в проекте.

Выбор показателя для подсчета определяется несколькими целями.

Ищите счетный показатель, высоко коррелированный с размером оцениваемого проекта. Если вы оцениваете затраты и сроки при фиксированной функциональности, наибольшее влияние на оценку проекта оказывает его размер. Ищите показатели, убедительно обозначающие размер программного проекта. Количество маркетинговых требований, количество инженерных требований, функциональные пункты — все это примеры счетных показателей, тесно связанных с итоговым размером системы.

В разных средах наиболее точными показателями размера проекта оказываются разные величины. В одной среде лучшим показателем может быть количество веб-страниц; в другой — количество маркетинговых требований, тестовых сценариев или параметров конфигурации. Трудность в том, чтобы подобрать хороший показатель размера именно для вашей среды.

СОВЕТ № 31

Найдите счетный показатель, который может использоваться для осмысленного измерения содержания работы в вашей среде.

Когда ищите счетный показатель, доступный на ранней, а не на поздней стадии цикла разработки. Чем скорее вы найдете осмысленный показатель для подсчета, тем скорее сможете обеспечить долгосрочную прогнозируемость. Количество строк программного кода в проекте часто является отличным показателем объема работы, однако этот показатель становится доступным лишь после завершения проекта. Функциональные пункты тесно связаны с окончательным размером проекта, но они остаются недоступными вплоть до появления подробных требований. Если вам удастся найти счетный показатель, доступный на более ранней стадии, используйте его для создания более ранней оценки. Например, можно создать грубую оценку на основании числа маркетинговых требований, а затем уточнить ее позднее по количеству функциональных пунктов.

Ищите счетный показатель, дающий статистически осмысленное среднее значение. С точки зрения статистики для получения осмысленного среднего значения выборка должна содержать не менее 20 элементов. Здесь 20 — не волшебное число, а всего лишь хорошая рекомендация для статической состоятельности.

Понимайте, что вы считаете. Чтобы подсчет мог послужить точной основой для оценки, необходимо проследить за тем, чтобы на них распространялись те же предположения, что и для исторических данных, используемых в оценке. Если вы считаете маркетинговые требования, убедитесь в том, что понимание «маркетинговых требований», действовавшее в отношении исторических данных, совпадает с пониманием «маркетинговых требований» в вашей оценки. Если исторические данные показывают, что в предыдущем проекте группа обрабатывала 7 неформальных требований заказчика (также называемых «историями пользователей», user story) в неделю, проследите за тем, что ваши предположения относительно размера группы, опыта программистов, технологии разработки и других факторов аналогичны предположениям в оцениваемом проекте.

Найдите показатель, подсчитываемый с минимальными усилиями. При прочих равных обстоятельствах предпочтение отдается показателям, подсчет которых требует минимальных усилий. В сценарии, описанном в начале главы, количество людей в комнате уже было зафиксировано на сканере. Если бы вам пришлось обходить все столы и считать присутствующих вручную, возможно, вы решили бы, что игра не стоит свеч.

Один из выводов проекта Cocomo II заключается в том, что метрика оценки размера, называемая объектными пунктами, в такой же степени коррелируется с объемом работ, что и функциональные пункты, но ее вычисление требует примерно половинного объема работы. По этой причине объектные пункты рассматриваются как эффективная альтернатива для функциональных пунктов при проведении оценки в широкой части конуса неопределенности (Boehm et al. 2000).

7.3. Используйте вычисления для преобразования счетных показателей в оценки

Собранные исторические данные, относящиеся к счетным показателям, преобразуются в нечто более полезное — например, оценку объема работы. В табл. 7.1 приведены примеры счетных показателей и данных, необходимых для вычисления по ним оценки.

Таблица 7.1. Примеры счетных показателей

Счетный показатель	Исторические данные, необходимые для преобразования результатов в оценку
Маркетинговые требования	<ul style="list-style-type: none"> Среднее количество рабочих часов на требование при разработке Среднее количество рабочих часов на требование при независимом тестировании Среднее количество рабочих часов на требование при документировании Среднее количество рабочих часов на требование при формировании технических требований по маркетинговым требованиям
Функциональность	<ul style="list-style-type: none"> Среднее количество рабочих часов на функцию при разработке и/или тестировании
Сценарии использования	<ul style="list-style-type: none"> Среднее общее количество рабочих часов на сценарий Среднее количество сценариев, реализуемых за определенный календарный период
Истории пользователя	<ul style="list-style-type: none"> Среднее количество рабочих часов на требование Среднее количество историй, реализуемых за определенный календарный период
Технические требования	<ul style="list-style-type: none"> Среднее количество технических требований, формально анализируемых за час Среднее количество рабочих часов на требование при разработке/тестировании/документировании

Счетный показатель	Исторические данные, необходимые для преобразования результатов в оценку
Функциональные пункты	<ul style="list-style-type: none"> Средний объем работы по разработке/тестированию/документированию на функциональный пункт Среднее количество строк программного кода используемого языка программирования на функциональный пункт
Запросы на внесение изменений	<ul style="list-style-type: none"> Средний объем работы по разработке/тестированию/документированию на запрос (в зависимости от разнообразия изменений данные могут подвергаться дополнительному разбиению на средний объем работы для малых, средних и крупных запросов)
Веб-страницы	<ul style="list-style-type: none"> Средний объем работы на веб-страницу для обеспечения работоспособного интерфейса Средний объем работы на веб-страницу в масштабе всего проекта (менее надежно, но может содержать интересную информацию)
Отчеты	<ul style="list-style-type: none"> Средний объем работы для получения работоспособного отчета
Диалоговые окна	<ul style="list-style-type: none"> Средний объем работы на диалоговое окно для обеспечения работоспособного интерфейса
Таблицы баз данных	<ul style="list-style-type: none"> Средний объем работы на таблицу для получения работоспособной базы данных
Классы	<ul style="list-style-type: none"> Среднее количество рабочих часов на класс при разработке Среднее количество рабочих часов для формального анализа класса Среднее количество рабочих часов для тестирования класса
Найденные дефекты	<ul style="list-style-type: none"> Среднее количество рабочих часов на дефект для исправления Среднее количество рабочих часов на дефект для регрессионного тестирования Среднее количество дефектов, исправляемых за определенный календарный период
Параметры конфигурации	<ul style="list-style-type: none"> Средний объем работы на параметр
Количество строк уже написанного кода	<ul style="list-style-type: none"> Среднее количество дефектов на строку кода Среднее количество строк кода, которые могут быть формально проанализированы за час Среднее количество новых строк кода при переходе к следующей версии
Тестовые сценарии (уже написанные)	<ul style="list-style-type: none"> Средний объем работы на тестовый сценарий на стадии выпуска

СОВЕТ № 32

Соберите исторические данные, которые позволят вам вычислить оценку по счетным показателям.

Пример подсчета дефектов на поздней стадии проекта. Данные, описанные в таблице, закладывают более прочную основу для создания оценок, нежели экспертное суждение. Если вы знаете, что проект содержит 400 открытых дефектов, а при исправлении 250 дефектов, обнаруженных ранее, потребовалось по 2 часа

на дефект, значит, на исправление всех открытых дефектов потребуется примерно $400 \times 2 = 800$ часов.

Пример оценки подсчетом веб-страниц. Если данные указывают на то, что до настоящего момента на проектирование, кодирование и тестирование каждой веб-страницы с динамическим содержимым ушло около 40 часов, а в проекте осталось еще 12 веб-страниц, можно сделать вывод, что работа над ними займет $12 \times 40 = 480$ часов.

В этих примерах важно то, что оценки не строятся на *экспертном суждении*. Сначала мы считаем значение некоторого показателя, а затем вычисляем по нему оценку. Этот процесс помогает избавить оценки от субъективного смещения, отрицательно сказывающегося на их точности. Для уже известных счетных показателей (скажем, количества дефектов) построение таких оценок также требует минимальных усилий.

СОВЕТ № 33

Не пренебрегайте возможностями простых, грубых оценочных моделей, таких как средний объем работы на дефект, средний объем работы на веб-страницу, средний объем работы на историю пользователя и средний объем работы на сценарий использования.

7.4. Используйте экспертное суждение только в крайнем случае

Так называемое экспертное суждение является наименее точным методом оценки. Похоже, наибольшая точность достигается тогда, когда оценка привязывается к чему-то конкретному. В истории, изложенной в начале главы, наихудшей была оценка, выданная экспертом исключительно на основе личного суждения. Привязка оценки к вместимости зала дала чуть лучший результат; тем не менее и эта оценка содержала немалую ошибку, потому что эксперту потребовалось субъективно оценить степень заполненности зала, а это открыло возможности для порчи оценки субъективными факторами.

Исторические данные в сочетании с вычислениями оказываются на удивление свободными от смещений, подрывающих оценки, основанные на экспертных мнениях. Не поддавайтесь искушению подстраивать вычисленные оценки под свое экспертное суждение. Во время работы над вторым изданием книги «Code Complete» (McConnell 2004a) я руководил группой, проводившей формальный анализ первого издания — всех 900 страниц. Во время нашей первой встречи средняя скорость анализа составила 3 минуты на страницу. Выходило, что при такой скорости на анализ всей книги уйдет 45 часов. После первой встречи я заметил, что мы еще только «срабатываемся» как единая команда, а в будущем, как мне кажется, скорость должна возрасти. При планировании будущих встреч я по рекомендовал использовать рабочую оценку в 2–2,5 минуты на страницу вместо 3 минут. Руководитель проекта ответил, что на данный момент мы располагаем данными только одной встречи, поэтому за основу нескольких будущих встреч нужно взять имеющиеся показатели, то есть 3 минуты на страницу. Позднее при

необходимости планы можно будет подрегулировать на основании данных по следующих собраний.

И как вы думаете, каким оказалось среднее время после завершения книги, всех 900 страниц? Вы не ошиблись, 3 минуты на страницу!

СОВЕТ № 34

Не используйте экспертные суждения для подгонки оценок, полученных на основании вычислений. Подобная «экспертиса» обычно лишь ухудшает точность оценки.

Дополнительные ресурсы

Boehm, Barry, et al. «Software Cost Estimation with Cocomo II». Reading, MA: Addison-Wesley, 2000. У Бема представлено краткое описание метрики объектных пунктов.

Lorenz, Mark and Jeff Kidd. «Object-Oriented Software Metrics». Upper Saddle River, NJ: PTR Prentice Hall, 1994. Лоренц и Кидд предлагают ряд количественных показателей, которые могут использоваться в объектно-ориентированных программах.

Калибровка и исторические данные



В этих примерах важно то, что оценки не строятся на экспертных суждениях. Сначала мы считаем значение некоторого показателя, а затем вычисляем по нему оценку. Этот процесс помогает избавиться от субъективности оценки, основанной на личном мнении оценщика. Это способствует объективности оценки, полученной с помощью калибровки.

Область применения методов этой главы

	Калибровка средними данными по отрасли	Калибровка историческими данными	Калибровка проектными данными
Что оценивается	Размер, объем работ, сроки, функциональность	Размер, объем работ, сроки, функциональность	Размер, объем работ, сроки, функциональность
Размер проекта	М С Б	М С Б для, когда оценка используется	М С Б
Стадия разработки	Ранняя–средняя	Ранняя–средняя	Средняя–поздняя
Итеративный или последовательный стиль	Оба	Оба	Оба
Возможная точность	Низкая–средняя	Средняя–высокая	Высокая

Калибровка используется для преобразования счетных показателей в оценки — строк кода в объем работы, историй пользователей в календарное время, требований в тестовые сценарии и т. д. Оценка всегда подразумевает некую разновидность калибровки, прямую или косвенную. Калибровка с использованием различных типов данных образует вторую часть метода «сначала подсчет, затем вычисления», описанного в главе 7.

Ваша оценка может калиброваться по трем типам данных.

- *Среднеотраслевые данные* — данные других организаций, занимающихся разработкой аналогичного программного обеспечения.
- *Исторические данные* — в книге так называются данные организации, которая занимается оцениваемым проектом.
- *Проектные данные* — данные, полученные ранее в ходе оцениваемого проекта.

Исторические и проектные данные приносят огромную пользу и позволяют существенно повышать точность оценок. Отраслевые данные — не более чем временная, резервная информация, которая может пригодиться при отсутствии исторических или проектных данных.

8.1. Улучшение точности и другие преимущества исторических данных

Основная причина для использования исторических данных вашей организации заключается в том, что они заметно повышают точность оценки. Использование исторических данных, или «документированных фактов», отрицательно коррелируется с превышением сроков и затрат — иначе говоря, для проектов, оцениваемых на основе исторических данных, превышения не характерны (Lederer and Prasad 1992).

В следующих разделах представлены некоторые причины для повышения точности оценок.

Учет организационных факторов

Прежде всего, исторические данные учитывают великое множество организационных факторов, влияющих на результат проекта. В очень малых проектах результат определяется личными качествами работников. С увеличением размера проекта талантливые личности по-прежнему важны, однако их вклад либо поддерживается, либо подрывается влиянием организационных факторов. В средних и крупных проектах организационные характеристики начинают играть не менее важную роль, чем личные качества.

Далее перечислены некоторые организационные факторы, влияющие на результат проекта.

- Насколько сложна программа, какие ограничения накладываются на ее быстродействие, какую надежность необходимо обеспечить, сколько документации требуется, существуют ли предыдущие аналоги — другими словами, какое место занимает проект в системе факторов Сосомо II, относящихся к типу разрабатываемого проекта (см. главу 5)?
- Может ли организация определиться с устойчивыми требованиями, или рабочая группа должна учитывать возможные изменения на протяжении всего проекта?
- Может ли руководитель проекта избавиться от проблемного участника, или же кадровая политика организации затрудняет (или делает невозможной) его увольнение?
- Может ли группа полностью сосредоточиться на текущем проекте, или ее участникам приходится часто отвлекаться на запросы, связанные с поддержкой предыдущих проектов?

- Может ли организация включать новых участников в проект, как было запланировано, или она откажется «выдергивать» работников из других проектов вплоть до их завершения?
- Поддерживает ли организация использование эффективных методов проектирования, конструирования, контроля качества и тестирования?
- Работает ли организация в регламентированной среде (например, под действием нормативных актов FAA или FDA), предписывающей обязательное следование некоторым правилам?
- Может ли руководитель проекта рассчитывать на то, что участники группы останутся на работе вплоть до завершения проекта, или для организации характерна высокая текучесть кадров?

Пытаться учитывать каждый из этих факторов по отдельности трудно, к тому же при этом повышается вероятность ошибки. Исторические данные вносят поправку на все эти факторы независимо от того, известны вам конкретные детали или нет.

Предотвращение субъективизма и необоснованного оптимизма

Один из путей, по которым субъективизм проникает в оценку: руководитель проекта или оценщик смотрит на новый проект, сравнивает его со старым, замечает множество различий между ними, а затем делает вывод, что новый проект пойдет лучше старого. Он говорит: «В предыдущем проекте у нас была большая текучесть кадров; на этот раз этого не будет, и наша работа станет более эффективной. К тому же нас постоянно отвлекали на поддержку предыдущей версии; мы позабочимся о том, чтобы этого не произошло. Отдел маркетинга вносил поздние изменения в требования. Мы и с этим справимся лучше. Вдобавок на этот раз мы будем использовать более современную технологию и новые, более эффективные методы разработки. При таких усовершенствованиях наша работа будет гораздо более продуктивной».

В обоснованиях такого рода легко прослеживается оптимизм. Однако перечисленные факторы находятся под контролем организации, а не конкретного руководителя проекта, поэтому большинство из них будет трудно контролировать на уровне отдельного проекта. Другие факторы подвергаются оптимистичной переоценке, из-за чего в оценку вносится смещение.

При наличии исторических данных используется упрощенное предположение о том, что следующий проект пойдет примерно так же, как прошлый проект. Вполне разумное предположение. По словам гуру в области оценки Лоренса Путнэма, производительность является атрибутом организации и не может легко изменяться между проектами (Putnam and Myers 1992, Putnam and Myers 2003). Аналогичная концепция представлена в экстремальном программировании в виде «принципа вчерашней погоды»: сегодняшняя погода не всегда будет такой же, как вчера, но она будет похожа на вчерашнюю погоду с большей вероятностью, чем на что-либо другое (Beck and Fowler 2001).

СОВЕТ № 35

Стройте свои оценки производительности на исторических данных. Производительность вашей организации в прошлом дает наилучшее представление о ее производительности в будущем.

Снижение влияния политических факторов при оценке

Одна из потенциальных проблем моделей оценки с большим количеством регулируемых факторов заключается в том, что многие регуляторы верхнего уровня относятся к персоналу. Например, модель Сосомо II требует оценки навыков аналитиков и программистов наряду с несколькими менее субъективными факторами, относящимися к опыту. Оценщик должен дать оценку программистов, выбирая между 90-й, 75-й, 55-й, 35-й и 15-й процентилю (эти значения процентиля являются общеотраслевыми).

Представьте, что руководитель проекта представляет высшему начальству оценку Сосомо II, и теперь идет собрание, в ходе которого выясняется, не заложены ли в оценке какие-либо излишества. Легко представить себе следующий разговор.

Руководитель: Я знаю, что мы должны выпустить продукт за 12 недель, но мои оценки показывают, что работа займет 16 недель. Давайте проанализируем оценку при помощи этой программы. Вот несколько предположений, которые я сделал. Сначала нужно откалибровать модель оценки. Для фактора «квалификация программистов» я указал, что наши программисты относятся к 35-й процентилю...

Начальство: Что?! В нашем штате нет ни одного человека ниже среднего уровня! Нужно быть более уверенным в своих оценках! Что же вы за руководитель? Возможно, у нас есть несколько человек, которые чуть отстают от остальных, но в целом группа не может быть настолько плохой. Давайте предположим, что они по крайней мере на среднем уровне, хорошо? Это можно ввести в программу?

Руководитель: Ладно. Следующий фактор – квалификация разработчиков требований. Мы никогда не ставили себе задачу набрать хороших аналитиков или развить эти навыки в своих работниках, поэтому я использовал 15-ю процентиль...

Начальство: Постойте! 15-я процентиль? Наши люди очень талантливы, хотя они и не прошли формального обучения в разработке требований. Они должны быть по крайней мере не хуже среднего. Можно заменить этот фактор средним значением?

Руководитель: Не представляю, как можно назвать их средними. У нас нет ни одного человека, которого можно было бы назвать специалистом по требованиям.

Начальство: Хорошо. Предлагаю компромисс: сойдемся на 35-й процентилю.

Руководитель: Ладно (вздыхает).

В результате разговора оценка объема работы, полученная руководителем с использованием регулирующих факторов Сосомо II, сократилась на 23 %. А если бы начальству удалось уговорить руководителя на среднюю оценку специалистов по

требованиям вместо 35-й процентиля, то оценка бы сократилась на 39 %. Так или иначе, один разговор приводит к весьма существенным различиям.

Руководитель, калибрующий оценку историческими данными, обходит все рассуждения о том, как работают программисты – лучше или хуже среднего. Производительность непосредственно следует из данных. Человеку, ответственному за принятие решений, но не имеющему технического образования, трудно спорить с простыми выкладками типа: «По статистике один человек за месяц у нас выдает от 300 до 450 строк кода; мы откалибровали модель с предположением 400 строк за человека-месяц. Возможно, это немного оптимистично, но в пределах разумного».

Бессспорно, квалификация половины программистов в отрасли ниже средней. Тем не менее я почти не встречал руководителей проектов или начальников, которые бы согласились признать, что *их* программисты имеют квалификацию ниже средней.

СОВЕТ № 36

Исторические данные помогают избежать решений, имеющих политическую подоплеку и возникающих из предположений типа «Моя группа ниже среднего».

8.2. Сбор данных

Если вы еще не занимались сбором исторических данных, начните с очень маленького набора.

- Размер (строки программного кода или другой показатель, который можно получить после выпуска программы).
- Объем работы (человеко-месяцы).
- Время (календарные месяцы).
- Дефекты (с классификацией по степени тяжести).

Эта информация, даже собранная по завершении всего двух-трех проектов, даст достаточные основания для калибровки нескольких коммерческих пакетов оценки. Кроме того, по ней можно будет вычислять простые производные показатели, такие как количество строк кода на человека-месяц.

Наряду с признанием того факта, что этих четырех видов данных достаточно для калибровки моделей оценки, большинство экспертов рекомендует начинать с малого и хорошо разобраться в сути собираемых данных (Pietrasanta 1990, NASA SEL 1995). В противном случае может оказаться, что собранные данные не согласуются между проектами и потому становятся бессмысленными. В зависимости от того, как именно определяются эти четыре показателя, полученные числа могут отличаться в два раза и более.

Проблемы определения размера

Размер завершенных проектов может измеряться в функциональных пунктах, историях пользователей, веб-страницах, таблицах баз данных и множеством других способов, но большинство организаций в конечном счете предпочитает

измерять исторические данные, относящиеся к размеру проектов, в строках программного кода. (Достоинства и недостатки измерения в строках программного кода обсуждаются в разделе 18.1.)

Чтобы вычислить размер проекта в строках кода, необходимо принять решения по нескольким вопросам, в том числе:

- Учитывать ли весь код, или только код, входящий в выпущенную версию программы? (Например, нужно ли считать временный связующий код, код фиктивных объектов, код модульных тестов и код тестирования системы?)
- Как учитывать код, позаимствованный из предыдущих версий?
- Как учитывать свободно распространяемый код или код библиотек сторонних производителей?
- Включать ли в подсчет пустые строки и комментарии?
- Как учитывать интерфейсы классов?
- Как учитывать объявления данных?
- Как подсчитывать одну логическую строку кода, разбитую на несколько физических строк для удобства чтения?

Никаких отраслевых стандартов на этот счет не существует. Более того, не так уж важно, как именно вы на них ответите¹. Важно другое: чтобы вы не меняли принятых решений между проектами, а предположения, заложенные в уже собранные данные, сознательно распространялись на ваши будущие оценки.

Проблемы измерения объема работ

Аналогичные проблемы возникают при сборе данных по объему работ.

- Как измеряется время — в часах, днях или других единицах?
- Сколько рабочих часов в день заложено в вычисления? Стандартные 8 часов или фактическое время, действующее в конкретном проекте?
- Учитывать ли неоплаченные сверхурочные?
- Учитывать ли выходные, отпуска и время обучения?
- Учитывать ли время, проведенное на собраниях уровня компании?
- Какие работы включать в подсчет? Тестирование? Управление первого уровня? Написание документации? Требования? Проектирование? Исследования?
- Как учитывать время, разделенное между несколькими проектами?
- Как учитывать время, потраченное на поддержку предыдущих версий той же программы?

¹ Самое близкое, что могло бы сойти за стандарт в отрасли разработки программного обеспечения, — непустые строки, не состоящие из одних комментариев, включая интерфейсы и объявления данных. Впрочем, даже при таком определении некоторые вопросы остаются без ответов (скажем, как учитывать код, позаимствованный из предыдущих проектов).

- Как учитывать время, потраченное на общение со службой продаж, на проведение выставок и т. д.?
- Как учитывать время командировок?
- Как учитывать нечеткий первоначальный период — время, потраченное на проработку концепции программы до полного определения проекта?

Как и в предыдущем случае, главной целью должно стать достаточно четкое определение собираемых данных. Представьте, что данные из прошлых проектов содержали большой процент неоплаченных сверхурочных и эти исторические данные используются для оценки будущего проекта. Как вы думаете, что при этом произойдет? Вы сами закладываете в свой будущий проект высокий процент сверхурочных.

Проблемы измерения календарного времени

Как ни странно, во многих организациях возникают затруднения с определением сроков работы над тем или иным проектом.

- Что следует считать началом работы над проектом? Формальное согласование бюджета? Начало первых обсуждений проекта? Полное комплектование персоналом? Кейперс Джонс сообщает, что четко определенная начальная точка встречается менее чем в 1 % проектов (Jones 1997).
- Когда проект может считаться завершенным? Когда будет собрана окончательная версия? Когда предполагаемая окончательная версия передается на тестирование? А если большинство программистов прекратило работу над проектом за месяц до выхода официальной версии? Джонс сообщает, что в 15 % проектов встречаются неоднозначные сроки завершения (Jones 1997).

В этой области организации сильно пригодятся четкие определениях ключевых точек начала и завершения проекта. Как и в предыдущих случаях, прежде всего следует стремиться к простому пониманию собираемых данных.

Проблемы измерения дефектов

Наконец, результаты измерения дефектов также могут изменяться в 2–3 раза в зависимости от того, что именно считается дефектом.

- Считываются ли дефектами все запросы на изменения или только те, которые в конечном итоге были отнесены к дефектам (то есть за исключением запросов на внесение новых возможностей)?
- Как рассматривать разные сообщения об одном дефекте — как один или как несколько дефектов?
- Учитывать ли дефекты, обнаруженные разработчиками, или только дефекты, обнаруженные при тестировании?
- Учитывать ли дефекты программирования, обнаруженные до начала альфа- или бета-тестирования?
- Учитывать ли дефекты, о которых сообщили пользователи после выпуска программы?

СОВЕТ № 37

При сборе исторических данных для оценок убедитесь в том, что вы понимаете смысл показателей, и не изменяйте его между проектами.

Другие проблемы сбора данных

Исторические данные проще всего собирать во время работы над проектом. Трудно будет вернуться через полгода после завершения проекта и попытаться припомнить «нечеткий первоначальный период», чтобы определить дату начала проекта, сколько приходилось работать сверхурочно в конце проекта и т. д.

СОВЕТ № 38

Собирайте исторические данные как можно раньше после начала проекта.

Данные, собранные в конце проекта, также полезны, но еще полезнее «моментальные снимки», сделанные непосредственно в ходе работы. Статистика по размерам, объему работ и дефектам, собираемая каждые 1–2 недели, способна дать ценную информацию о динамике проекта.

Например, сбор информации о выявленных дефектах помогает прогнозировать частоту обнаружения дефектов и их исправления в будущих проектах. Данные о распределении объема работы по времени помогут понять, в какой степени ваша организация способна мобилизовать персонал на поддержку проекта. Если комплектование персоналом одного проекта происходит медленнее, чем требуется, возможно, это случайное отклонение. Если же исторические данные указывают на то, что три последних проекта комплектовались с одинаковой скоростью, вполне возможно, что вы столкнулись с организационным фактором влияния, который не удастся легко изменить в следующем проекте.

СОВЕТ № 39

Организуйте периодический сбор исторических данных во время работы над проектом. Это позволит вам позднее построить профиль выполнения проекта, базирующийся на собранных данных.

8.3. Способ калибровки

Конечной целью сбора данных является преобразование данных в модель, которая может использоваться для оценки. Несколько примеров возможных моделей.

- Наши разработчики в среднем выдают X строк кода за человека-месяц.
- Группа из трех человек может реализовать X историй пользователей за календарный месяц.
- Нашей группе в среднем требуется X человеко-часов на создание сценария использования и Y часов на его конструирование и реализацию.
- Наши специалисты по тестированию в среднем создают тестовый сценарий за X часов.

- В нашей рабочей среде один функциональный пункт обычно реализуется X строками кода на языке C# и Y строками на Python.
- До текущего момента работа по исправлению дефектов в проекте в среднем занимала X часов на дефект.

Эти примеры лишь дают представление о моделях, строящихся по историческим данным. Другие примеры приведены в табл. 7.1 из предыдущей главы.

Общей характеристикой всех моделей является их линейность. Математические формулы работают одинаково в системах как на 10 000, так и на 100 000 строк. Тем не менее из-за эффекта издержек масштаба некоторые модели нуждаются в дополнительной регулировке для разных диапазонов размеров. Разбиение по размерам можно попробовать выполнить на неформальном уровне. В табл. 8.1 показан один из примеров.

Таблица 8.1. Пример неформального учета издержек масштаба (только для демонстрационных целей)

Размер группы	Среднее количество историй пользователя на календарный месяц
1	5
2–3	12
4–5	22
6–7	31
8	Для проектов этого размера данные отсутствуют

Такой подход состоятелен при малых расхождениях в размерах проекта. О том, как учитываются большие расхождения в размерах, рассказано в разделах 5.1 и 5.6.

8.4. Уточнение оценок по данным проекта

Ранее в этой главе я указывал, что исторические данные полезны тем, что они учитывают влияние организационных факторов — как признанных, так и скрытых. Тот же принцип применим к использованию исторических данных в конкретных проектах (Gillb 1988, Cohn 2006). Отдельные проекты обладают динамикой, несколько отличающейся от динамики организаций. Использование данных из самого проекта учитывает все факторы влияния, уникальные для данного проекта. Чем скорее в проекте вы начнете основывать оценки на данных самого проекта, тем ваши оценки станут по-настоящему точными.

СОВЕТ № 40

Используйте данные, полученные в ходе текущего проекта (проектные данные), для создания высокоточных оценок для оставшейся части проекта.

Даже если вы не располагаете историческими данными из прошлых проектов, соберите данные по текущему проекту и используйте их для оценки оставшейся части проекта. Ваша задача — как можно скорее переключиться с организационных или отраслевых данных на проектные. Чем более итеративными будут ваши проекты, тем скорее появится такая возможность.

Сбор и использование данных из ваших проектов более подробно рассматриваются в разделе 16.4. В разделе 12.3 представлен конкретный пример использования проектных данных для уточнения оценок.

8.5. Калибровка средними отраслевыми данными

Если вы не располагаете собственными историческими данными, особого выбора не остается — используйте средние отраслевые показатели; это приемлемый, хотя и не лучший вариант. Как видно из табл. 5.2, производительность разных организаций в одной отрасли может отличаться на порядок. Средняя производительность не учитывает то обстоятельство, что ваша организация может находиться на верхней или нижней части диапазона.

На рис. 8.1 показан пример оценки, созданной по отраслевым данным. Каждая точка графика представляет возможный результат проекта, созданный с использованием статистического моделирования по методу Монте-Карло. Сплошные черные линии представляют срединные значения объемов работ и сроков, найденные в результате моделирования. Пунктирные черные линии представляют 25-й и 75-й процентили объема работ и сроков.

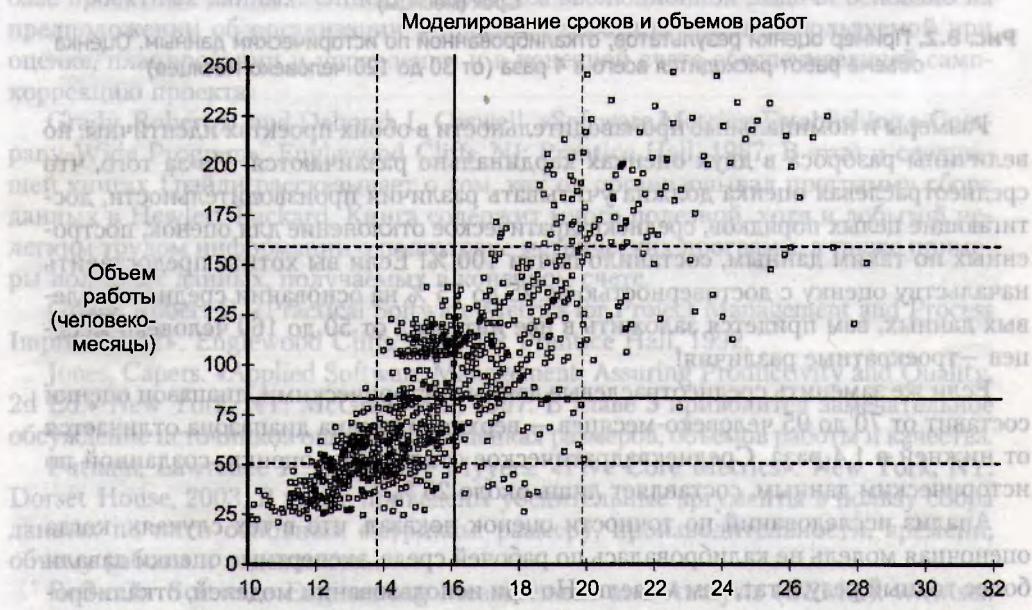


Рис. 8.1. Пример оценки результатов, откалиброванной по среднеотраслевым данным. Разброс оценок объема работ составляет целый порядок (от 25 до 250 человеко-месяцев)

На рис. 8.2 изображен пример сходной оценки, откалиброванной по историческим данным одного из моих клиентов.



Рис. 8.2. Пример оценки результатов, откалиброванной по историческим данным. Оценка объема работ расходится всего в 4 раза (от 30 до 120 человеко-месяцев)

Размеры и номинальные производительности в обоих проектах идентичны, но величины разброса в двух оценках кардинально различаются. Из-за того, что среднеотраслевая оценка должна учитывать различия производительности, достигающие целых порядков, среднеквадратическое отклонение для оценок, построенных по таким данным, составило почти 100%! Если вы хотите предоставить начальству оценку с достоверностью от 25 до 75% на основании среднеотраслевых данных, вам придется заложить в нее интервал от 50 до 160 человеко-месяцев — троекратные различия!

Если же заменить среднеотраслевые данные историческими, диапазон оценки составит от 70 до 95 человеко-месяцев — верхняя граница диапазона отличается от нижней в 1,4 раза. Среднеквадратическое отклонение оценки, созданной по историческим данным, составляет лишь около 25%.

Анализ исследований по точности оценок показал, что в тех случаях, когда оценочная модель не калибровалась по рабочей среде, экспертные оценки давали более точный результат, чем модели. Но при использовании моделей, откалиброванных по историческим данным, выяснялось, что модели дают такой же или лучший результат, чем экспертные оценки (Jørgensen 2002).

СОВЕТ № 41

Там, где это возможно, используйте для калибровки оценок проектные или исторические данные вместо среднеотраслевых. Помимо повышения точности оценок, исторические данные сокращают разброс оценок, обусловленный неопределенностью предположений по поводу производительности.

8.6. Итоги

Теперь, когда вы знаете, какими возможностями обладают исторические данные, пренебрегать ими в ваших оценках было бы непростительно. В следующем году, когда вы будете снова перечитывать эту главу, вы уже не должны горченно повторять: «Как жаль, что у меня нет исторических данных!»

СОВЕТ № 42

Если в настоящий момент у вас еще нет исторических данных, начните собирать их как можно скорее.

Дополнительные ресурсы

Boehm, Barry, et al. «Software Cost Estimation with Cocomo II». Reading, MA: Addison-Wesley, 2000. Контрольный список из приложения Е книги Бема поможет вам определиться с тем, что же считать «строкой кода».

Gilb, Tom. «Principles of Software Engineering Management». Wokingham, England: Addison-Wesley, 1988. В разделе 7.14 книги Гилба описано уточнение оценок на базе проектных данных. Описание процесса эволюционной выдачи основано на предположении об организации в проектах обратной связи, используемой при оценке, планировании и управлении и в конечной счете обеспечивающей самокоррекцию проекта.

Grady, Robert B. and Deborah L. Caswell. «Software Metrics: Establishing a Company-Wide Program». Englewood Cliffs, NJ: Prentice Hall, 1987. В этой и следующей книгах Грэйди рассказывает о том, как он организовывал программу сбора данных в Hewlett-Packard. Книга содержит много полезной, хотя и добытой не легким трудом информации о подготовке метрических программ, а также примеры полезных данных, получаемых в конечном счете.

Grady, Robert B. «Practical Software Metrics for Project Management and Process Improvement». Englewood Cliffs, NJ: PTR Prentice Hall, 1992.

Jones, Capers. «Applied Software Management: Assuring Productivity and Quality, 2d Ed.» New York, NY: McGraw-Hill, 1997. В главе 3 приводится замечательное обсуждение источников ошибок при оценках размеров, объемов работы и качества.

Putnam, Lawrence H. and Ware Muuyers. «Five Core Metrics». New York, NY: Dorset House, 2003. В книге приведены убедительные аргументы в пользу сбора данных по пяти основным метрикам: размеру, производительности, времени, объему работы и надежности.

Веб-сайт Software Engineering Measurement and Analysis (SEMA): www.sei.cmu.edu/sema/. Сайт содержит разностороннюю информацию по формированию в организациях навыков сбора данных и их использования в оценке.

9

Индивидуальные экспертивные оценки

Область применения методов этой главы

	Структурированный процесс	Контрольный список оценки	Диапазонная оценка задач	Сравнение оценок задач с реальными значениями
Что оценивается	Объем работ, сроки, функциональность	Объем работ, сроки, функциональность	Размер, объем работ, сроки, функциональность	Объем работ, сроки, функциональность
Размер проекта	М С Б	М С Б	М С Б	М С Б
Стадия разработки	Ранняя–поздняя	Ранняя–поздняя	Ранняя–поздняя	Средняя–поздняя
Итеративный или последовательный стиль	Оба	Оба	Оба	Оба
Возможная точность	Высокая	Высокая	Высокая	—

Индивидуальные экспертные суждения до сих пор остаются самым распространенным методом оценки, применяемым на практике (Jørgensen 2002). Хин и Хабибагахи обнаружили, что 83 % оценщиков используют «неформальные аналогии» в качестве основного инструмента оценки (Hihn and Habib-agahi 1991). Проведенное в Новой Зеландии исследование показало, что 86 % фирм-разработчиков используют «экспертную оценку» (Paynter 1996). Барбара Китченхэм и ее коллеги также выяснили, что «мнение эксперта» задействовано в 72 % оценок проектов (Kitchenham et al. 2002).

Экспертные суждения относительно отдельных задач образуют основу для восходящей оценки, но все экспертные суждения равны. И действительно, в главе 7 экспертные суждения названы самым рискованным методом оценки.

Говоря о «мнении эксперта», прежде всего необходимо спросить — «эксперта в какой области?» Большой опыт в технологии или разработке еще не делает работника экспертом в области оценок. Йоргенсен сообщает, что накопление опыта в оцениваемой области не приводит к повышению точности оценок (Jørgensen 2002). В других аналитических работах выяснилось, что «эксперты» склонны применять простые стратегии оценки даже при высокой компетентности в оцениваемой теме (Josephs and Hahn 1995, Todd and Benbasat 2000).

В этой главе описано, как повысить эффективность экспертных суждений. Обсуждение тесно связано с материалом главы 10, посвященной точному объединению отдельных оценок.

9.1. Структурированные экспертные суждения

Индивидуальные экспертные оценки не обязаны быть неформальными или интуитивными. Исследователи обнаружили значительные расхождения в точности между «интуитивными экспертными суждениями», которые обычно оказывались неточными (Lederer and Prasad 1992), и «структурированными экспертными суждениями». Оценки, построенные на основании последних, не уступали по точности оценкам, полученным при помощи математических моделей (Jørgensen 2002).

Кто создает оценки?

Для конкретных задач — таких, как время, необходимое на программирование и откладку некоторой функции, или создание набора тестовых сценариев — наиболее точные оценки создаются людьми, непосредственно выполняющими эту работу. Оценки людей, не связанных с работой, оказываются менее точными (Lederer and Prasad 1992). Кроме того, сторонние оценщики в большей степени склонны к недооценкам, чем связанные с разработкой (Lederer and Prasad 1992).

СОВЕТ № 43

Оценки уровня задач следует поручать людям, непосредственно занимающимся выполнением соответствующей работы.

Приведенный совет относится именно к оценкам уровня задач. Если ваш проект все еще находится в широкой части конуса неопределенности (то есть конкретные задачи еще не были идентифицированы или назначены исполнителям), оценка должна создаваться опытным оценщиком или самыми опытными разработчиками, специалистами по контролю качества и документированию из имеющегося персонала.

Гранулярность

Одним из лучших способов повышения точности оценок уровня задач является разбиение крупной задачи на несколько меньших. При создании оценок разработчики, руководители и специалисты по тестированию обычно сосредоточиваются

на хорошо понятных им задачах и пренебрегают задачами, которые им незнакомы. Типичный результат — односторонняя запись в графике (скажем, «преобразование данных»), на выполнение которой должно было уйти две недели, занимает два месяца, потому что никто не удосужился выяснить, что именно необходимо сделать.

Оценки, составляемые на уровне задач, следует разбивать на задачи, на выполнение которых потребуется не более двух дней работы. Более крупные задачи содержат слишком много «подводных камней», требующих непредвиденной работы. Желательно, чтобы гранулярность полученных оценок составляла $1/4$, $1/2$ или полный день.

Диапазоны

Если попросить разработчика оценить некий набор функций, он обычно выдает набор чисел — вроде того, который приведен в табл. 9.1.

Таблица 9.1. Пример точечных оценок, выданных разработчиком

Функция	Оценка времени реализации (в днях)
Функция 1	1,5
Функция 2	1,5
Функция 3	2,0
Функция 4	0,5
Функция 5	0,5
Функция 6	0,25
Функция 7	2,0
Функция 8	1,0
Функция 9	0,75
Функция 10	1,25
ИТОГО	11,25

Если затем попросить того же разработчика выдать те же оценки для лучшего и худшего случая, разработчик обычно выдает другой набор чисел, наподобие приведенного в табл. 9.2.

Если сравнить исходные точечные оценки с оценками для лучшего и худшего случая, мы видим, что сумма 11,25 для точечных оценок гораздо ближе к оценке лучшего случая (10,5), чем к оценке худшего случая (18,25).

Взглянув на оценки функции 4, также можно заметить, что обе оценки выше исходной точечной оценки. В ходе анализа худшего случая иногда выявляется дополнительная работа, которая должна быть выполнена даже в лучшем случае, а это приводит к повышению номинальной оценки. Обычно при анализе худших случаев я спрашиваю разработчиков, сколько времени займет выполнение задачи, если все пойдет неправильно. Часто выясняется, что мне выдавали оценку для оптимистичного худшего случая, а не для настоящего худшего случая.

Таблица 9.2. Пример индивидуальных оценок для лучшего и худшего случаев

Функция	Оценка времени реализации (в днях)	
	Лучший случай	Худший случай
Функция 1	1,25	2,0
Функция 2	1,5	2,5
Функция 3	2,0	3,0
Функция 4	0,75	2,0
Функция 5	0,5	1,25
Функция 6	0,25	0,5
Функция 7	1,5	2,5
Функция 8	1,0	1,5
Функция 9	0,5	1,0
Функция 10	1,25	2,0
ИТОГО	10,5 ¹	18,25

Если вы руководите проектом, поручите своим разработчикам создать набор точечных оценок. Скройте от них результаты и предложите создать набор оценок для лучших или худших случаев. Сравните полученные оценки с исходными точечными — результат часто оказывается неожиданным.

Это упражнение полезно по двум причинам. Во-первых, оно помогает понять, что точечные оценки ближе к оценкам для лучшего случая. Во-вторых, после многократной формулировки оценок в вашем сознании начнет формироваться привычка к анализу худшего из возможных случаев. Если вы привыкнете рассматривать как лучший, так и худший результат, вы постепенно начнете включать весь диапазон возможных результатов в точечные оценки задач независимо от того, были реально сформулированы оценки для лучшего и худшего случаев или нет.

СОВЕТ № 44

Создание оценок для лучшего и худшего случаев стимулирует мышление и помогает учесть весь возможный диапазон возможных результатов.

Формулы

Создание оценок для лучшего и худшего случаев — всего лишь первый шаг. Вы по-прежнему стоите перед вопросом, какую из оценок использовать. А может, вычислить среднее арифметическое? В действительности оба варианта не годятся. Во многих случаях худший случай намного хуже того, что можно называть «ожидаемым случаем». Вычисление середин по диапазонам приведет к нежелательному смещению оценки.

¹ При простом суммировании оценок для лучшего и худшего случаев возникают некоторые статистические аномалии. Эта тема более подробно обсуждается в главе 10.

118 Глава 9 • Индивидуальные экспертные оценки

Методика PERT (Program Evaluation and Review Technique) предназначена для вычисления ожидаемого случая, который может и не находиться в средней точке диапазона между лучшим и худшим случаями (Putnam and Myers 199, Stutzke 2005). Для использования PERT в набор случаев включается дополнительный «наиболее вероятный» случай, который оценивается посредством экспертного суждения. Затем ожидаемый случай вычисляется по формуле:

ФОРМУЛА № 1

$$\text{Ожидаемый случай} = [\text{ЛучшийСлучай} + (4 \times \text{НаиболееВероятныйСлучай}) + \text{ХудшийСлучай}] / 6.$$

Формула учитывает как полную длину диапазона, так и позицию наиболее вероятного случая внутри диапазона. В табл. 9.3 показаны оценки из табл. 9.2 с добавлением наиболее вероятного и ожидаемого случаев. Как видно из таблицы, общая оценка 13,62 ближе к нижнему концу диапазона, чем среднее значение 14,4.

Таблица 9.3. Пример индивидуальной оценки с лучшим, худшим и наиболее вероятным случаями

Функция	Оценка времени реализации (в днях)			
	Лучший случай	Наиболее вероятный случай	Худший случай	Ожидаемый случай
Функция 1	1,25	1,5	2,0	1,54
Функция 2	1,5	1,75	2,5	1,83
Функция 3	2,0	2,25	3,0	2,33
Функция 4	0,75	1	2,0	1,13
Функция 5	0,5	0,75	1,25	0,79
Функция 6	0,25	0,5	0,5	0,46
Функция 7	1,5	2	2,5	2,00
Функция 8	1,0	1,25	1,5	1,25
Функция 9	0,5	0,75	1,0	0,75
Функция 10	1,25	1,5	2,0	1,54
ИТОГО	10,5	13,25	18,25	13,62

Как обсуждалось в главе 4, «наиболее вероятные» оценки склонны к излишнему оптимизму, что может привести к оптимистическому завышению общих оценок при использовании этого подхода. Некоторые эксперты в области оценки рекомендуют изменить базовую формулу PERT, чтобы учесть смещение в оценке (Stutzke 2005). Измененная формула выглядит так:

ФОРМУЛА № 2

$$\text{Ожидаемый случай} = [\text{ЛучшийСлучай} + (3 \times \text{НаиболееВероятныйСлучай}) + (2 \times \text{ХудшийСлучай})] / 6.$$

Это разумное решение проблемы для краткосрочной перспективы. Долгосрочное решение проблемы — работа с людьми, направленная на повышение точности их оценок наиболее вероятного случая.

Контрольные списки

Даже эксперты иногда забывают учесть все необходимые факторы. При изучении прогнозирования в различных дисциплинах выяснилось, что простые контрольные списки способствуют повышению точности и напоминают о тех факторах, о которых эксперт мог бы забыть (Park 1996, Harvey 2001, Jørgensen 2002). В табл. 9.4 представлен контрольный список, который можно было бы использовать для повышения точности оценок.

Таблица 9.4. Контрольный список для индивидуальной оценки

1. Насколько четко определен оцениваемый показатель?
2. Включает ли оценка все виды работ, необходимые для завершения задачи?
3. Включает ли оценка все функциональные области, необходимые для завершения задачи?
4. Достаточно ли детализирована оценка для выявления скрытой работы?
5. Вы ознакомились с документированными фактами (письменными заметками) из прошлой работы, не ограничиваясь оценкой исключительно по собственным воспоминаниям?
6. Согласована ли оценка с человеком, который будет заниматься непосредственным выполнением работы?
7. Производительность, заложенная в оценку, сходна с той, которая достигалась в аналогичных ситуациях?
8. Включены ли в оценку различные случаи — лучший, худший и наиболее вероятный?
9. Действительно ли худший случай является худшим? Нельзя ли сделать его еще хуже?
10. Насколько правильно вычисляется ожидаемый случай?
11. Документированы ли предположения по поводу оценки?
12. Изменилась ли ситуация с момента подготовки оценки?

Чтобы ничего не упустить, также сверьтесь с перечнем часто пропускаемых действий в разделе 4.5.

СОВЕТ № 45

Используйте контрольные списки для улучшения индивидуальных оценок. Составляйте и ведите собственные контрольные списки.

9.2. Сравнение оценок с фактическими значениями

Отказ от точечных оценок и лучших случаев — всего лишь половина дела. Другой половиной должно стать сравнение фактических результатов с оцениваемыми, чтобы вы могли подрегулировать свои личные способности к оценке.

Ведите список оценок и дополняйте его фактическими результатами по завершении проекта. Затем вычислите величину относительной ошибки (MRE, Magnitude of Relative Error) своих оценок (Conte, Dunsmore, and Shen 1986). MRE вычисляется по следующей формуле:

ФОРМУЛА № 3

$$\text{MRE} = \text{Абсолютное Значение} \times [(\text{Фактический Результат} - \text{Оценка Результата}) / \text{Фактический Результат}]$$

В табл. 9.5 приведены результаты вычисления MRE для оценок лучшего и худшего случаев из предыдущего примера.

Таблица 9.5. Пример таблицы для отслеживания точности отдельных оценок

Функция	Оценка времени реализации (в днях)			Фактический результат	MRE	Входит в диапазон между лучшим и худшим случаем?
	Лучший случай	Худший случай	Ожидаемый случай			
Функция 1	1,25	2,0	1,54	2	23 %	Да
Функция 2	1,5	2,5	1,83	2,5	27 %	Да
Функция 3	2,0	3,0	2,33	1,25	87 %	Нет
Функция 4	0,75	2,0	1,13	1,5	25 %	Да
Функция 5	0,5	1,25	0,79	1	21 %	Да
Функция 6	0,25	0,5	0,46	0,5	8 %	Да
Функция 7	1,5	2,5	2,00	3	33 %	Нет
Функция 8	1,0	1,5	1,25	1,5	17 %	Да
Функция 9	0,5	1,0	0,75	1	25 %	Да
Функция 10	1,25	2,0	1,54	2	23 %	Да
ИТОГО	10,5	18,25	13,62	16,25	80 %	Да
Среднее					29 %	

В этой таблице для каждой оценки вычисляется относительная ошибка (MRE). Ее среднее значение по набору оценок, показанное в нижней строке, составляет 29 %. Вы можете использовать среднюю относительную ошибку для измерения точности своих оценок. По мере улучшения точности MRE начинает убывать. Правый столбец показывает, сколько оценок попадает в диапазон между лучшим и худшим случаем. Также можно проследить за тем, как со временем растет процент оценок, попадающих в диапазон.

СОВЕТ № 46

Сравнивайте оценки с фактическими результатами, чтобы повышать качество своих оценок со временем.

Сравнивая оценки с результатами, попытайтесь понять, что было сделано верно, что неверно, что вы упустили, и как избежать повторения этих ошибок в будущем.

Другим средством организации обратной связи и повышения точности оценок является общественное обсуждение. Я работал с компаниями, в которых разработчики должны были на собраниях по понедельникам сообщать, в какой степени их оценки соответствовали результатам. Таким образом укреплялись представления о точности оценок как об одном из стратегических приоритетов компании.

Какой бы способ вы ни выбрали, ключевой принцип остается одним и тем же: организация цикла обратной связи на основании фактических оценок, чтобы ваши оценки улучшались со временем. Чтобы обратная связь была эффективной, она должна быть своевременной; задержка снижает эффективность цикла обратной связи (Jørgensen 2002).

Дополнительные ресурсы

Jørgensen, M. «A Review of Studies on Expert Estimation of Software Development Effort», 2002. В статье приводится разносторонний обзор исследований в области экспертных оценок. Автор пользуется многими выводами, полученными в ходе стандартных исследований, и представляет 12 рекомендаций для достижения точных экспертных оценок.

Humphrey, Watts S. «A Discipline for Software Engineering». Reading, MA: Addison-Wesley, 15. Хамфри излагает подробную методологию для сбора разработчиками личных данных производительности, сравнения запланированных результатов с фактическими и постепенного совершенствования оценок.

Stutzke, Richard D. «Estimating Software-Intensive Systems». Upper Saddle River, NJ: Addison-Wesley, 2005. В главе 5 книги Стуцке рассматриваются методы экспертных оценок, а также приводятся математические обоснования некоторых формул, упоминавшихся в этой главе.

Декомпозиция и сводные оценки

10

Область применения методов этой главы

	Декомпозиция по функциям или задачам	Декомпозиция WBS (Work Breakdown Structure)	Вычисление лучшего и худшего случаев по стандартному отклонению
Что оценивается	Размер, объем работ, функциональность	Объем работ	Объем работ, сроки
Размер проекта	М С Б	– С Б	М С Б
Стадия разработки	Ранняя–поздняя (малые проекты); средняя–поздняя (средние и крупные проекты)	Ранняя–поздняя	Ранняя–поздняя (малые проекты); средняя–поздняя (средние и крупные проекты)
Итеративный или последовательный стиль	Оба	Оба	Оба
Возможная точность	Средняя–высокая	Средняя	Средняя

Декомпозицией называется разбиение оценки на фрагменты, раздельная оценка каждого фрагмента и последующее объединение отдельных оценок в составную оценку. Данная методика также известна под названием «восходящей оценки», «микрооценки», «модульного наращивания» и другими названиями (Tockey 2005).

Декомпозиция является краеугольным камнем оценки, однако при ее использовании необходимо остерегаться некоторых ошибок. В этой главе более подробно обсуждается базовая схема и объясняется, как обойти скрытые ловушки.

10.1. Вычисление ожидаемого случая

Сцена: Еженедельное собрание группы...

Вы: Мы должны создать оценку для нового проекта. Чтобы подчеркнуть, насколько важны точные оценки для этой группы, я готов поспорить на обед с пиццей, о точности оценок как об одном из стратегических приоритетов компании.

что моя оценка проекта будет точнее вашей. Если вы выигрываете, с меня пицца; если выигрываю я — пицца за ваш счет. Желающие есть?

Группа: По рукам!

Вы: Ладно, начинаем.

После поиска информации о похожем прошлом проекте выясняется, что проект занял 18 человеко-недель. Вы прикидываете, что нынешний проект примерно на 20 % больше предыдущего, и создаете общую оценку в 22 человека-недели.

Тем временем ваши коллеги создали более подробную оценку с разбиением на функции. У них получилась оценка, показанная в табл. 10.1.

Таблица 10.1. Пример оценки, полученной посредством декомпозиции

Функция	Оценка времени реализации (в человеко-неделях)
Функция 1	1,5
Функция 2	4
Функция 3	4
Функция 4	1
Функция 5	4
Функция 6	6
Функция 7	2
Функция 8	1
Функция 9	3
Функция 10	1,5
ИТОГО	27

Вы: 27 недель? Ого. По-моему, многовато, но это скоро выяснится...

Несколько недель спустя...

Вы: Теперь, когда проект завершен, мы знаем, что он занял 29 человеко-недель. Похоже, ваша оценка в 27 человеко-недель оказалась оптимистичной на 2 недели, то есть ошибка составила 7 %. Моя оценка в 22 человека-недели смещена на 7 человеко-недель, ошибка 24 %. Похоже, вы выиграли, и я покупаю пиццу.

Кстати говоря, я хочу знать, из-за кого я « попал » на пиццу. Давайте посмотрим, какие из частичных оценок были самыми точными.

Несколько минут уходит на то, чтобы проанализировать величины относительной ошибки по всем оценкам и выписать данные на доску. Результат показан в табл. 10.2.

Группа: Ого, как интересно. Большинство наших индивидуальных оценок не были точнее вашей — почти все они содержали ошибку от 30 до 50 %. Наша средняя ошибка составляла 46 %, что гораздо больше, чем у вас. И все же наша общая ошибка составила всего 7 %, а ваша — 24 %.

Однако договор дороже денег. Хотя наши оценки были хуже вашей, пицца все равно с вас!

Таблица 10.2. Примеры результатов оценок, полученных посредством декомпозиции

Функция	Оценка времени реализации (в человеко-неделях)	Фактический объем работы	Непосредственная ошибка	Величина относительной ошибки
Функция 1	1,5	3,0	-1,5	50 %
Функция 2	4,5	2,5	2,0	80 %
Функция 3	3	1,5	1,5	100 %
Функция 4	1	2,5	-1,5	60 %
Функция 5	4	4,5	-0,5	11 %
Функция 6	6	4,5	1,5	33 %
Функция 7	2	3,0	-1,0	33 %
Функция 8	1	1,5	-0,5	33 %
Функция 9	3	2,5	0,5	20 %
Функция 10	1,5	3,5	-2,0	57 %
ИТОГО	27	29	-2	-
Среднее	-	-	-7 %	46 %

Каким-то образом итоговая оценка группы оказалась точнее вашей, хотя оценки по отдельным функциям были хуже. Как такое возможно?

Закон больших чисел

Оценка группы выиграла от статистического свойства, называемого **законом больших чисел**. Суть закона заключается в том, что при создании одной «большой» оценки ошибочные тенденции будут полностью сосредоточены на верхней или нижней стороне. Но если создать несколько меньших оценок, часть ошибок окажется с одной стороны, а часть — с другой. Ошибки до определенной степени компенсируют друг друга. В одних случаях ваша группа недооценивала результат, но в других — переоценивала его, поэтому ошибка совокупной оценки составила всего 7 %. В вашей оценке все 24 % ошибок были сосредоточены с одной стороны.

Такой подход должен работать в теории, и исследования говорят о том, что он также работает на практике. Ледерер и Прасад обнаружили, что суммирование продолжительности подзадач отрицательно коррелируется с превышениями сроков и объемов работ (Lederer and Prasad 1992).

СОВЕТ № 47

Разбейте общую оценку на фрагменты, чтобы воспользоваться действием закона больших чисел: ошибки в сторону завышения и занижения до определенной степени компенсируют друг друга.

Насколько мелкими должны быть оцениваемые фрагменты?

Если взглянуть на ситуацию с точки зрения рис. 10.1, разработка программного обеспечения представляет собой постепенное сокращение масштаба принимаемых решений. В начале проекта вы принимаете решения типа: «Какие основные

области должна содержать эта программа?» Простое решение о включении или исключении одной области способно существенно сместить общий объем работ или сроки проекта в одну или другую сторону. При приближении к постановке требований высокого уровня вы принимаете большее количество решений о том, какие функции должны присутствовать или отсутствовать в проекте, но каждое из этих решений в среднем оказывает минимальное воздействие на общий результат проекта. При переходе к детализированным требованиям обычно приходится принимать сотни решений; одни из них имеют большие последствия, другие — меньшие, но в среднем их влияние оказывается гораздо меньшим, чем у решений, принимаемых на более ранней стадии проекта.

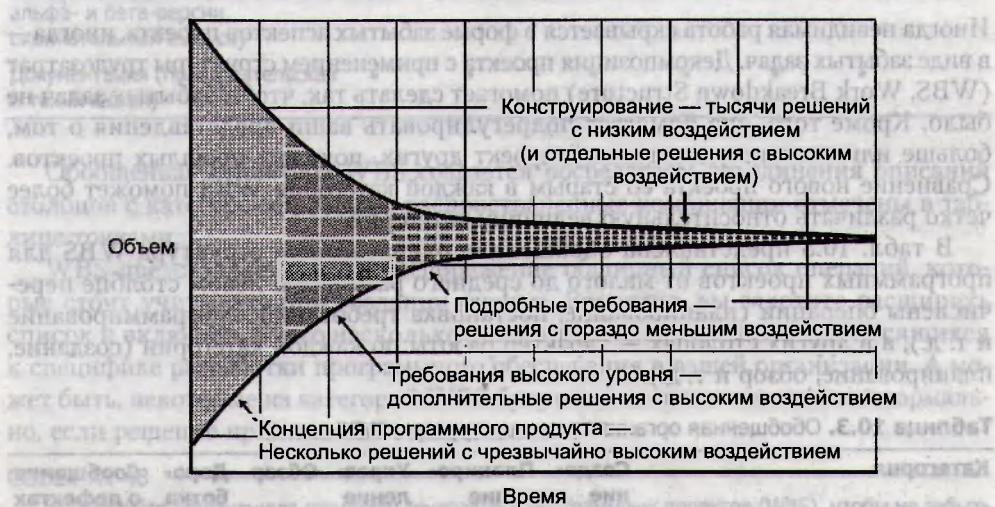


Рис. 10.1. Программные проекты обычно прогрессируют от крупномасштабных решений на начальной стадии к мелкомасштабным решениям в конце. Эта тенденция повышает роль оценок, получаемых с использованием декомпозиции, по мере развития проекта

К тому моменту, когда вы сосредоточитесь на конструировании программы, масштаб принимаемых решений становится совсем малым: «Как спроектировать интерфейс этого класса? Как назвать эту переменную? Как организовать этот цикл?» И т. д. Эти решения по-прежнему играют важную роль, однако эффект любого отдельного решения получается локализованным по сравнению с крупными решениями, принимаемыми на исходном, концептуальном уровне.

Таким образом, разработка программного обеспечения представляет собой процесс постепенного уточнения требований, и чем дальше продвигается проект к завершению, тем более детализированными становятся оценки, полученные в результате декомпозиции. На ранней стадии проекта восходящая оценка может базироваться на функциональных областях. Позднее за основу берется оценка маркетинговых требований. Еще позднее используются подробные или технические требования. На завершающей стадии проекта можно использовать оценки уровня задач, предоставленные разработчиками и специалистами по тестированию.

Ограничения количества оцениваемых показателей имеют более практический, нежели теоретический смысл. На очень ранней стадии проекта бывает не просто получить достаточное количество подробной информации для создания детализированной оценки. На более поздних стадиях информации может оказаться слишком много. Чтобы закон больших чисел дал сколько-нибудь заметный выигрыш, потребуется от 5 до 10 элементов, но даже 5 элементов лучше, чем один.

10.2. Декомпозиция с использованием WBS

Иногда невидимая работа скрывается в форме забытых аспектов проекта, иногда — в виде забытых задач. Декомпозиция проекта с применением структуры трудозатрат (WBS, Work Breakdown Structure) помогает сделать так, чтобы забытых задач не было. Кроме того, она помогает подрегулировать ваши представления о том, больше или меньше оцениваемый проект других, похожих прошлых проектов. Сравнение нового проекта со старым в каждой категории WBS поможет более четко различать относительную величину категорий.

В табл. 10.3 представлена обобщенная операционная структура WBS для программных проектов от малого до среднего размера. В левом столбце перечислены операции (планирование, постановка требований, программирование и т. д.), а в других столбцах — характер работы по каждой категории (создание, планирование, обзор и т. д.).

Таблица 10.3. Обобщенная организационная структура WBS

Категория	Создание	Планирование	Управление	Обзор	Доработка	Сообщение о дефектах
Общее управление	•	•	•	•	•	•
Планирование	•	•	•	•	•	•
Корпоративная деятельность (собрания, отпуска, выходные и т. д.)	•					
Настройка конфигурации оборудования/программного обеспечения/Сопровождение	•	•	•	•	•	•
Подготовка персонала	•	•	•	•	•	•
Технические/технологические процессы	•	•	•	•	•	•
Работа по требованиям	•	•	•	•	•	•
Координация с другими проектами	•	•	•	•	•	•
Управление изменениями	•	•	•	•	•	•
Макетирование пользовательского интерфейса	•	•	•	•	•	•
Проработка архитектуры	•	•	•	•	•	•
Подробное проектирование	•	•	•	•	•	•

Категория	Создание	Планирование	Управление	Обзор	Доработка	Сообщение о дефектах
Программирование	•	•	•	•	•	•
Приобретение компонентов	•	•	•	•	•	•
Автоматизированная сборка	•	•	•	•	•	•
Интеграция	•	•	•	•	•	•
Ручное тестирование системы	•	•	•	•	•	•
Автоматизированное тестирование системы	•	•	•	•	•	•
Выпуск программы (внутренние, альфа- и бета-версии, окончательный выпуск)	•	•	•	•	•	•
Документация (пользовательская и техническая)	•	•	•	•	•	•

Обобщенная структура WBS создается посредством объединения описаний столбцов с категориями. Самые распространенные комбинации отмечены в таблице точками.

WBS представляет в ваше распоряжение обширный список операций, которые стоит учитывать при создании оценки. Вероятно, вы захотите расширить список и включить в него несколько дополнительных элементов, относящихся к специфике разработки программного обеспечения в вашей организации. А может быть, некоторые из категорий WBS будут исключены — это вполне нормально, если решение принимается сознательно.

СОВЕТ № 48

Используйте обобщенную структуру трудозатрат программных проектов (WBS), чтобы не забыть о типовых операциях.

10.3. Риск суммирования оценок для лучших и худших случаев

Представьте, что вы составили подробный список задач. Вы тщательно оцениваете каждую задачу в списке, думая: «Возможно, у нас это и получится, если как следует постараться». После тщательного планирования вы усердно работаете над первой задачей и выдаете ее в срок. Во второй задаче возникают непредвиденные проблемы, но вы сидите допоздна и сдаете ее по графику. В третьей задаче появляется еще несколько проблем; в конце дня вы оставляете ее незавершенной, полагая, что завершите ее следующим утром. Но к концу следующего дня вам едва удается справиться с ней, а к той задаче, которую полагалось выполнять в этот день, вы даже не притронулись. К концу недели вы более чем на полную задачу отстаете от графика.

Что произошло? Вы ошиблись в оценках или просто недостаточно усердно работали?

Осторожно: впереди математика!

Ответ кроется в статистических тонкостях, возникающих при объединении отдельных оценок. *Статистические тонкости?* Да, нравится вам этот или нет, но чтобы понять, как избежать распространенных проблем построения сводных оценок по декомпозиционным оценкам задач или возможностей, нам придется немного заняться математикой.

Где произошел сбой?

Чтобы понять, что случилось в предыдущем сценарии, давайте вернемся к сценарию, описанному в начале главы. Группа выдала точную оценку, однако точность ее точечных оценок была необычной. Более типичная попытка выдать оценку посредством декомпозиции не приведет к оценкам, представленным в табл. 10.1; скорее всего, полученные оценки будут выглядеть примерно так, как показано в табл. 10.4.

Таблица 10.4. Примеры более типичной, подверженной ошибкам попытки создания оценки посредством декомпозиции

Функция	Оценка времени реализации (в человеко-неделях)	Фактический объем работы
Функция 1	1,6	3,0
Функция 2	1,8	2,5
Функция 3	2,0	1,5
Функция 4	0,8	2,5
Функция 5	3,8	4,5
Функция 6	3,8	4,5
Функция 7	2,2	3,0
Функция 8	0,8	1,5
Функция 9	1,6	2,5
Функция 10	1,6	3,5
ИТОГО	20,0	29,0

В этом примере точность 20-недельной оценки, полученной простым суммированием фрагментарных точечных оценок, оказывается хуже сводной оценки в 22 человеко-недели, предоставленной вами в этом сценарии. Как такое могло случиться?

Корнем проблемы является сочетание проблемы «90 % уверенности», обсуждавшейся в главе 1, и проблемы необоснованного оптимизма из главы 4. Когда разработчика просят предоставить точечную оценку, он часто бессознательно выдает оценку для лучшего случая. Допустим, что каждая из оценок лучшего случая вероятна на 25 % (другими словами, вероятность того, что работа будет выполнена с указанным результатом или лучше, составляет всего 25 %). Шансы выдачи любой отдельной задачи в соответствии с оценкой лучшего случая невелики:

1 из 4 (25 %). Однако шанс выдачи *всех* задач по графику стремится к нулю. Чтобы выдать вовремя первую и вторую задачи, необходимо уложиться в вероятность $1/4$ в первой задаче, а потом сделать то же самое во второй. Со статистической точки зрения вероятности перемножаются, поэтому вероятность своевременного завершения обеих задач составит всего $1/16$. Чтобы вычислить вероятность своевременного завершения всех 10 задач, необходимо перемножить $1/4$ 10 раз; результат составляет примерно $1/1\ 000\ 000$, или 0,000095 %. На уровне отдельных задач вероятность $1/4$ выглядит неплохо, но объединенная вероятность губит программные проекты. Статистика объединения оценок для худших случаев выглядит аналогично.

Эти статистические аномалии образуют другую причину для определения нескольких оценок для лучшего, худшего, наиболее вероятного и ожидаемого случаев — см. главу 9. В табл. 10.5 показано, что может произойти, если попросить эти оценки у разработчиков, выдавших оценки из табл. 10.4, и вычислить по ним оценки для ожидаемого случая.

Таблица 10.5. Пример оценки, полученной посредством декомпозиции, с выделением лучшего, ожидаемого и худшего случаев

Функция	Недель на завершение			
	Лучший случай (вероятность 25 %)	Наиболее вероятный случай	Худший случай (вероятность 75 %)	Ожидаемый случай (вероятность 50 %)
Функция 1	1,6	2,0	3,0	2,10
Функция 2	1,8	2,5	4,0	2,63
Функция 3	2,0	3,0	4,2	3,03
Функция 4	0,8	1,2	1,6	1,20
Функция 5	3,8	4,5	5,2	4,50
Функция 6	3,8	5,0	6,0	4,97
Функция 7	2,2	2,4	3,4	2,53
Функция 8	0,8	1,2	2,2	1,30
Функция 9	1,6	2,5	3,0	2,43
Функция 10	1,6	4,0	6,0	3,93
ИТОГО	20,0	28,3	38,6	28,62

Как обычно, выясняется, что точечные оценки разработчиков из табл. 10.4 в действительности были оценками лучшего случая.

10.4. Создание осмысленных общих оценок для лучшего и худшего случаев

Если для получения общих оценок лучшего и худшего случаев нельзя использовать суммирование, что же делать? В статистике часто используется одно стандартное приближение, в соответствии с которым 1/6 диапазона от минимума до максимума

считается равной стандартному отклонению. Приближение основано на предположении, что минимум достигается с вероятностью 0,135 %, а максимум с вероятностью 99,86 % включает все возможные значения.

Вычисление сводных оценок лучшего и худшего случаев при малом количестве задач (упрощенная формула стандартного отклонения)

При малом количестве задач (10 и менее) оценки лучшего и худшего случаев могут базироваться на упрощенной формуле стандартного отклонения. Сначала раздельно суммируются оценки лучших и худших случаев, после чего по ним вычисляется стандартное отклонение по формуле:

ФОРМУЛА № 4

$$\text{СтандартноеОтклонение} = (\text{СуммаOценокХудшегоСлучая} + \text{СуммаOценокЛучшегоСлучая})/6.$$

Если взять 1/6 от диапазона между 20,0 и 38,6 из табл. 10.5, результат будет равен одному стандартному отклонению в распределении результатов данного проекта. Одна шестая разности составляет 3,1. Далее в зависимости от необходимой достоверности вычисляется статистически состоятельная оценка. В табл. 10.6 приведены формулы для ее вычисления.

Таблица 10.6. Вычисление достоверных оценок с использованием стандартного отклонения

Достоверность	Формула
2 %	ОжидаемыйСлучай - (2 × СтандартноеОтклонение)
10 %	ОжидаемыйСлучай - (1,28 × СтандартноеОтклонение)
16 %	ОжидаемыйСлучай - (1 × СтандартноеОтклонение)
20 %	ОжидаемыйСлучай - (0,84 × СтандартноеОтклонение)
25 %	ОжидаемыйСлучай - (0,67 × СтандартноеОтклонение)
30 %	ОжидаемыйСлучай - (0,52 × СтандартноеОтклонение)
40 %	ОжидаемыйСлучай - (0,25 × СтандартноеОтклонение)
50 %	ОжидаемыйСлучай
60 %	ОжидаемыйСлучай + (2 × СтандартноеОтклонение)
70 %	ОжидаемыйСлучай + (0,52 × СтандартноеОтклонение)
75 %	ОжидаемыйСлучай + (0,67 × СтандартноеОтклонение)
80 %	ОжидаемыйСлучай + (0,84 × СтандартноеОтклонение)
84 %	ОжидаемыйСлучай + (1 × СтандартноеОтклонение)
90 %	ОжидаемыйСлучай + (1,28 × СтандартноеОтклонение)
98 %	ОжидаемыйСлучай + (2 × СтандартноеОтклонение)

Так, с применением этой методики статистически достоверная на 75 % оценка вычисляется по формуле ОжидаемыйСлучай (29 недель) + 0,67 × СтандартноеОтклонение, то есть 29 + (0,67 × 3,1), что составит 31 неделю.

Почему я назвал 31 неделю вместо 31,1? Потому что базовая оценка задачи имеет точность не более двух значащих цифр, не говоря уже о трех, так что относитесь к результатам трезво. Вероятно, в нашем примере оценка даже в 31 неделю приведет к завышению точности результата, и 30 может оказаться более содержательным числом.

СОВЕТ № 49

Используйте упрощенную формулу стандартного отклонения для вычисления содержательных оценок лучшего и худшего случаев в проектах, состоящих из 10 и менее задач.

Вычисление сводных оценок лучшего и худшего случаев при большом количестве задач (сложная формула стандартного отклонения)

Если в проекте задействовано более 10 задач, формула стандартного отклонения в предыдущем разделе становится несостоительной, и вам придется искать более сложное решение. Научный метод начинается с применения формулы стандартного отклонения к каждой из отдельных оценок (Stutzke 2005):

ФОРМУЛА № 5

$$\text{ОтдельноеСтандартноеОтклонение} = \frac{(\text{ОтдельнаяОценкаХудшегоСлучая} + \text{ОтдельнаяОценкаЛучшегоСлучая})}{2}$$

По этой формуле вычисляется значение в столбце «Стандартное отклонение» в табл. 10.7. Затем по довольно сложным математическим формулам определяется стандартное отклонение сводной оценки.

1. Вычислите стандартное отклонение для каждой задачи или функции по указанной формуле.
2. Вычислите квадрат стандартного отклонения каждой задачи (эта величина, называемая *дисперсией*, приведена в правом столбце табл. 10.7).
3. Просуммируйте дисперсии.
4. Вычислите квадратный корень из суммы.

В таблице сумма дисперсий равна 1,22, а квадратный корень из нее равен 1,1; это значение и определяет стандартное отклонение сводной оценки.

СОВЕТ № 50

Используйте сложную формулу стандартного отклонения для вычисления содержательных сводных оценок лучшего и худшего случаев в проектах, состоящих из 10 и более задач.

Вспомним, что стандартное отклонение, полученное предыдущим способом, составило 3,1. Новый подход дает ответ 1,10 при тех же данных — различия более чем заметные! Как такое может быть?

Как выясняется, дело в различиях между четкостью и точностью. Проблема формулы ($\text{ОценкаХудшегоСлучая} - \text{ОценкаЛучшегоСлучая}$) / 6 заключается в том, что с точки зрения статистики вы предполагаете, что человек, создавший оценки лучшего и худшего случаев, включил шестикратный диапазон стандартного

отклонения от лучшего случая к худшему. Если бы это было правдой, диапазон оценки должен был учитывать 99,7 % всех возможных результатов. Другими словами, из 1000 оценок только три фактических результата будут выпадать за пределы оценочных диапазонов!

Таблица 10.7. Пример вычисления стандартного отклонения по сложной формуле

Функция	Недель на завершение			
	Лучший случай	Худший случай	Стандартное отклонение	Дисперсия (квадрат стандартного отклонения)
Функция 1	1,6	3,0	0,233	0,054
Функция 2	1,8	4,0	0,367	0,134
Функция 3	2,0	4,2	0,367	0,134
Функция 4	0,8	1,6	0,133	0,018
Функция 5	3,8	5,2	0,233	0,054
Функция 6	3,8	6,0	0,367	0,134
Функция 7	2,2	3,4	0,200	0,040
Функция 8	0,8	2,2	0,233	0,054
Функция 9	1,6	3,0	0,233	0,054
Функция 10	1,6	6,0	0,733	0,538
ИТОГО	20,0	38,6	—	1,22
Стандартное отклонение	—	—	—	1,1

Конечно, это совершенно смехотворное предположение. В нашем примере два результата из 10 вышли за пределы оценочных диапазонов. Как было показано в главе 1, 90%-я достоверность в представлении большинства людей в действительности ближе к 30%-й достоверности. После некоторой тренировки человек может научиться задавать диапазоны, содержащие правильное значение в 70 % случаев, но у оценщика нет даже малейшей возможности выдавать оценки с достоверностью 99,7 %.

В более реалистичном способе вычисления стандартного отклонения по лучшему и худшему случаю каждый отдельный диапазон делится на число, более близкое к 2, нежели к 6. С точки зрения статистики деление на 2 подразумевает, что диапазоны оценщика будут включать фактический результат в 68 % случаев; эта цель достигается практикой.

В табл. 10.8 представлены значения делителей в зависимости от процента фактических результатов, попадающих в ваши диапазоны оценки.

После этого число из таблицы включается в сложную формулу стандартного отклонения:

ФОРМУЛА № 6

$$\text{ОтделочноеСтандартноеОтклонение} = (\text{ОтдельнаяОценкаХудшегоСлучая} + \text{ОтдельнаяОценкаЛучшегоСлучая}) / \text{ДелительИзТаблицы10_8}$$

Таблица 10.8. Делители, используемые при сложном вычислении стандартного отклонения

Если этот процент фактических результатов попадает в оценочный диапазон...	...Используйте это число в качестве делителя при вычислении стандартного отклонения отдельных оценок
10 %	0,25
20 %	0,51
30 %	0,77
40 %	1,0
50 %	1,4
60 %	1,7
70 %	2,1
80 %	2,6
90 %	3,3
99,7 %	6,0

СОВЕТ № 51

Не используйте деление диапазона между лучшим и худшим случаем на 6 при вычислении отклонений стандартных оценок. Выбирайте делитель в зависимости от точности диапазонов ваших оценок.

Создание сводных оценок для лучшего и худшего случаев

В рассматриваемом нами сценарии фактические результаты группы попадали в диапазон от лучшего до худшего случая 8 раз из 10. Из табл. 10.9 следует, что группы с 80%-м попаданием должны использовать делитель 2,6. В табл. 10 показаны результаты пересчета стандартных отклонений, дисперсий и сводного стандартного отклонения после деления диапазонов на 2,6 вместо 6.

Таблица 10.9. Пример вычисления стандартного отклонения по сложной формуле

Функция	Недель на завершение			
	Лучший случай	Худший случай	Стандартное отклонение	Дисперсия (квадрат стандартного отклонения)
Функция 1	1,6	3,0	0,538	0,290
Функция 2	1,8	4,0	0,846	0,716
Функция 3	2,0	4,2	0,846	0,716
Функция 4	0,8	1,6	0,308	0,095
Функция 5	3,8	5,2	0,538	0,290
Функция 6	3,8	6,0	0,846	0,716
Функция 7	2,2	3,4	0,462	0,213
Функция 8	0,8	2,2	0,538	0,290
Функция 9	1,6	3,0	0,538	0,290
Функция 10	1,6	6,0	1,692	2,864
ИТОГО	20,0	38,6	—	6,48
Стандартное отклонение	—	—	—	2,55

Этот вариант дает для сводной оценки стандартное отклонение в 2,55 недели. Для вычисления достоверных оценок используется оценка ожидаемого случая в 28,6 недели из табл. 10.5 и множители из табл. 10.6. В результате будет получен набор достоверных оценок, показанных в табл. 10.10.

Таблица 10.10. Пример достоверных оценок, вычисленных по стандартному отклонению

Достоверность	Оценка объема работ
2 %	23,5
10 %	25,4
16 %	26,1
20 %	26,5
25 %	26,9
30 %	27,3
35 %	27,7
40 %	28,0
50 %	28,6
60 %	29,3
70 %	30,0
75 %	30,3
80 %	30,8
84 %	31,2
90 %	31,8
98 %	33,7

В зависимости от аудитории содержимое этой таблицы может подвергаться значительной правке. Тем не менее в некоторых ситуациях будет полезно указать, что хотя вычисление для лучшего случая дает оценку в 20 человеко-недель, вероятность преодоления порога в 23,5 недели составляет всего 2 %, а вероятность преодоления порога в 26,9 недели — 25 %.

Как обычно, перед представлением оценок следует учитывать их четкость — я обычно указываю 24 недели вместо 23,5 и 27 недель вместо 26,9.

Предупреждения по поводу достоверных оценок

Общая проблема только что описанной методики заключается в том, что оценки общего случая должны быть точными — иначе говоря, они должны быть действительно вероятными на 50 %. Занижение в этих оценках должно встречаться с такой же частотой, как и завышение. Если обнаружится, что оценка завышается чаще, чем занижается, значит, оценка не является вероятной на 50 % и не может использоваться в ожидаемых случаях. Если ожидаемые случаи неточны, то и их сумма не будет точной.

В главе 9 представлены рекомендации для повышения точности отдельных оценок.

СОВЕТ № 52

Направьте усилия на повышение точности оценок ожидаемого случая. Если отдельные оценки точны, то их объединение не создаст проблем. С другой стороны, если отдельные оценки неточны, объединение станет возможным лишь после того, как вы найдете способ повысить их точность.

Дополнительные ресурсы

Humphrey, Watts S. «A Discipline for Software Engineering». Reading, MA: Addison-Wesley, 1995. В приложении А книги Хамфри приведена короткая сводка стандартных методов, применяемых при оценке программных проектов.

Stutzke, Richard D. «Estimating Software-Intensive Systems», Upper Saddle River, NJ: Addison-Wesley, 2005. Глава 5 содержит более подробные описания статистических показателей, представленных в этой главе. В главе 20 описан процесс создания WBS.

Gonick, Larry and Woollcott Smith. «The Cartoon Guide to Statistics». New York, NY: Harper Collins, 1993. Несмотря на несерьезное название, это весьма авторитетное (и занятное) введение в методы статистического анализа. По мнению многих читателей, иллюстрации помогают лучше освоить статистические концепции. Впрочем, некоторым читателям труднее сосредоточиться на графике, нежели на тексте, и это усложняет понимание материала.

Larsen, Richard J. and Morris L. Marx. «An Introduction to Mathematical Statistics and Its Applications, Third Edition». Upper Saddle River, NJ: Prentice Hall, 2001. Доступно и традиционное введение в математическую статистику... насколько оно может быть доступным для данной темы. В конце концов, каждому, кто хочет использовать в своей работе статистические методы, рано или поздно придется заняться математическими вычислениями.

Динамика оценки по аналогии в зависимости от количества аналогов. Стандартные оценки для каждого из трех методов показаны на рисунке, показанном в главе 10, 19.

Оценка по аналогии



Область применения методов этой главы

Оценка по аналогии

Что оценивается	Размер, объем работ, сроки, функциональность
Размер проекта	МСБ
Стадия разработки	Ранняя–поздняя
Итеративный или последовательный стиль	Оба
Возможная точность	Средняя

Некая (вымыселенная) корпорация Gigacorp собиралась начать работу над Triad 1.0 – обновленной версией успешного пакета торговых презентаций AccSellerator 1.0. Майк был назначен руководителем проекта Triad 1.0, и ему требовалась хотя бы приблизительная оценка для предстоящего собрания по планированию продаж. Он созвал своих работников.

«Как вы знаете, мы приступаем к разработке Triad 1.0, – сказал Майк. – Техническая сторона очень близка к AccSellerator 1.0. Мне представляется, что проект будет чуть крупнее AccSellerator 1.0, но не намного».

«База данных будет намного больше, – откликнулась Дженнифер. – Но пользовательский интерфейс останется примерно таким же».

«Графиков и отчетов тоже будет гораздо больше, чем в AccSellerator 1.0, но основные классы очень похожи; думаю, количество классов останется прежним», – сказал Джо.

«Я думаю примерно так же, – сказал Майк. – Пожалуй, на основании этих данных можно предварительно рассчитать объем работы. В моих записках указано, что общий объем работ по предыдущей системе составил 30 человеко-месяцев. Как вы полагаете, какой должна быть разумная оценка объема работы по новой системе?»

А как *вы* думаете, каким будет объем работы по новой системе?

11.1. Основные принципы оценки по аналогиям

В приведенном примере Майк использует методику, называемую оценкой по аналогии. В ее основе лежит простая мысль — точную оценку нового проекта можно получить, сравнивая новый проект с похожим прошлым проектом.

Несколько сотен разработчиков представили свои оценки для проекта Triad. Их оценки были разбросаны в диапазоне от 30 до 144 человеко-месяцев, а среднее значение составило 53 человека-месяца. Стандартное отклонение оценок составило 24, или 46 % от среднего ответа. Нехорошо! Некоторое структурирование процесса принесет существенную пользу.

Вот как выглядит базовый процесс оценки по аналогии, который даст более точный результат.

1. Получите подробные данные об итоговом размере, объеме работ и затратах для предыдущего аналогичного проекта. Если возможно, получите данные, фрагментированные по функциональности, структуре трудозатрат (WBS) или другой схеме декомпозиции.
2. Шаг за шагом сравните размер нового проекта с размером старого проекта.
3. Постройте оценку размера нового проекта в процентах от размера старого проекта.
4. Создайте оценку объема работ, руководствуясь размером нового проекта по сравнению с размером предыдущего проекта.
5. Следите за тем, чтобы показатели старого и нового проектов базировались на единых предположениях.

СОВЕТ № 53

Оценивайте новые проекты, сравнивая их с похожими прошлыми проектами. Постарайтесь разбить оценку минимум на пять составляющих.

Давайте рассмотрим эту процедуру на примере Triad.

Шаг 1. Получение подробных данных об итоговом размере, объеме работ и затратах для предыдущего аналогичного проекта

После первого собрания Майк попросил участников проекта Triad собрать более конкретную информацию о размере старой системы и относительном объеме функциональности старой и новой систем. Когда сбор данных был завершен, Майк поинтересовался результатами: «Вы собрали те данные, о которых мы говорили на прошлой неделе?»

«Конечно, Майк, — ответила Дженифер. — Проект AccSellerator 1.0 состоял из 5 подсистем. Структура проекта была примерно такой:

База данных	5000 строк кода
Пользовательский интерфейс	14 000 строк кода
Диаграммы и отчеты	9000 строк кода
Библиотека классов	4500 строк кода
Бизнес-логика	11 000 строк кода
Итого	43 500 строк кода

У нас также имеется общая информация об общем количестве элементов в каждой подсистеме. Вот что мы нашли:

База данных	10 таблиц
Пользовательский интерфейс	14 веб-страниц
Диаграммы и отчеты	10 диаграмм + 8 отчетов
Библиотека классов	15 классов
Бизнес-логика	???

Мы постарались определить аналогичные показатели для новой системы. Вот что у нас получилось:

База данных	14 таблиц
Пользовательский интерфейс	19 веб-страниц
Диаграммы и отчеты	14 диаграмм + 16 отчетов
Библиотека классов	15 классов
Бизнес-логика	???

«Между большинством подсистем старого и нового проектов существует прямое соответствие, но с бизнес-логикой возникли проблемы, — сказала Дженнифер. — Мы полагаем, что она будет сложнее, чем в старой системе, но не уверены в цифровом выражении. Мы обсудили эту проблему, и по нашему ощущению, бизнес-логика по крайней мере на 50 % сложнее, чем в старой системе».

«Отличная работа, — сказал Майк. — Теперь у меня есть все необходимое для вычисления оценки к собранию. Завтра я немного повожусь с цифрами и покажу вам результат перед собранием».

Шаг 2. Сравнение размера нового проекта с аналогичным прошлым проектом

Подробная информация о предыдущем проекте дает нам все необходимое для создания содержательной оценки методом аналогии. Группа Triad уже выполнила шаг 1, «Получение подробных данных об итоговом размере, объеме работ и затратах для предыдущего аналогичного проекта». Далее собранные данные шаг за шагом сравниваются с соответствующими показателями нового проекта. Результаты сравнения показаны в табл. 11.1.

Таблица 11.1. Сравнение размеров подсистем между AccSellerator 1.0 и Triad 1.0

Подсистема	Фактический размер в AccSellerator 1.0	Оцениваемый размер в Triad 1.0	Множитель
База данных	10 таблиц	14 таблиц	1,4
Пользовательский интерфейс	14 веб-страниц	19 веб-страниц	1,4
Диаграммы и отчеты	10 диаграмм + 8 отчетов	14 диаграмм + 16 отчетов	1,7
Библиотека классов	15 классов	15 классов	1,0
Бизнес-логика	???	???	1,5

Записать данные в столбцах 2 и 3 несложно — проблема в том, что делать с множителем в столбце 4. Основной принцип вам уже знаком: сначала подсчет, затем вычисления и в последнюю очередь субъективная оценка. Если найти какой-нибудь счетный показатель, результат будет лучше, чем при использовании субъективной оценки.

С множителями 1,4 для базы данных, 1,4 для пользовательского интерфейса и 1,0 для библиотеки классов вроде бы все понятно.

С множителем 1,7 для диаграмм и отчетов дело обстоит сложнее. Должны ли мы присвоить диаграммам тот же весовой коэффициент, что и отчетам? Возможно, диаграммы требуют больше работы, чем отчеты, и наоборот. Если бы кодовая база AccSellerator 1.0 была доступна, мы могли бы свериться с ней и определить, присвоить ли диаграммам и отчетам одинаковые или разные весовые коэффициенты. В данном примере мы будем полагать, что веса одинаковы. Это допущение следует документировать, чтобы позднее при необходимости процедуру оценки можно было бы воссоздать.

Бизнес-логика также порождает проблемы. Группа не нашла никакого показателя, поэтому наша оценка в этой области поконится на более шатком основании, чем в других областях. Для определенности мы согласимся с оценкой, что бизнес-логика Triad будет примерно на 50 % сложнее бизнес-логики AccSellerator.

Шаг 3. Построение оценки размера нового проекта в процентах от размера старого проекта

На шаге 3 метрики размеров из разных областей приводятся к общей единице измерения — в нашем случае это строки программного кода. Преобразование позволяет выполнить общесистемное сравнение размеров между AccSellerator и Triad. Таблица 11.2 показывает, как это происходит.

Размеры подсистем AccSellerator в строках кода были получены по данным, собранным на шаге 1. Множители определились в ходе шага 2. Оцениваемый размер подсистемы Triad попросту равен произведению размера подсистемы в AccSellerator и множителя. Общий размер системы в строках кода становится основой для вычисления оценки объема работ, которая, в свою очередь, станет основой для оценок сроков и затрат.

Таблица 11.2. Вычисление предполагаемого размера Triad 1.0 на основании размера AccSellerator

Подсистема	Размер кода AccSellerator 1.0	Множитель в	Оценка размера Triad 1.0
База данных	5000	1,4	7000
Пользовательский интерфейс	14 000	1,4	19 600
Диаграммы и отчеты	9000	1,7	15 300
Библиотека классов	4500	1,0	4500
Бизнес-логика	11 000	1,5	16 500
ИТОГО	43 500	—	62 900

Шаг 4. Создание оценки объема работ на основе размера нового проекта, полученного сравнением с размером предыдущего проекта

Теперь мы располагаем достаточной информацией для вычисления оценки объема работ. Это делается так, как показано в табл. 11.3.

Таблица 11.3. Итоговое вычисление объема работ для Triad 1.0

Показатель	Значение
Размер Triad 1.0	62 000 строк кода
Размер AccSellerator 1.0	~43 500 строк кода
Соотношение размеров	= 1,45
Объем работ для AccSellerator 1.0	× 30 человеко-месяцев
Оценка объема работ для Triad 1.0	= 44 человеко-месяца

Разделив размер проекта Triad на размер AccSellerator, мы получаем соотношение размеров двух систем. Умножение его на объем работ AccSellerator дает оценку для Triad — 44 человеко-месяца.

Оценка вычисленная и оценка субъективная — две совершенно разные вещи. В вычислениях вы получаете точечную оценку, но можете представить ее в диапазонном виде (см. главу 22).

Я предложил оценщикам, создавшим исходные оценки для Triad, воспользоваться этой процедурой, и их результаты стали более точными и согласованными. Стандартное отклонение результатов составило всего 7 % вместо 46 %, даже с учетом неопределенности, связанной с диаграммами, отчетами и бизнес-логикой.

Шаг 5. Проверка согласованности предположений между старым и новым проектами

Проверяйте свои предположения на каждом шаге. Впрочем, полноценная проверка некоторых предположений становится возможной только после завершения оценки. Далее перечислены основные источники рассогласования.

- Существенно различающиеся размеры старого и нового проектов, то есть различия более чем в 3 раза (см. раздел 5.1). В нашем случае размеры отличаются всего в 1,45 раза; этого недостаточно, чтобы беспокоиться об издержках масштаба.
- Разные технологии (например, один проект написан на C#, а другой на Java).
- Существенные различия в квалификации отдельных участников (в малых проектах) или целых групп (в больших проектах). Небольшие различия вполне допустимы, а часто неизбежны.
- Существенные различия в типе программы. Например, если старая система была внутренним интрасетевым проектом, а новая представляет собой критическую встроенную систему, сравнивать их было бы некорректно.

11.2. Замечания по поводу неопределенности в оценке Triad

Информация, доступная для оценки бизнес-логики, была довольно туманной. Стоит ли повысить количество бизнес-правил, чтобы обеспечить консервативность оценки? Нет, не стоит. Целью оценки должна быть *точность*, а не консерватизм. Как только вы перестаете стремиться к точности, в оценку из разных источников начинают проникать субъективные смещения, и полезность оценки падает. Правильной реакцией на неопределенность должно быть не смещение оценки, а гарантия того, что любая базовая неопределенность будет точно отражена в оценке. Если вы абсолютно уверены в количестве бизнес-правил, то оценку можно было бы считать точной до $\pm 10\%$. Вероятно, с учетом неопределенности в бизнес-логике уровень неопределенности стоит повысить до ($+25\%$, -10%).

Более качественное решение проблемы неопределенности, обусловленной составляющей бизнес-логики, заключается в использовании диапазонного фактора бизнес-логики вместо отдельного числа. Вместо точечного коэффициента 1,5 можно оценить фактор с 50%-м отклонением (иначе говоря, диапазон от 0,75 до 2,25). В этом случае вместо точечной оценки в 44 человека-месяца вычисления дают диапазон от 38 до 49 человеко-месяцев.

В отличие от оценок, созданных описанным способом, в оценках «монолитных» (то есть не использующими декомпозицию) неопределенность в одной области может распространяться в другие области. Скажем, если в бизнес-логике существует 50%-я неопределенность, оценщик может применить ее ко всей оценке, а не только к четверти, относящейся к бизнес-правилам. Если применить то же 50%-е отклонение ко всей оценке, то мы получим диапазон от 22 до 66 месяцев, вместо диапазона от 38 до 49 месяцев. Умение идентифицировать факторы неопределенности и их влияние на оценку способствует сужению общего диапазона оценки.

COBET № 54

Не решайте проблему неопределенности смещением оценки. Постарайтесь отразить неопределенность в самой оценке.

Неопределенность оценки, планы и обязательства

В конечном счете неопределенность в оценке перейдет в *планы и обязательства* проекта. Поскольку планы и обязательства направлены на обеспечение максимальной производительности, а не на точность, отрегулируйте обязательства в консервативном направлении, руководствуясь неопределенностью заложенной в них оценки.

Наноформам, подготувани відповідно до методу, що використовується в табліці 11.3. Після цього вони висушуються в сушильному шкафі при температурі 40 °C протягом 12 годин. Далі вони оброблюються засобами очищення та дезактивування.

Раздел на размер проекта 1 гига из размер AccSellerator, мы получаем соотношение размеров двух систем. Умножение его на объем работ AccSellerator дает со временем количество, которое необходимо для выполнения проекта. Время выполнения проекта определяется количеством рабочих единиц и количеством рабочего времени. Оценка на выполнение и оценка существующая – два совершенно различные вещи. Время выполнения оценки – это время, необходимое для выполнения проекта, но может не соответствовать реальному времени выполнения (если в оценке не учтены факторы, влияющие на реальное время выполнения). Оценка на выполнение и оценка существующая – это две совершенно различные вещи. Время выполнения оценки – это время, необходимое для выполнения проекта, но может не соответствовать реальному времени выполнения (если в оценке не учтены факторы, влияющие на реальное время выполнения).

12

Опосредованные оценки

Очень малый 117 14 859 127

При классификации новой функциональности оценщик вычисляет показатель, который коррелируется с количеством тестовых случаев, потребуемых для реализации функции. Для этого он определяет количество тестов, необходимое для реализации функции, и делит это значение на общее количество тестов, необходимое для реализации всей функциональности. Результатом является коэффициент, называемый коэффициентом определенности. Он может быть от 0 до 1. Чем выше этот коэффициент, тем выше вероятность того, что функция будет реализована в соответствии с требованиями.

Область применения методов этой главы

	Нечеткая логика	Стандартные компоненты	Абстрактные рейтинги	Метод футболки
Что оценивается	Размер, функциональность	Размер, объем работ	Размер, объем работ, сроки, функциональность	Размер, объем работ, сроки, функциональность
Размер проекта	– С Б	М С Б	М С Б	– С Б
Стадия разработки	Ранняя	Ранняя–средняя	Ранняя–средняя	Ранняя
Итеративный или последовательный стиль	Последовательный	Оба	Оба	Последовательный
Возможная точность	Средняя	Средняя	Средняя–высокая	—

Как правило, оценщик не может взглянуть на описание функции и точно оценить: «Ее реализация будет состоять из 253 строк кода». Также трудно прямо оценить, сколько тестовых случаев потребует ваш проект, сколько в нем отыщется дефектов, сколько будет задействовано классов и т. д.

Семейство методов оценки, объединенных общим названием *опосредованных методов*, помогают решать подобные проблемы. При опосредованной оценке сначала идентифицируется *посредник* — показатель, коррелированный с оцениваемым, который проще вычисляется или подсчитывается (или становится доступным на более ранней стадии проекта, чем интересующая вас величина). Скажем, если вы хотите оценить количество тестовых сценариев, может оказаться, что оно коррелируется с количеством требований; если оценивается размер в строках кода, может оказаться, что оно коррелируется с количеством функций (с поправкой на категорию размера).

После того как показатель-посредник будет идентифицирован, вы оцениваете или подсчитываете его значение, а затем по формулам, основанным на исторических данных, преобразуете его в нужную оценку.

В этой главе обсуждаются некоторые из самых полезных опосредованных методов. Суть каждого метода заключается в том, что целое содержит более достоверную информацию, чем отдельные части. Таким образом, эти методы полезны для создания оценок уровня проекта или итерации, но не для создания подробных оценок с разбиением на задачи или отдельные функции.

12.1. Нечеткая логика

Метод, называемый *нечеткой логикой*, применяется для оценки размера проекта в строках кода (Putnam and Myers 1992, Humphrey 1995). Оценщики обычно могут классифицировать функции по таким группам: «очень малые», «малые», «средние», «большие» и «очень большие». Затем по историческим данным мы определяем, сколько строк кода в среднем требует очень малая функция, сколько — малая и т. д. В табл. 12.1 показан пример создания подобных оценок.

Таблица 12.1. Пример использования нечеткой логики для оценки размера программы

Размер функции	Среднее количество строк кода на функцию	Количество функций	Оценка количества строк кода
Очень малая	127	22	2794
Малая	253	15	3795
Средняя	500	10	5000
Большая	1014	30	30 420
Очень большая	1998	27	53 946
ИТОГО		104	95 955

Содержимое столбца «Среднее количество строк кода на функцию» базируется на исторических данных вашей организации и фиксируется до начала оценки. В столбце «Количество функций» содержится количество функций в каждой категории размера. Содержимое столбца «Оценка количества строк кода» вычисляется по двум другим столбцам. Как видно из таблицы, оценка состоит из 5 значащих цифр, что выходит за пределы точности базовых данных. Я бы представлял эту оценку как «96 000 строк кода» или даже «100 000 строк кода» (то есть с 1–2 значащими цифрами), чтобы не создавать ложного впечатления высокой точности.

Как получить средние размеры

Метод нечеткой логики лучше всего работает при калибровке размеров по историческим данным вашей организации. Обычно размеры смежных категорий должны отличаться как минимум в два раза. Некоторые эксперты рекомендуют использовать 4-кратные различия (Putnam and Myers 1992).

Начальные средние показатели размеров создаются на основе данных, получаемых классификацией одной или нескольких законченных систем. Проанализируйте старую систему и отнесите каждую функцию к одной из категорий —

очень малая, малая, средняя, большая или очень большая. Затем подсчитайте суммарное количество строк кода в функциях одной категории и разделите его на количество функций; вы получите средний размер в строках кода одной функции данной категории. Пример приведен в табл. 12.2.

Таблица 12.2. Пример вычисления среднего размера функции в строках кода

Размер	Количество функций	Общее количество строк кода	Средний размер функции
Очень малый	117	14 859	127
Малый	71	17 963	253
Средний	56	28 000	500
Большой	169	171 366	1014
Очень большой	119	237 762	1998

Данные в таблице приведены исключительно для примера. Используйте собственные показатели, базирующиеся на исторических данных вашей организации.

СОВЕТ № 55

Используйте нечеткую логику для оценки размера программы в строках кода.

Классификация новой функциональности

При классификации новой функциональности по размерам очень важно, чтобы ваши предположения относительно того, что считать очень малой, малой, средней, большой или очень большой функцией, в оценке соответствовали предположениям, которые использовались при исходном вычислении средних размеров. Это можно сделать тремя способами:

- Поручите исходные вычисления тем же людям, которые будут заниматься созданием оценок.
- Проведите обучение оценщиков, чтобы они правильно классифицировали функции.
- Документируйте критерии классификации, чтобы оценщики последовательно применяли их в своей работе.

Как не следует использовать нечеткую логику

У статистики имеется один интересный аспект: статистические сводные показатели часто оказываются более достоверными, чем отдельные элементы данных. Как упоминалось в главе 10, закон больших чисел наделяет сводные оценки точностью, превосходящей точность отдельных оценок. Целое действительно становится чем-то большим, нежели простой совокупностью частей.

При использовании нечеткой логики важно помнить об этом явлении. Работа нечеткой логики основана на наших предположениях о том, что если каждая из 71 мелких функций в прошлом состояла в среднем из 235 строк кода, то и в будущем каждая из 15 мелких функций будет состоять примерно из 253 строк. Тем не менее

из этого вовсе *не следует*, что любая отдельная функция будет действительно состоять из 253 строк. Размеры отдельных мелких функций могут лежать в диапазоне от 50 до 1000 строк кода. Следовательно, хотя монолитная оценка, полученная с применением нечеткой логики, может быть на удивление точной, не следует расширять методику для оценки размеров отдельных функций.

По тем же причинам нечеткая логика хорошо работает при 20 и более составляющих. Если вы не располагаете данными по крайней мере по 20 функциям, статистика этого метода будет работать некорректно, и вам лучше поискать другой метод.

Расширения нечеткой логики

Нечеткая логика также может использоваться для оценки объема работ — разумеется, если вы располагаете соответствующими данными. Пример показан в табл. 12.3.

Таблица 12.3. Пример использования нечеткой логики для оценки объема работ

Размер	Среднее количество человеко-дней на функцию	Количество функций	Оценка объема работ (в человеко-днях)
Очень малый	4,2	22	92,4
Малый	8,4	15	126
Средний	17	10	170
Большой	34	30	1020
Очень большой	67	27	1809
ИТОГО	—	104	3217

Данные в таблице приведены исключительно для примера. Используйте собственные значения количества человеко-дней на функцию, базирующиеся на исторических данных вашей организации.

Окончательная оценка в 3217 человеко-дней снова получилась излишне четкой. Ее желательно упростить до 3200 человеко-дней, 3000 человеко-дней или 13 человеко-лет (для 250 рабочих дней в году). Также всегда следует рассматривать возможность представления результата в виде диапазона — скажем, от 10 до 15 человеко-лет. Такая оценка создает совершенно иное ощущение точности, чем 3217 человеко-дней.

12.2. Стандартные компоненты

Если вы разрабатываете несколько программ со сходной архитектурой, для оценки размера можно воспользоваться методом *стандартных компонентов*. Сначала необходимо найти подходящие элементы для подсчета в предыдущих системах. Более конкретные рекомендации зависят от работы, которую вы хотите выполнить. Типичная система содержит динамические и статические веб-страницы, таблицы баз данных, бизнес-логику, диаграммы, диалоговые окна, отчеты и т. д.

После идентификации стандартных компонентов вычисляется среднее количество строк кода на компонент в прошлых системах. В табл. 12.4 показан пример исторических данных для стандартных компонентов.

Таблица 12.4. Пример исторических данных по количеству строк кода на стандартный компонент

Стандартный компонент	Количество строк кода на компонент
Динамические веб-страницы	487
Статические веб-страницы	58
Таблицы баз данных	2437
Отчеты	288
Бизнес-правила	8327

Собрав исторические данные, оцените количество стандартных компонентов в новой программе и вычислите размер новой программы по старым размерам. Пример показан в табл. 12.5.

Таблица 12.5. Пример использования стандартных компонентов для создания оценки размера

Стандартный компонент	Строк программного кода на компонент	Минимально возможное число	Наиболее вероятное число	Максимально возможное число	Оцениваемое число	Оценка в строках кода
Динамические веб-страницы	487	11	25	50	26,8	13 052
Статические веб-страницы	58	20	35	40	33,3	1931
Таблицы баз данных	2437	12	15	20	15,3	37 286
Отчеты	288	8	12	20	12,7	3658
Бизнес-правила	8327	—	1	—	1	8327
ИТОГО	—	—	—	—	—	64 254

В столбцах 3–5 вводятся ваши оценки. Столбец 3 содержит минимальное количество компонентов, которые, по вашему представлению, может содержать проект. Например, для динамических веб-страниц в данном примере этот показатель равен 11. В следующем столбце вводится число, по вашему мнению наиболее вероятное (в нашем примере 25). Затем в столбце 5 вводится максимально возможное количество (50). Оценка в столбце 6 вычисляется по формуле PERT (Program Evaluation and Review Technique), обсуждавшейся в главе 9. Вот как выглядит эта формула для оценки количества компонентов:

ФОРМУЛА № 7

Ожидаемое Количество Компонентов = [Минимум + (4 × Наиболее Вероятное Количество) + Максимум]/6.

В нашем примере оцениваемое количество динамических веб-страниц оказывается равным $[11 + (4 \times 25) + 50]/6 = 26,8^1$.

Как и прежде, данные в таблице приведены исключительно для примера. Используйте собственные значения, базирующиеся на исторических данных.

Использование стандартных компонентов с процентилями

В одной из разновидностей представленной методики вместо количества компонентов оценивается количество процентиелей. Для этого вам также потребуется достаточное количество исторических проектов для вычисления осмысленных процентиелей (иначе говоря, по меньшей мере 10 исторических проектов, а в идеале ближе к 20). Но если вы располагаете таким объемом исторических данных, вместо оценки количества можно оценить предполагаемое отклонение от среднего значения по каждому из компонентов. В табл. 12.6 показано, как выглядит составляемая таблица.

Таблица 12.6. Пример таблицы с эталонными данными для стандартных компонентов

Стандартные компоненты	Очень малый (10-й)	Малый (25-й)	Средний (50-й)	Большой (75-й)	Очень большой (90-й)
Динамические веб-страницы	5105	6037	12 123	24 030	35 702
Статические веб-страницы	1511	1751	2111	2723	3487
Таблицы баз данных	22 498	30 020	40 027	45 776	47 002
Отчеты	1518	2518	3530	5833	5533
Бизнес-правила	7007	7534	8509	10 663	12 111

Данные в таблице определяют размер стандартных компонентов по отношению к другим проектам, выполнявшимся вашей организацией. Согласно таблице, у 10 % проектов организации динамические веб-страницы содержали 5105 строк кода и менее, у 50 % проектов статические веб-страницы содержали 2111 строк кода и менее, у 75 % проектов бизнес-правила содержали 10 663 строки кода и менее и т. д. После заполнения эталонной таблицы классифицируйте размер, предполагаемый по каждому из стандартных компонентов, и найдите соответствующее количество строк кода в табл. 12.6. Пример показан в табл. 12.7.

Как видно из таблицы, вы ожидаете, что оцениваемый проект будет содержать средние динамические веб-страницы по сравнению с другими проектами, выполнявшимися вашей организацией; статические страницы будут больше среднего, таблицы баз данных — меньше среднего и т. д.

¹ Некоторые люди путаются с тем, нужно ли делить на 6 или на какое-нибудь другое число. Комментарии о других делителях из главы 10 относятся к вычислению стандартного отклонения. В данном случае по формуле вычисляется ожидаемое значение, а не стандартное отклонение, поэтому предупреждение на него не распространяется.

Таблица 12.7. Пример использования стандартных компонентов для оценки размера

Стандартный компонент	Классификация размера	Оценка в строках кода (из табл. 12.6)
Динамические веб-страницы	Средний	12 123
Статические веб-страницы	Большой	2723
Таблицы баз данных	Малый	30 020
Отчеты	Очень малый	1518
Бизнес-правила	Средний	8509
итого	—	54 893

Такой подход дает оценку в 54 893 строки кода. Как и прежде, при представлении оценки желательно упростить ее до 55 000 или 60 000 строк (то есть до одной или двух значащих цифр).

Ограничения метода стандартных компонентов

К преимуществам метода стандартных компонентов следует отнести то, что он требует минимальных усилий с вашей стороны; собственно, все сводится к интуитивной оценке размера стандартных компонентов в новой системе и поиску по таблице. Немного времени потребуется на конструирование и сопровождение справочной таблицы (вроде тех, что представлены в табл. 12.4 и 12.6).

Метод стандартных компонентов не базируется на подсчете, поэтому он нарушает общий принцип «сначала подсчет, затем вычисления и в последнюю очередь субъективная оценка». Впрочем, он связывает оценки с какими-то обоснованными показателями и поэтому в отдельных случаях может пригодиться.

В целом, хотя метод стандартных компонентов нельзя назвать лучшим методом для поздней стадии проекта, он помогает свести к минимуму усилия по созданию ранних оценок, которые все равно подвержены высокой неточности из-за конуса неопределенности.

СОВЕТ № 56

Рассматривайте метод стандартных компонентов как средство для получения оценки размера с минимальными усилиями на ранних стадиях проекта.

12.3. Абстрактные рейтинги

Другой разновидностью метода нечеткой логики является метод *абстрактных рейтингов*, изначально ассоциировавшийся с экстремальным программированием (Cohn 2006). В целом он сходен с нечеткой логикой, однако у него есть свои интересные и полезные особенности, из-за которых его стоит обсудить отдельно.

При использовании метода абстрактных рейтингов группа просматривает список пользовательских историй (или требований, или функций), которые она намеревается реализовать, и присваивает каждой истории некоторый размер, изменяемый в *пунктах*. В этом отношении абстрактные рейтинги сходны с нечеткой

логикой, однако историям обычно назначаются числовые значения по одной из числовых последовательностей, представленных в табл. 12.8.

Таблица 12.8. Наиболее распространенные шкалы абстрактных рейтингов

Шкала	Пункты
Степени 2	1, 2, 4, 8, 16
Числа Фибоначчи	1, 2, 3, 5, 8, 13

В результате оценки строится список наподобие представленного в табл. 12.9.

Таблица 12.9. Список историй с присвоенными пунктами

История	Пункты
История 1	2
История 2	1
История 3	4
История 4	8
История 60	2
ИТОГО	180

На этой стадии пункты особой пользы не принесут, поскольку являются чисто абстрактными единицами — они не преобразуются в количество строк программного кода, человеко-дни или календарное время. Важнейшая концепция абстрактных рейтингов заключается в том, что группа оценивает все истории одновременно и по одной шкале, а способ оценки в значительной степени свободен от смещения.

Затем группа планирует итерацию, включая в свои планы выдачу некоторого числа абстрактных пунктов. В основу планов может быть заложено предположение, что абстрактные рейтинги преобразуются в некоторый объем работ, но на столь ранней стадии проекта это всего лишь предположение.

После завершения первой итерации у команды появляется возможность для составления реальной оценки. Группа смотрит, сколько абстрактных пунктов она выдала, какой объем работ и календарное время для этого потребовался, и проводит предварительную калибровку преобразования абстрактных пунктов в объем работ и календарное время. Пример представлен в табл. 12.10.

Таблица 12.10. Данные итерации 1 и начальная калибровка

Данные итерации 1
Выдано 27 абстрактных пунктов
Затрачено 12 человеко-недель
Затрачено 3 календарных недели
Предварительная калибровка
Объем работ = 27 абстрактных пунктов/12 человеко-недель = 2,25 пункта/человеко-неделя
Сроки = 27 абстрактных пунктов/3 календарные недели = 9 пунктов/календарную неделю

На основании исходной калибровки руководитель проекта составляет оценку оставшейся части проекта, базирующуюся на абстрактных данных. Пример показан в табл. 12.11.

Таблица 12.11. Исходное планирование оставшейся части проекта

Данные итерации 1

Предположения (из предварительной калибровки)

Объем работ = 2,25 пункта/человеко-неделю

Срок = 9 пунктов/календарную неделю

Размер проекта = 180 пунктов

Предварительная оценка для всего проекта

Объем работ = 180 пунктов/2,25 пункта/человеко-неделю = 80 человеко-недель

Сроки = 180 пунктов/9 пунктов/календарную неделю = 20 календарных недель

Конечно, в вычислениях из табл. 12.11 подразумевается, что в будущих итерациях группа останется неизменной, а планирование не учитывает влияние праздников, отпусков и т. д. Тем не менее в итеративных проектах она обеспечивает очень раннее планирование результата проекта на основании абстрактных данных того же проекта.

Исходная оценка проекта должна уточняться по данным из последующих итераций. Чем короче итерации, тем скорее у вас появятся данные, которые могут использоваться для оценки оставшейся части проекта, и тем достовернее будут эти оценки.

СОВЕТ № 57

Метод абстрактных рейтингов используется для получения ранней оценки объема работ и сроков проекта, и в основу его закладываются данные того же проекта.

О шкале абстрактных рейтингов

В методе нечеткой логики используется описательная шкала размеров: очень малый, малый и т. д. В методе абстрактных рейтингов используются степени 2 или числа Фибоначчи. Что лучше?

На числовой шкале отношения между числами предполагают, что измеряемые величины также находятся в соответствующем отношении. Например, если абстрактные рейтинги присваиваются по шкале Фибоначчи, шкала 1, 2, 3, 5, 8, 13 наводит на мысль, что объем работ для выполнения истории из 5 пунктов составит $5/3$ от объема работ для истории из 3 пунктов, а история из 13 пунктов будет выполняться почти в 4 раза дольше, чем история из 3 пунктов.

Такие отношения оказываются «палкой о двух концах». Если вы приняли необходимые меры, чтобы 13-пунктовая история действительно занимала примерно в 4 раза больший объем работ, чем 3-пунктовая — замечательно. Это означает, что вы сможете вычислить средний объем работ на пункт абстрактного рейтинга (см. ранее), умножить его на общее количество пунктов и получить осмысленный результат.

Однако достижение такого уровня точности потребует большой осторожности при назначении пунктов историй. Кроме того, фактические данные проекта должны гарантировать, что оцениваемые соотношения действительно встречаются на практике.

Если не принять мер к обеспечению точности числовых соотношений, обусловленных шкалой Фибоначчи или степенью 2, то результаты, вычисленные на базе абстрактных рейтингов, могут оказаться менее состоятельными, чем кажется на первый взгляд. Применение числовой шкалы подразумевает возможность выполнения числовых операций: умножения, сложения, вычитания и т. д. Но если базовые отношения не состоятельны (то есть 13-пунктовая история в действительности не требует в 13/3 больших усилий, чем 3-пунктовая), то выполнение математических операций с числом 13 оказывается ничуть не более осмысленным, чем выполнение математических операций с оценкой «большой» или «очень большой».

В табл. 12.2 представлен другой способ описания этой проблемы.

Таблица 12.12. Пример того, что может произойти при деформации числовой шкалы

Классификация в пунктах	Количество историй	Условные пункты	Предполагаемое соотношение	Фактическое соотношение	Реальные пункты (по данным)
«1»	4	«4»	1	2	4
«2»	7	«14»	2	2,5	18
«3»	5	«15»	3	3	15
«5»	5	«25»	5	7	35
«8»	12	«96»	8	11	132
«13»	2	«26»	13	17	34
итого	43	«180»	—	—	238

В этом примере числовая шкала заставила нас поверить, что 180 пунктов образуют разумную оценку общего объема работ, но реальный объем оказался примерно на 30 % выше.

СОВЕТ № 58

Будьте внимательны при вычислении оценок, в которых используются числовые рейтинговые шкалы. Убедитесь в том, что числовые категории на шкале действительно ведут себя как числа, а не как отвлеченные категории вроде «малый», «средний» или «большой».

12.4. Метод футболки

Участникам проекта, не обладающим технической квалификацией, часто приходится принимать решения относительно объема проекта в широкой части конуса неопределенности. Хороший оценщик не станет предоставлять четкую оценку, пока проект находится на этой стадии. Специалисты по продажам и маркетингу запротестуют: «Как я могу сказать, нужна мне эта функция или нет, если я не знаю, сколько она стоит?» На что хороший оценщик скажет: «А я не могу ответить,

сколько она стоит, пока мы не получим более подробные требования». Ситуация заходит в тупик.

Чтобы выйти из тупика, необходимо понять, что целью оценки программного проекта является не идеальная точная оценка, а оценка, достаточно точная для обеспечения эффективного управления проектом. В данном случае у вас не требуют оценку в человеко-часах, а хотят узнать, с чем можно сравнить функцию по размерам — с мышью, кроликом, собакой или слоном? Это замечание приводит нас к чрезвычайно полезной методике оценки, называемой *методом футболки*.

В этом методе разработчики классифицируют затраты на разработку каждой функции по отношению к другим функциям: малые, средние, большие или очень большие. Параллельно отделы маркетинга, продаж, обслуживания клиентов и другие стороны, не обладающие технической квалификацией, классифицируют коммерческую ценность каждой функции по той же шкале. Затем два набора оценок объединяются, как показано в табл. 12.13.

Таблица 12.13. Пример использования метода футболки для классификации функций по коммерческой ценности и затратах на разработку

Функция	Коммерческая ценность	Затраты на разработку
Функция А	Большая	Малая
Функция В	Малая	Большая
Функция С	Большая	Большая
Функция D	Средняя	Средняя
Функция Е	Средняя	Большая
Функция F	Большая	Средняя
Функция G	Малая	Малая
Функция H	Малая	Средняя
...		
Функция ZZ	Малая	Малая

Определение подобных отношений между коммерческой ценностью и затратами на разработку позволяет специалисту, не обладающему технической квалификацией, высказывать мнения вроде следующего: «Если реализация функции В требует больших затрат, она мне не нужна, потому что функция обладает малой коммерческой ценностью». Возможность принятия таких решений на ранней стадии работы над функцией чрезвычайно полезна. Если бы вместо этого функция прошла через определенную работу по постановке требований, проектированию, выработке архитектуры и т. д., может оказаться, что вы напрасно расходуете усилия на функцию, не оправданную по затратам. В области разработки программного обеспечения ранний ответ «Нет» дорогостоящий. Метод футболки позволяет принимать решения об исключении функций на ранней стадии работы над проектом и исключить дальнейшее перемещение этих функций по конусу неопределенности.

Что стоит сохранить, а что выбросить? Обсуждать эту тему будет гораздо проще, если список функций будет хотя бы приблизительно отсортирован по отношению затраты/выигрыш. Как правило, для этого функциям присваивается показатель

чистой стоимости (еще одна абстрактная метрика), основанный на комбинации затрат на разработку и коммерческой ценности. В табл. 12.14 представлена одна из возможных схем присваивания чистой стоимости. Вы можете использовать эту схему или разработать другую, которая бы более точно отражала специфику вашей среды.

Таблица 12.14. Вычисление чистой стоимости по соотношению затрат к коммерческой ценности

Коммерческая ценность	Затраты на разработку			
	Очень большие	Большие	Средние	Малые
Очень большая	0	4	6	7
Большая	-4	0	2	3
Средняя	-6	-2	0	1
Малая	-7	-3	-1	0

Наличие подобной таблицы позволит включить третий столбец в исходную таблицу затрат/коммерческой ценности (см. табл. 12.13) и отсортировать ее по чистой стоимости, как показано в табл. 12.15.

Таблица 12.15. Пример сортировки оценок, полученных методом футболки, по приблизительной чистой стоимости

Функция	Коммерческая ценность	Затраты на разработку	Приблизительная чистая стоимость
Функция A	Б	М	3
Функция F	Б	С	-2
Функция C	Б	Б	0
Функция D	С	С	0
Функция G	М	М	0
Функция ZZ	М	М	0
Функция H	М	С	-1
Функция E	С	Б	-2
...			
Функция B	М	Б	-3

Помните, что последний столбец содержит приближенные данные. Я не предлагаю вычислить список и провести «пороговую линию» — сортировка по приближенной коммерческой ценности полезна прежде всего тем, что она позволяет быстро дать ответ «определенко да» для функций в верхней части списка и «определенко нет» для функций в нижней части списка. Основное обсуждение перемещается в середину списка, где оно будет наиболее продуктивным.

СОВЕТ № 59

Используйте метод футболки, чтобы помочь не-техническим сторонам проекта принять решения по включению или исключению тех или иных функций в широкой части конуса неопределенности.

12.5. Другие применения опосредованных методов

Приведенные в этой главе примеры показывают, как применять опосредованные методы для оценки объема работ и затрат. Аналогичные методы могут использоваться для оценки количества тестовых сценариев, дефектов, страниц документации или любых других показателей, которые проще оценивать опосредованно, нежели напрямую.

СОВЕТ № 60

Используйте опосредованные методы для оценки количества тестовых сценариев, дефектов, страниц документации и любых других показателей, которые трудно оценивать напрямую.

Как описано в главе 7, возможности выбора показателя почти не ограничены. В этой главе представлено лишь несколько конкретных примеров. Если вы считаете, что в вашей среде имеется другой, более адекватный показатель размера проекта, используйте его вместо методов нечеткой логики, стандартных компонентов или абстрактных рейтингов.

СОВЕТ № 61

Подсчитывайте тот показатель, который проще всего считается и обеспечивает максимальную точность в вашей среде. Соберите калибровочные данные и используйте их для создания оценки, адаптированной к вашей среде.

Дополнительные ресурсы

Cohn, Mike. «Agile Estimating and Planning». Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2006. В книге Кона приводится более подробное обсуждение абстрактных рейтингов, включая факторы планирования и методы оценки.

Humphrey, Watts S. «A Discipline for Software Engineering». Reading, MA: Addison-Wesley, 1995. В главе 5 книги Хамфри рассматривается опосредованный метод, названный автором «методом PROBE», а также более подробно рассматриваются статистические методы, на которых он основывается. Кроме того, в главе 5 обсуждается метод нечеткой логики.

рис. 13.1 составляет 35 %. У оценок, прошедших обсуждение в группах, ошибка снизилась до 30 %. В данном примере 92 % групповых оценок проходили индивидуальные оценки по точности, начиная с 10 % в группе, что соответствует однократному сокращению ошибки.

стартовом уровне на 100 % выше, чем на конечном этапе. На конечном этапе ошибка снизилась до 30 %. В данном примере 92 % групповых оценок проходили индивидуальные оценки по точности, начиная с 10 % в группе, что соответствует однократному сокращению ошибки.

Экспертные оценки в группах 13

Область применения методов этой главы

	Групповое обсуждение	Широкополосный Дельфийский метод
Что оценивается	Размер, объем работ, сроки, функциональность	Размер, объем работ, сроки, функциональность
Размер проекта	– С Б	– С Б
Стадия разработки	Ранняя–средняя	Ранняя
Итеративный или последовательный стиль	Оба	Последовательный
Возможная точность	Средняя	Средняя

Метод группового обсуждения полезен при оценке проекта на ранней стадии или при большом количестве факторов неопределенности. В этой главе представлен метод неструктурированной групповой экспертной оценки (групповое обсуждение) и структурированный метод, называемый «широкополосным Дельфийским методом».

13.1. Групповое обсуждение

Простой способ повышения точности оценок, созданных отдельными экспертами, заключается в проведении группового анализа оценок. При обсуждении оценок в группах я требую соблюдения трех простых правил.

- Каждый участник группы оценивает фрагменты проекта по отдельности, после чего группа встречается для сравнения оценок. Обсудите различия в оценках и постарайтесь понять причины различий. Работайте до тех пор, пока не достигнете единого мнения по поводу верхней и нижней границ оценок.

- Не ограничивайтесь принятием усредненной оценки. Вычислить среднее значение можно, но непременно обсудите различия между отдельными результатами. Не принимайте вычисленное среднее значение автоматически.
- Согласуйте оценку, которая будет принята всей группой. Если обсуждение зайдет в тупик, не прибегайте к голосованию. Обсудите различия и добейтесь консенсуса от всех членов группы.

Несмотря на простоту, эта методика улучшает точность оценки. На рис. 13.1 показаны результаты по 24 группам оценщиков, с которыми я работал.

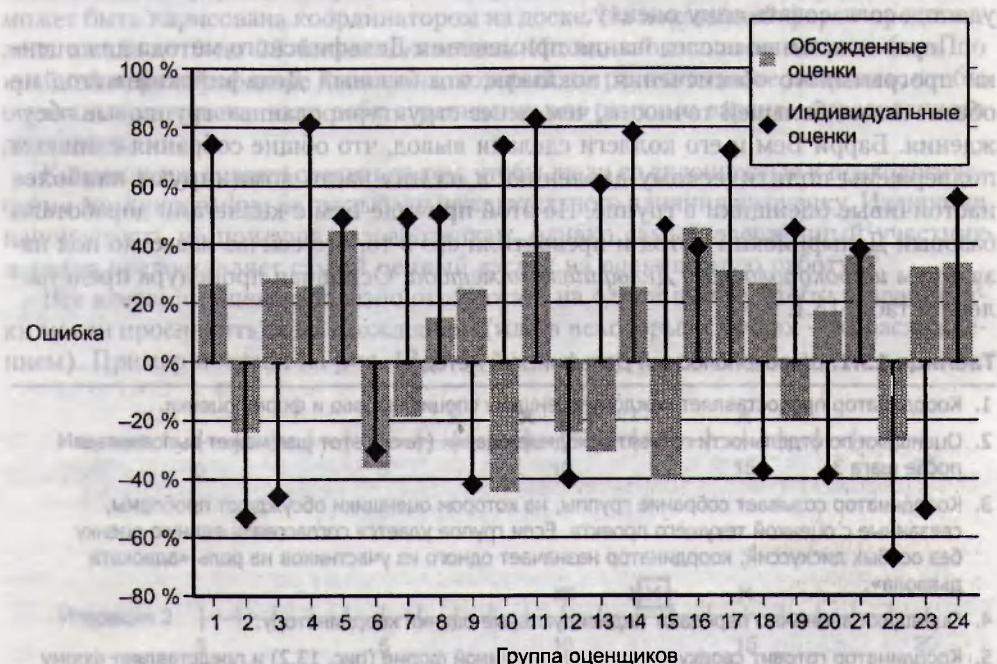


Рис. 13.1. Простое обсуждение индивидуальных оценок существенно улучшает их точность

Средняя величина относительной ошибки для индивидуальных оценок на рис. 13.1 составляет 55 %. У оценок, прошедших обсуждение в группах, она уменьшилась до 30 %. В данном примере 92 % групповых оценок превосходили индивидуальные оценки по точности, причем обсуждение в среднем приводило к двукратному сокращению ошибки.

СОВЕТ № 62

Используйте групповое обсуждение для повышения точности оценки.

Какое количество экспертов считать достаточным? Исследования в других областях показали, что от 3 до 5 экспертов с разной квалификацией обычно оказывается достаточно (Libby and Blashfield 1978, Jørgensen 2002).

Также может быть полезно пригласить экспертов с подготовкой в разных областях или пользующихся разными методиками (Armstrong 2001, Jørgensen 2002).

13.2. Широкополосный Дельфийский метод

Широкополосный Дельфийский метод относится к структурированным методам групповой оценки. Исходный Дельфийский метод был разработан в Rand Corporation в конце 1940-х годов для прогнозирования технологических тенденций (Boehm 1981). Его название происходит от древнегреческого города Дельфы, где находился оракул. В базовом варианте Дельфийского метода несколько экспертов создают независимые оценки, а затем встречаются до тех пор, пока им не удастся согласовать одну оценку.

Первоначальные исследования применения Дельфийского метода для оценки программного обеспечения показали, что базовый Дельфийский метод не обеспечивает большей точности, чем менее структурированные групповые обсуждения. Барри Бем и его коллеги сделали вывод, что общие собрания слишком подвержены политическому давлению, и на них часто доминируют наиболее настойчивые оценщики в группе. По этой причине Бем с коллегами доработали базовый Дельфийский метод и превратили его в то, что сейчас известно под называнием *широкополосного Дельфийского метода*. Основная процедура представлена в табл. 13.1.

Таблица 13.1. Широкополосный Дельфийский метод

1. Координатор предоставляет каждому оценщику спецификацию и форму оценки.
2. Оценщики по отдельности готовят исходные оценки (также этот шаг может выполняться после шага 3).
3. Координатор созывает собрание группы, на котором оценщики обсуждают проблемы, связанные с оценкой текущего проекта. Если группе удается согласовать единую оценку без особых дискуссий, координатор назначает одного из участников на роль «адвоката дьявола».
4. Оценщики анонимно передают индивидуальные оценки координатору.
5. Координатор готовит сводку оценок в итеративной форме (рис. 13.2) и представляет форму оценщикам, чтобы они видели, как выглядят их оценки в сравнении с оценками других участников группы.
6. Координатор организует встречу, на которой оценщики обсуждают расхождения в оценках.
7. Оценщики проводят анонимное голосование по принятию средней оценки. Если хотя бы один из оценщиков проголосует отрицательно, процедура возвращается к шагу 3.
8. Окончательной считается точечная оценка, полученная по Дельфийской процедуре. Также окончательная оценка может представлять собой диапазон, созданный в ходе обсуждения, а точечная оценка представляет ожидаемый случай.

Источник: Адаптировано по материалам «Software Engineering Economics» (Boehm 1981).

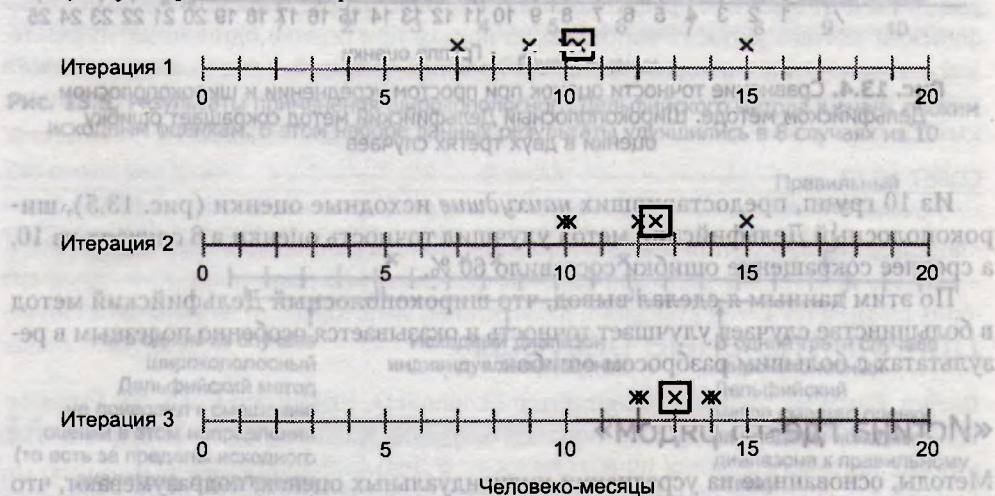
Шаги 3–7 выполняются в личном общении, на групповых собраниях, по электронной почте или в чате. Выполнение их на электронном уровне помогает сохранить анонимность экспертов. Итерации шагов 3–7 выполняются немедленно или в пакетном режиме, в зависимости от критичности оценки по времени и доступности оценщиков.

**Рис. 13.2.** Форма оценки для широкополосного Дельфийского метода

Оценочная форма, изображенная на рис. 13.2, может быть бумажной или же может быть нарисована координатором на доске. На показанной форме представлена шкала от 0 до 20 человеко-месяцев. Исходный диапазон должен быть по крайней мере втрое шире диапазона, который вы рассчитываете получить, чтобы оценщики не чувствовали себя ограниченными рамками заранее определенного диапазона.

Координатор должен следить за тем, чтобы люди со склонностью к психологическому доминированию не оказывали нежелательного влияния на оценку. Излишняя напористость не присуща разработчикам, однако самый сдержаный участник нередко предоставляет самый ценный взгляд на оцениваемую работу.

Все итерации оценок полезно отображать на одной шкале, чтобы разработчики могли проследить за их схождением (или в некоторых случаях — за расхождением). Пример показан на рис. 13.3.

**Рис. 13.3.** Оценочная форма широкополосного Дельфийского метода после трех итераций

После итерации 3 группа согласилась на оценку в виде диапазона от 12 до 14 месяцев, при этом ожидаемое значение составляет 13 человеко-месяцев.

Эффективность широкополосного Дельфийского метода

Я собрал данные по применению широкополосного Дельфийского метода в очень трудной оценочной проблеме. На рис. 13.4 представлены данные об ошибках,

полученных при простом усреднении исходных оценок, по сравнению с ошибками при использовании широкополосного Дельфийского метода.

Мой опыт применения широкополосного Дельфийского метода показывает, что он снижает ошибку примерно на 40 % по сравнению со усредненной оценкой группы. Из групп, участвовавших в моем исследовании, две трети предоставили более точный ответ при использовании широкополосного Дельфийского метода, нежели при простом усреднении.

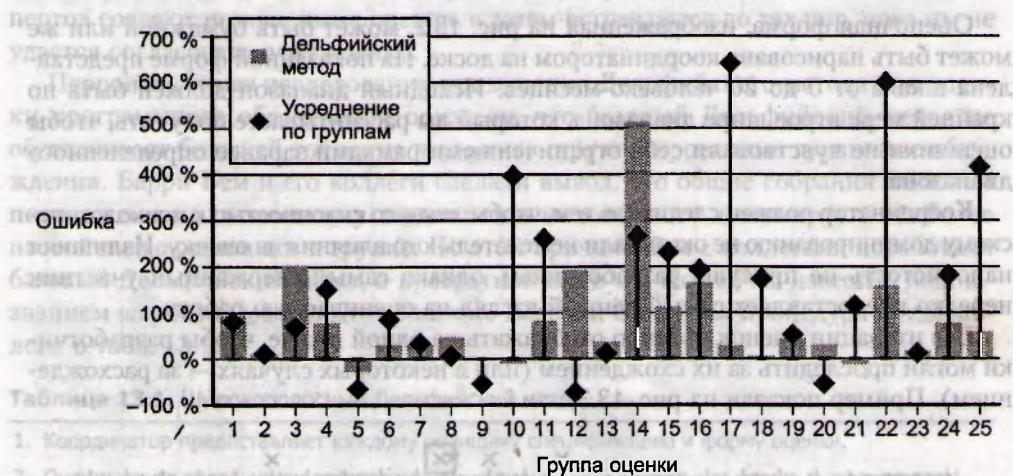


Рис. 13.4. Сравнение точности оценок при простом усреднении и широкополосном Дельфийском методе. Широкополосный Дельфийский метод сокращает ошибку оценки в двух третях случаев

Из 10 групп, предоставивших *наихудшие* исходные оценки (рис. 13.5), широкополосный Дельфийский метод улучшил точность оценки в 8 случаях из 10, а среднее сокращение ошибки составило 60 %.

По этим данным я сделал вывод, что широкополосный Дельфийский метод в большинстве случаев улучшает точность и оказывается особенно полезным в результатах с большим разбросом ошибок.

«Истина где-то рядом»

Методы, основанные на усреднении индивидуальных оценок, подразумевают, что правильный ответ лежит где-то в диапазоне между самой низкой и самой высокой оценкой. Тем не менее в моих данных широкополосного Дельфийского метода исходные оценочные диапазоны 20 % групп не включали правильный ответ. Это означает, что усреднение таких исходных оценок в принципе не может привести к точному результату.

Вероятно, самое интересное свойство широкополосного Дельфийского метода заключалось в том, что одна треть групп, у которых исходный диапазон не включал правильный ответ, немедленно соглашались с оценкой, лежащей за пределами их исходного диапазона ближе к правильному ответу. Другими словами, в таких

группах широкополосный Дельфийский метод работает лучше, чем лучшая из индивидуальных оценок. Ситуация продемонстрирована на рис. 13.6. Обратите внимание: ни одна из групп не согласилась с окончательной оценкой, которая была хуже наихудшей индивидуальной оценки.

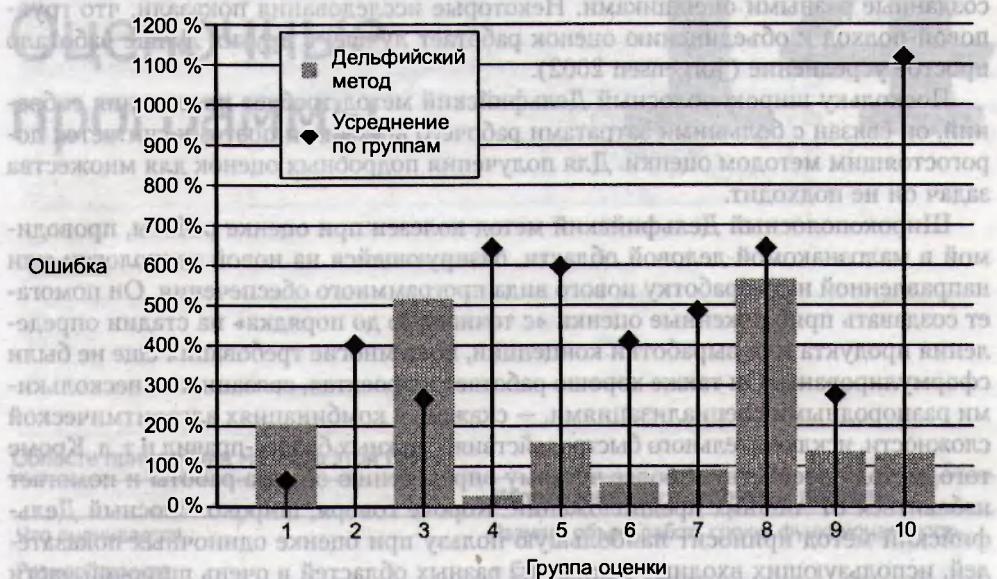


Рис. 13.5. Результаты применения широкополосного Дельфийского метода к очень плохим исходным оценкам. В этом наборе данных результаты улучшились в 8 случаях из 10

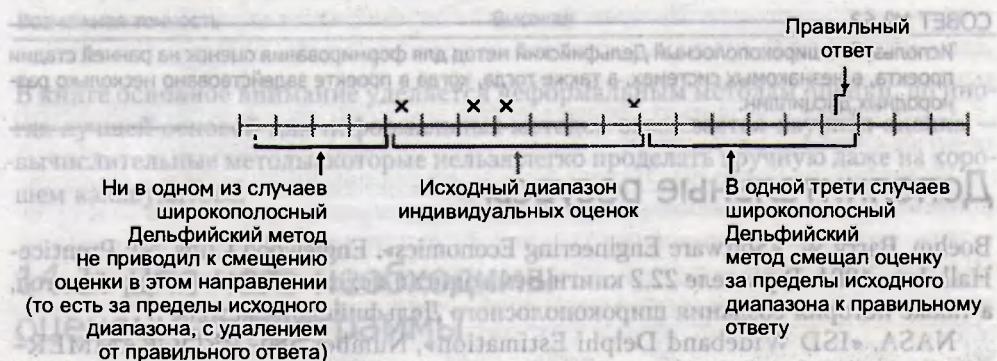


Рис. 13.6. Примерно в трети случаев широкополосный Дельфийский метод помогает группам, у которых правильный ответ не входил в исходный оценочный диапазон, выйти за пределы диапазона и приблизиться к правильному ответу

Когда применять широкополосный Дельфийский метод

В сценарии с трудной оценкой, описанном в этой главе, широкополосный Дельфийский метод сократил среднюю ошибку оценки с 290 до 170 %. Ошибки в 290 % и 170 % чрезвычайно велики, но это характерно для оценок, создаваемых в широкой

части конуса неопределенности. Тем не менее сокращение ошибки на 40 % весьма ценно, будь то уменьшение с 290 до 170 % или с 50 до 30 %.

Хотя мои данные, казалось бы, однозначно одобряют широкополосный Дельфийский метод, в отрасли сейчас изучается вопрос о том, как объединять оценки, созданные разными оценщиками. Некоторые исследования показали, что групповой подход к объединению оценок работает лучше; в других лучше работало простое усреднение (Jørgensen 2002).

Поскольку широкополосный Дельфийский метод требует проведения собраний, он связан с большими затратами рабочего времени и поэтому считается дорогостоящим методом оценки. Для получения подробных оценок для множества задач он не подходит.

Широкополосный Дельфийский метод полезен при оценке работы, проводимой в малознакомой деловой области, базирующейся на новой технологии или направленной на разработку нового вида программного обеспечения. Он помогает создавать приближенные оценки «с точностью до порядка» на стадии определения продукта или выработки концепции, пока многие требования еще не были сформулированы. Он также хорошо работает в проектах, связанных с несколькими разнородными специализациями, — скажем, в комбинациях алгоритмической сложности, исключительного быстродействия, сложных бизнес-правил и т. д. Кроме того, метод способствует более четкому определению объема работы и помогает избавиться от лишних предположений. Короче говоря, широкополосный Дельфийский метод приносит наибольшую пользу при оценке одиночных показателей, использующих входные данные из разных областей в очень широкой части конуса неопределенности. В подобных неопределенных ситуациях широкополосный Дельфийский метод способен оказать неоценимую помощь.

СОВЕТ № 63

Используйте широкополосный Дельфийский метод для формирования оценок на ранней стадии проекта, в незнакомых системах, а также тогда, когда в проекте задействовано несколько разнородных дисциплин.

Дополнительные ресурсы

Boehm, Barry W. «Software Engineering Economics». Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981. В разделе 22.2 книги Бема описан исходный Дельфийский метод, а также история создания широкополосного Дельфийского метода.

NASA, «ISD Wideband Delphi Estimation», Number 580–PROGRAMMER–016–01, September 1, 2004, <http://software.gsfc.nasa.gov/AssetsApproved/PA1.2.1.2.pdf>. Документ описывает разновидность широкополосного Дельфийского метода, используемую Центром управления полетами NASA.

Wiegers, Karl. «Stop Promising Miracles», Software Development, February 2000. В статье Вигерса описана разновидность широкополосного Дельфийского метода.

и 90%-й достоверностью, но в данном случае нужно всего использовать эти данные в общем случае, или они являются неизвестными, недостоверны, в гипотетической форме.

Оценочные программы

14

Область применения методов этой главы

Использование оценочных программ

Что оценивается	Размер, объем работ, сроки, функциональность
Размер проекта	С Б
Стадия разработки	Ранняя–средняя
Итеративный или последовательный стиль	Оба
Возможная точность	Высокая

В книге основное внимание уделяется неформальным методам оценки, но иногда лучшей основой для неформальных методов оказывается научная оценка — вычислительные методы, которые нельзя легко проделать вручную даже на хорошем калькуляторе.

14.1. Для чего необходимы оценочные программы

Оценочные программы позволяют выполнить некоторые операции, связанные с оценкой, которые трудно проделать вручную.

Моделирование результатов проекта. Оценочные программы способны выполнять сложное статистическое моделирование, которое помогает ключевым сторонам проекта представить объем работы. На рис. 14.1 показан пример имитации результатов программного проекта.

Сплошные черные линии обозначают медианы (50/50) срока и объема работ. Пунктирные черные линии представляют 25-й и 75-й процентили результатов.

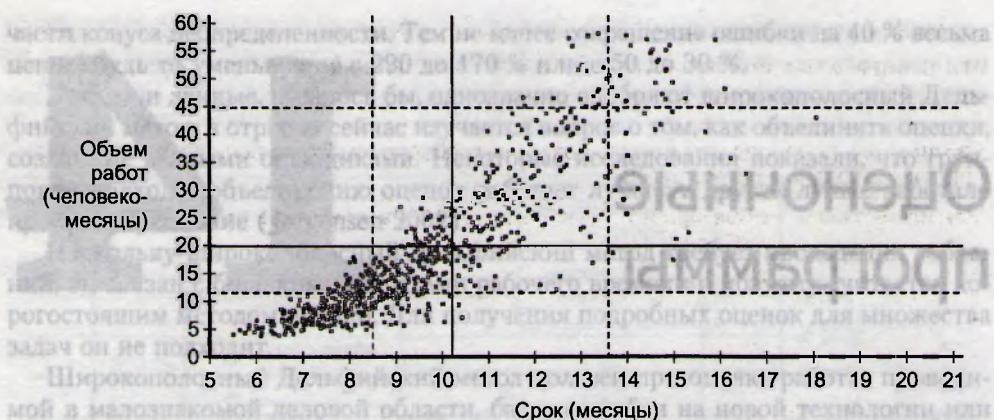


Рис. 14.1. Компьютерное моделирование 1000 результатов проекта (выходные данные от созданной приблизительной оценки Construx Estimate)

Оценочная программа учитывает несколько источников изменчивости:

- различия в производительности;
- различия в размере программы, возможно с декомпозицией на несколько модулей;
- различия в темпах комплектования персонала.

Для каждого моделируемого результата программа использует метод статистического моделирования, называемый методом Монте-Карло, и моделирует один возможный результат для конкретной комбинации производительности, размера и состояния персонала. По трем факторам на диаграмме строится одна точка. Для создания всей диаграммы программа повторяет цикл 1000 раз. Сами понимаете, делать это вручную никому не захочется!

В некоторых программах (более дорогих, чем Construx Estimate) применяются более сложные технологии.

Вероятностный анализ. В главе 1 объясняются опасности оценок с формулировками типа «с уверенностью на 90 %». Когда оценка создается на базе субъективного суждения, подобные выражения изначально подвержены ошибкам. Но если оценка генерируется специальной программой, откалиброванной по историческим данным, процентные показатели несут более содержательную информацию. Например, на рис. 14.1 объем работ в 45 человеко-месяцев имеет 75 %-ю вероятность, потому что 75 % смоделированных проектов заняли менее 45 человеко-месяцев.

В табл. 14.1 представлен пример вероятностного анализа объема работ, вычисленного оценочной программой. «Номинал», упоминаемый в заголовке третьего столбца, относится к оценке в 20 человеко-месяцев, вероятной на 50 %.

Самая интересная особенность таблицы заключается в том, что для повышения достоверности с 70 до 80 % или с 80 до 90 % требуется очень большой рост объема работ. При оценках, основанных на субъективном суждении, лишь немногие оценщики умножают свои номинальные оценки на 6 для вычисления оценки

с 90%-й достоверностью, но в данном случае нужно именно это (не используйте эти данные в общем случае; они были вычислены по конкретным предположениям, введенным в оценочную программу).

Таблица 14.1. Пример вероятностей различных объемов работ по проекту, вычисленных при помощи оценочной программы

Вероятность	Объем работ будет меньше либо равен	Расхождение с номинальной оценкой
10 %	7	-64 %
20 %	10	-50 %
30 %	13	-37 %
40 %	16	-20 %
50 %	20	0 %
60 %	26	30 %
70 %	37	84 %
80 %	58	189 %
90 %	142	611 %

На рис. 14.2 показано графическое представление данных в таблице.

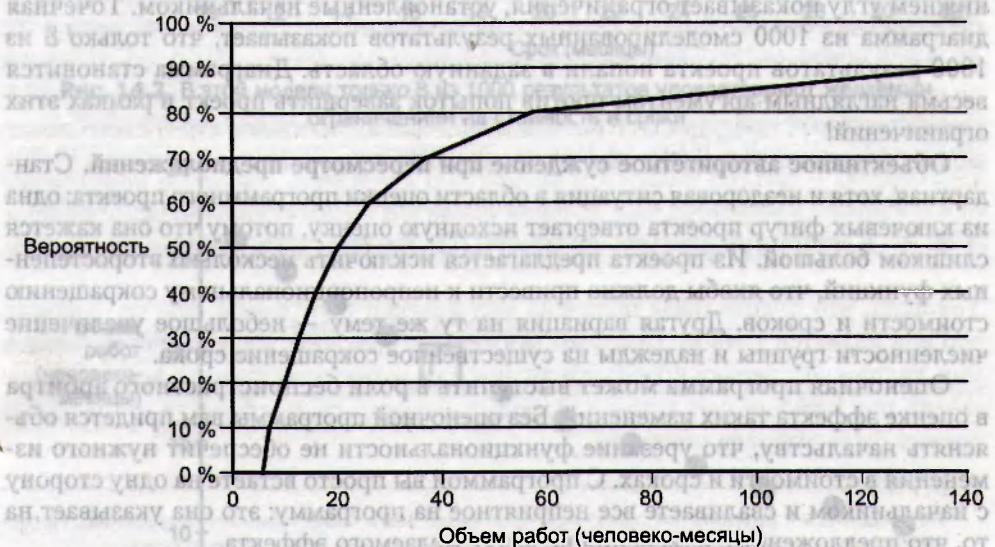


Рис. 14.2. Пример вероятных результатов проекта по данным оценочной программы

Учет издержек масштаба. Оценочные программы автоматически учитывают различия в размерах проектов и их влияние на производительность.

Учет непредвиденного расширения требований. Проблема расширения требований встречается настолько часто, что в большинстве коммерческих оценочных программ может учитываться в ходе проекта.

Оценка вторичных аспектов разработки программного обеспечения. Оценочные программы обычно поддерживают возможность оценки размера документации по требованиям, размера архитектурной документации, количества тестовых сценариев, количества дефектов, среднего времени работы до отказа и множества других величин.

Вычисление плановых показателей и интеграция с программами планирования. Некоторые оценочные программы позволяют задавать объемы работ на постановку требований, проектирование, конструирование, тестирование и отладку, а также делить проект на нужное количество итераций. Подобные вычисления крайне утомительно производить вручную, но при наличии должного инструментария они выполняются легко. Некоторые программы также интегрируются с Microsoft Project и другими системами планирования проектов.

Анализ «что-если». Оценочные программы дают возможность быстро скорректировать оценочные предположения и увидеть, к каким последствиям это приведет. Необходимые вычисления на компьютере выполняются мгновенно, тогда как их выполнение вручную занимает слишком много времени и подвержено ошибкам.

Арбитраж нереалистичных ожиданий. Допустим, ваш начальник настаивает, что проект должен быть завершен за 50 человеко-месяцев и 11 календарных месяцев; вы создали оценку, показанную на рис. 14.3. Светлый прямоугольник в левом нижнем углу показывает ограничения, установленные начальником. Точечная диаграмма из 1000 смоделированных результатов показывает, что только 8 из 1000 результатов проекта попали в заданную область. Диаграмма становится весьма наглядным аргументом против попыток завершить проект в рамках этих ограничений!

Объективное авторитетное суждение при пересмотре предположений. Стандартная, хотя и нездоровая ситуация в области оценки программного проекта: одна из ключевых фигур проекта отвергает исходную оценку, потому что она кажется слишком большой. Из проекта предлагается исключить несколько второстепенных функций, что якобы должно привести к непропорциональному сокращению стоимости и сроков. Другая вариация на ту же тему — небольшое увеличение численности группы и надежды на существенное сокращение срока.

Оценочная программа может выступить в роли беспристрастного арбитра в оценке эффекта таких изменений. Без оценочной программы вам придется объяснять начальству, что урезание функциональности не обеспечит нужного изменения в стоимости и сроках. С программой вы просто встаете на одну сторону с начальником и сваливаете все неприятное на программу: это она указывает на то, что предложенные изменения не дадут желаемого эффекта.

На рис. 14.4 показан пример вычисленного соотношения между объемом работ и сроками проекта, то есть экономия в объеме работ при возможности продления сроков. Возможно, вам будет проще убедить начальника в своей правоте, если программа скажет, что сокращение срока на один месяц достигается увеличением объема работ с 20 до 26 человеко-месяцев.

Логическая проверка оценок, полученных неформальными методами. Лучшие оценщики применяют несколько методов оценки, а затем анализируют

тенденции к схождению или расхождению между полученными результатами. Оценка, созданная с помощью коммерческой программы, может стать одним из таких результатов.

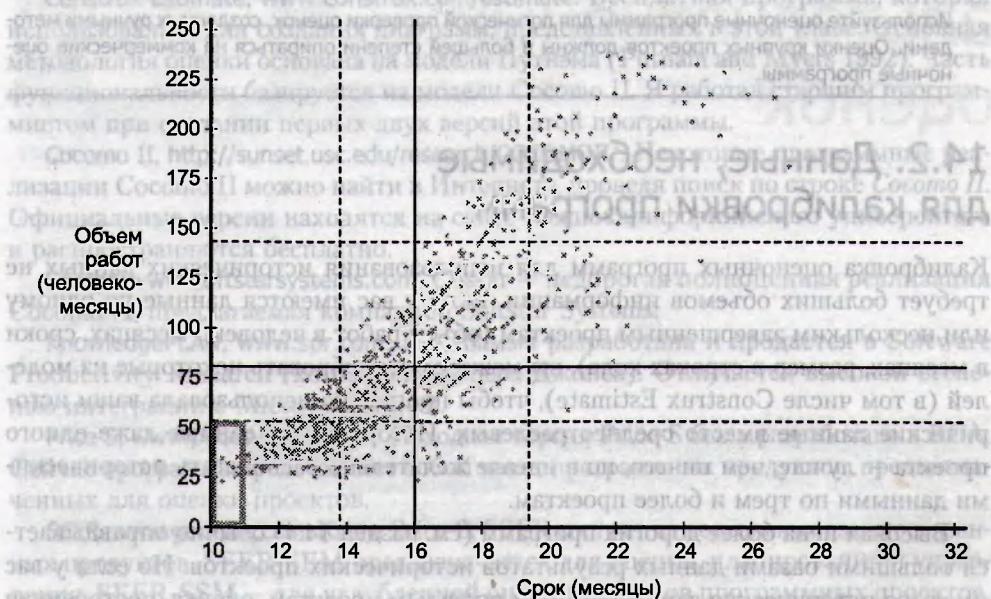


Рис. 14.3. В этой модели только 8 из 1000 результатов удовлетворяют желаемым ограничениям на стоимость и сроки

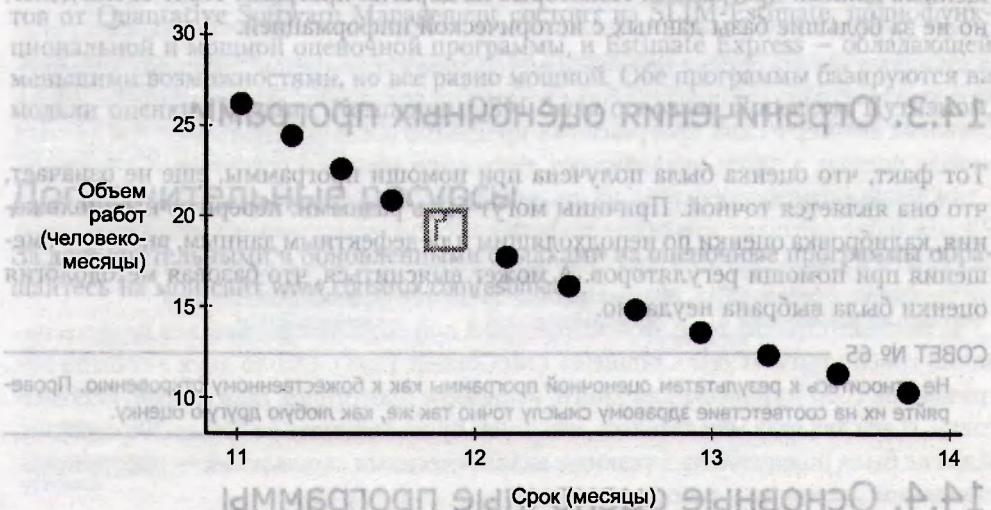


Рис. 14.4. Вычисленный эффект от сокращения или увеличения срока

Оценка крупных проектов. Чем крупнее оцениваемый проект, тем в меньшей степени он может оцениваться исключительно неформальными методами.

В крупном проекте оценочная программа должна предоставить по крайней мере одну из задействованных оценок.

СОВЕТ № 64

Используйте оценочные программы для логической проверки оценок, созданных ручными методами. Оценки крупных проектов должны в большей степени опираться на коммерческие оценочные программы.

14.2. Данные, необходимые для калибровки программ

Калибровка оценочных программ для использования исторических данных не требует больших объемов информации. Если у вас имеются данные по одному или нескольким завершенным проектам (объем работ в человеко-месяцах, сроки в месяцах, размер в строках кода), вы можете откалибровать некоторые из моделей (в том числе Construx Estimate), чтобы программа использовала ваши исторические данные вместо среднеотраслевых. Исторические данные даже одного проекта — лучше, чем ничего, но в идеале желательно располагать историческими данными по трем и более проектам.

Высокая цена более дорогих программ (см. раздел 14.4) обычно оправдываеться большими базами данных результатов исторических проектов. Но если у вас имеются исторические данные по трем прошлым проектам, оценка, построенная по собственным данным, обычно будет точнее оценки, созданной на основе обобщенных данных программы. Некоторые из дорогих программ стоят своих денег, но не за большие базы данных с исторической информацией.

14.3. Ограничения оценочных программ

Тот факт, что оценка была получена при помощи программы, еще не означает, что она является точной. Причины могут быть разными: неверные предположения, калибровка оценки по неподходящим или дефектным данным, внесение смещения при помощи регуляторов. А может выясниться, что базовая методология оценки была выбрана неудачно.

СОВЕТ № 65

Не относитесь к результатам оценочной программы как к божественному откровению. Проверяйте их на соответствие здравому смыслу точно так же, как любую другую оценку.

14.4. Основные оценочные программы

Существует множество оценочных программ. Разброс цен очень широк, от бесплатного распространения до \$20 000 за годовую лицензию на одно рабочее место. Далее представлены некоторые из наиболее популярных программ.

Angel, <http://dec.bournemouth.ac.uk/ESERG/ANGEL/>. Analogy Software Tool — интересная программа с возможностью оценки будущих проектов по аналогии с прошлыми проектами.

Construx Estimate, www.construx.com/estimate. Бесплатная программа, которая использовалась для создания диаграмм, представленных в этой главе. Основная методология оценки основана на модели Путнэма (Putnam and Myers 1992). Часть функциональности базируется на модели Cocomo II. Я работал старшим программистом при создании первых двух версий этой программы.

Cocomo II, <http://sunset.usc.edu/research/COCOMOII>. Некоторые программные реализации Cocomo II можно найти в Интернете, проведя поиск по строке *Cocomo II*. Официальные версии находятся на сайте Южнокалифорнийского университета и распространяются бесплатно.

Costar, www.softstarsystems.com. Costar — недорогая полноценная реализация Cocomo II, предлагаемая компанией Softstar Systems.

KnowledgePLAN, www.spr.com. Программа разработана и продается в Software Productivity Research (компания Кейпера Джонса). Отличается высокой степенью интеграции с Microsoft Project.

Price-S, www.pricesystems.com. Исходная версия Price-S была разработана в RCA. Сейчас программа представляет собой пакет программных продуктов, предназначенных для оценки проектов.

SEER, www.galarath.com. Как и Price-S, SEER состоит из нескольких взаимосвязанных продуктов. SEER-SEM предназначается для оценки, планирования и управления; SEER-SSM — для углубленной оценки размеров программных проектов, а SEER-AccuScope — для простой оценки размеров.

SLIM-Estimate и Estimate Express, www.qsm.com. Семейство программных продуктов от Quantitative Software Management состоит из SLIM-Estimate, полнофункциональной и мощной оценочной программы, и Estimate Express — обладающей меньшими возможностями, но все равно мощной. Обе программы базируются на модели оценки Путнэма. Компания QSM была основана Лоренсом Путнэмом.

Дополнительные ресурсы

За дополнительными и обновленными ссылками на оценочные программы обращайтесь на мой сайт www.construx.com/estimate/.

Использование альтернативных оценок 15

Область применения методов этой главы

Альтернативные оценки	
Что оценивается	Размер, объем работ, сроки, функциональность
Размер проекта	М С Б
Стадия разработки	Ранняя–поздняя
Итеративный или последовательный стиль	Оба
Возможная точность	Высокая

Идеального метода оценки не существует, поэтому применение нескольких альтернативных методов приносит пользу во многих ситуациях. Производители самых сложных коммерческих программных продуктов обычно используют три разных метода оценки, а затем анализируют тенденции между полученными оценками. Схождение оценок указывает на то, что оценки, вероятно, достаточно хороши. Расхождение означает, что какие-то факторы были упущены и в проблеме следует разобраться получше. Методика альтернативных оценок в равной степени относится к оценкам размера, объема работ, сроков и функциональности.

Я впервые столкнулся с этой концепцией при создании оценки для первого издания своей книги «Code Complete» (McConnell 1993). Около двух лет было потрачено на сбор материала для книги, написание прототипов глав и прочую подготовку. В эти два года мне казалось, что книга будет занимать от 250 до 300 страниц. Идея не была подкреплена никакими аналитическими выкладками — просто представления, засевшие у меня в голове.

До этого я ни разу не публиковал книги и поэтому решил, что предложение издастельству должно убедительно показать, что книга действительно будет закончена. Когда подготовка предложения подходила к концу, я создал свою первую оценку методом декомпозиции. Я перебрал все пункты примерной структуры

книги в плане и оценил каждую секцию по отдельности. В табл. 15.1 показан примерный вид полученных данных.

Таблица 15.1. Оценки количества страниц в черновом варианте «Code Complete», полученные субъективным суждением и методом декомпозиции

Глава	Оценка № 1. Исходная оценка «по интуиции»	Оценка № 2. Экспертная оценка с декомпозицией
Предисловие	—	4
Введение	—	5
Модели	—	11
Предварительные условия	—	52
Послесловие	—	20
Обзор тем	—	20
ИТОГО	250–300	802

Фактически до этого момента я использовал два метода оценки: собственную интуицию, подсказывавшую диапазон от 250 до 300 страниц, и экспертную оценку с декомпозицией, которая вела к оценке в 802 страницы. Между двумя оценками существовало значительное расхождение, и я должен был понять, почему они так сильно отличаются.

Я был привязан к своим 250-страничным представлениям о длине книги и подумал: «Оценка в 802 страницы не может быть правильной. Наверняка я где-то ошибся». Я решил оценить книгу еще раз и получить «правильный» ответ.

Для третьей оценки было взято количество страниц в каждой из прототипов глав. Я разделил количество страниц на количество пунктов в примерной структуре книги. Получилось, что один пункт примерной структуры соответствует 1,64 страницы. Далее я перебрал все пункты примерной структуры всей книги, просуммировал пункты и умножил сумму на 1,64. В табл. 15.2 показана оценка, полученная этим способом.

Таблица 15.2. Оценка количества страниц в черновом варианте «Code Complete» с использованием примерной структуры и исторических данных

Глава	Оценка № 1. Исходная оценка «по интуиции»	Оценка № 2. Экспертная оценка с декомпозицией	Оценка № 3. Примерная структура и исторические данные
Предисловие	—	4	4
Введение	—	5	5
Модели	—	11	11
Предварительные условия	—	52	52
Послесловие	—	20	16
Обзор тем	—	20	21
ИТОГО	250–300	802	759

Третья оценка в 759 страниц менее чем на 5 % отличалась от второй оценки (802 страницы). Из-за близости оценок я четко уяснил, что моя книга займет не 250–300 страниц, как думал в течение двух лет, а от 750 до 800 страниц.

Мой опыт был типичным проявлением более общего принципа: объединение результатов, полученных разными оценщиками (или с применением разных методов), повышает точность оценки (Jørgensen 2002, Tockey 2004).

СОВЕТ № 66

Используйте несколько альтернативных методов оценки и проанализируйте совпадения или расхождения результатов.

Между моей оценкой книги и программными проектами можно провести прямую аналогию. Наши представления о возможных затратах, сроках и функциональности проектов не строятся ни на какой конкретной основе. Люди склонны придерживаться своего предвзятого мнения, пока кто-нибудь не представит им достаточно данных, чтобы поколебать это предубеждение. Без дополнительных данных я бы не поверил, что мой проект окажется в три раза больше, чем я представлял. Но даже после получения некоторой информации мне все равно потребовались новые данные, которые убедили меня окончательно.

Есть и другая параллель с программным обеспечением: плохие новости лучше узнавать раньше, чем позже. Я изменил свои ожидания относительно размера проекта на стадии составления предложения. Вообще говоря, на этом можно было остановиться, однако я проанализировал объем проекта и решил, что за него все равно стоит взяться, — оценка дала мне более реалистичное представление о планируемых сроках и обязательствах, которые я на себя принимал.

Тот факт, что два совершенно разных метода дали сходные оценки, укрепил мою уверенность в точности оценок. Обязательно применяйте в программных проектах различные виды оценочных методов для создания разных оценок. Например, воспользуйтесь опосредованным методом, экспертной оценкой, оценкой по аналогии и какой-нибудь оценочной программой.

Как только я принял сходжение моих оценок, внезапно стало очевидно, что этот размер подтверждается и другими данными. Одна из моих глав-прототипов занимала 72 страницы. В 250-страничной книге это составило бы 29 % общего объема. Тем не менее после завершения этой главы у меня и в мыслях не было, что проект готов на 29 %. На интуитивном уровне я знал, что глава составляет не более 10 % книги. Когда две сходящиеся оценки открыли мне глаза, я понял, что длина прототипа подтверждает общую длину книги в диапазоне 750–800 страниц.

Наиболее точная оценка обычно формируется по данным, специфическим для конкретного проекта. В итоге у меня получилась рукопись из 749 страниц; ее объем всего на 10 страниц (1 %) отличался от оценки, полученной с использованием исторических данных того же проекта.

Исходная интуитивная оценка была основана на чтении других книг из диапазона 250–300 страниц. До этого я книг не писал и поэтому наивно предположил, что если написать 250–300 страниц текста, то получится книга из 250–300 страниц. Однако количество страниц в издаваемой книге увеличивается из-за пустых

Формирование

страниц между главами, пустых страниц между частями книги, содержания, списка иллюстраций, алфавитного указателя и прочих служебных данных. Кроме того, объем книги зависит от выбора шрифта, размера полей, межстрочных интервалов и т. д. Все эти обстоятельства вполне очевидны, если вы о них знаете, однако неопытному автору вроде меня легко забыть о них в своей оценке. Даже в своей декомпозиционной оценке я допустил классическую ошибку: я тщательно оценил стороны проекта, хорошо мне известные, но забыл учесть некоторые важные составляющие проекта.

Наконец, две оценки сошлись в диапазоне 5 %. Тогда я еще этого не знал, но такое значение является хорошим показателем сходимости вообще (в широкой части конуса неопределенности обычно приходится довольствоваться большим расхождением).

СОВЕТ № 67

Если разные методы оценки приводят к разным результатам, попытайтесь выявить факторы, из-за которых возникают различия. Продолжайте оценивать проект заново до тех пор, пока результаты, полученные разными методами, не будут расходиться в пределах 5 %.

Позднее меня попросили провести оценку программного проекта для одного из моих клиентов. На рис. 15.1 крестиками отмечены отдельные оценки, созданные мной для этого проекта. Размер крестика представляет мою уверенность в каждой оценке. Треугольником обозначена «самая точная оценка» в 75 человеко-месяцев и 12 календарных месяцев. Хотя в расположении крестиков существует некоторый разброс, все оценки, в которых я был уверен, отличались от «самой точной оценки» в пределах 5 %. Квадрат изображает деловую цель моего клиента — 25 человеко-месяцев при 5 календарных месяцах.

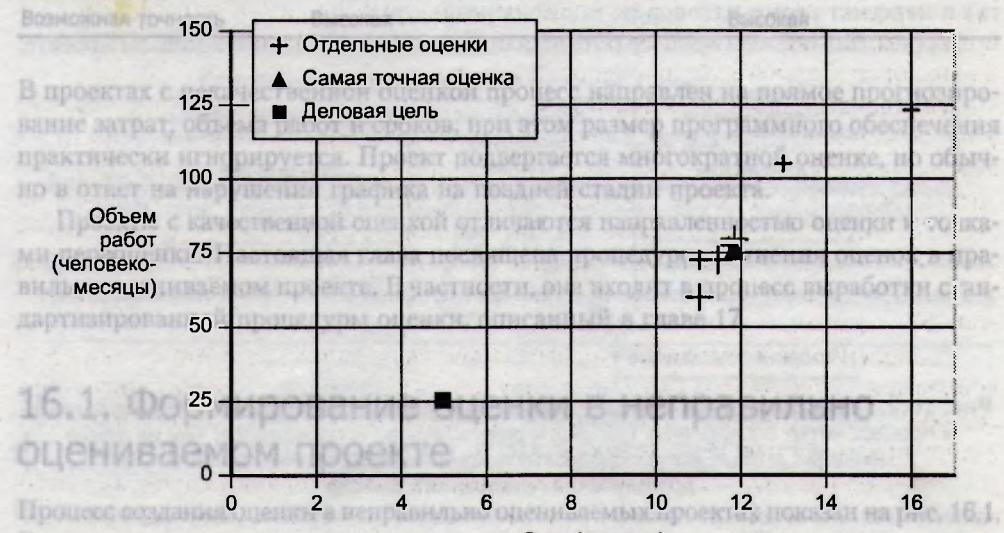


Рис. 15.1. Пример альтернативных оценок в программном проекте

В этом конкретном случае клиент решил действовать, руководствуясь деловой целью в 5 календарных месяцев и 25 человеко-месяцам. Решение оказалось неудачным. В конечном счете проект был выдан через 14 календарных месяцев при 80 человеко-месяцах работы, причем группа реализовала гораздо меньше функциональности, чем планировалось первоначально.

СОВЕТ № 68

Если деловая цель противоречит нескольким сходящимся оценкам, лучше доверьтесь оценкам.

Дополнительные ресурсы

Tockey, Steve. «Return on Software». MA: Addison-Wesley, 2005. В главе 22 книги Токи обсуждаются оценки с применением альтернативных методов. Глава 23 посвящена учету неточности в оценках, а главы 24–25 – особенностям принятия решений в условиях риска и неопределенности (Jørgensen 2002).

Когда я принял окончание моих оценок линейкой, мне было ясно, что этот размер подтверждается и другими данными. Одна из моих глав-прототипов занимала 72 страницы. В 250-страницной книге это составляло бы 29 % общего объема. Тем не менее после завершения этой главы у меня и в мыслях не было, что проект готов на 29 %. На интуитивном уровне я знал, что книга составляет не более 10 страниц. Когда две сходящиеся оценки открыли глаза, я понял, что длина профита подтверждает общую длину книги – примерно 750–800 страниц.

Наиболее точная оценка обычно формируется по данным, специфическим для конкретного проекта. В итоге у меня получилась книжка из 740 страниц; ее объем всего на 10 страниц (1 %) отличается от оценки, полученной с использованием исторических данных того же проекта.

Исходная интуитивная оценка была основана на чтении других книг из диапазона 250–300 страниц. До этого я книг не писал и поэтому наивно предположил, что если написать 250–300 страниц текста, то получится книга из 250–300 страниц. Однако это могло бы привести к ошибке из-за пустых

Формирование оценок в правильно оцениваемых проектах

16

входные данные для ее подготовки и выходные данные для ее реализации. Сама процедура оценки предполагает определение оценки для ее подгото-

вки (так же как и в предыдущем случае, заданного для подготовки под нужный резуль-

тат). Некоторые стандартизированы. Процедуры оценки описаны в главе 17.

Выходные данные, то есть оценка, система расчетов, сроков, стоимости и функциональности, создаются следованием некоторому процессу, используя различные методы оценки.

В этом контексте особенно интересно уточнение различия между оценками, используемыми в различных областях.

Все эти различия обусловлены тем, что если оценка отходит от желаемой, у руководства проекта

также есть возможность отходить от реальной, но это не всегда возможно.

Область применения методов этой главы

	Переход к более точной оценке на более поздней стадии проекта	Уточнение оценки по данным проекта
Что оценивается	Размер, объем работ, сроки, функциональность	Размер, объем работ, сроки, функциональность
Размер проекта	– СБ	МСБ
Стадия разработки	Ранняя–поздняя	Ранняя–поздняя
Итеративный или последовательный стиль	Последовательный	Оба
Возможная точность	Высокая	Высокая

В проектах с некачественной оценкой процесс направлен на прямое прогнозирование затрат, объема работ и сроков; при этом размер программного обеспечения практически игнорируется. Проект подвергается многократной оценке, но обычно в ответ на нарушения графика на поздней стадии проекта.

Проекты с качественной оценкой отличаются направленностью оценки и точками переоценки. Настоящая глава посвящена процедуре уточнения оценок в правильно оцениваемом проекте. В частности, она входит в процесс выработки стандартизированной процедуры оценки, описанный в главе 17.

16.1. Формирование оценки в неправильно оцениваемом проекте

Процесс создания оценки в неправильно оцениваемых проектах показан на рис. 16.1. Входные данные, процесс оценки и выходные данные не имеют четкого определения, хотя и остаются открытыми для оспаривания и изучения. Изучение процесса

оценки сыграло бы положительную роль, если бы оно было направлено на получение более точных результатов. Но как правило, целью изучения является уменьшение оценки. Иначе говоря, изучение давит на оценку сверху, и это давление не уравновешивается соответствующим давлением снизу.

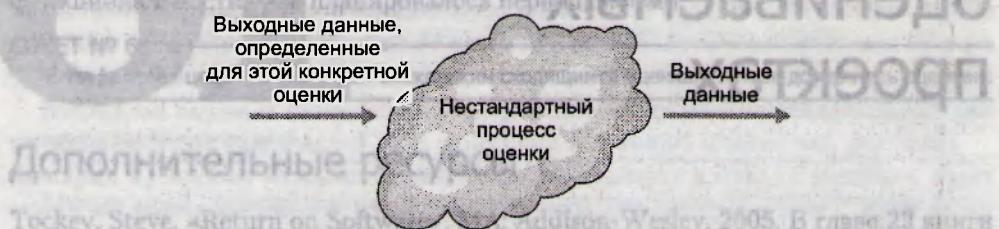


Рис. 16.1. Процесс создания оценки в неправильно оцениваемых проектах. Ни входные данные, ни процесс не имеют четкого определения

СОВЕТ № 69

Не оспаривайте результаты оценки, а принимайте их как данность. Изменение результата может производиться только одним способом: изменением входных данных и повторным вычислением.

16.2. Формирование оценки в правильно оцениваемом проекте

Если входные данные и процесс четко определены, произвольное изменение вывода не рационально. Возможно, лицам, ответственным за принятие решений, результат оценки не понравится, но правильным действием по его корректировке должна быть регулировка входных данных (например, сокращение объема проекта) и пересчет результатов, а не подмена результата.

На рис. 16.2 показан процесс создания оценки в правильно оцениваемом проекте.

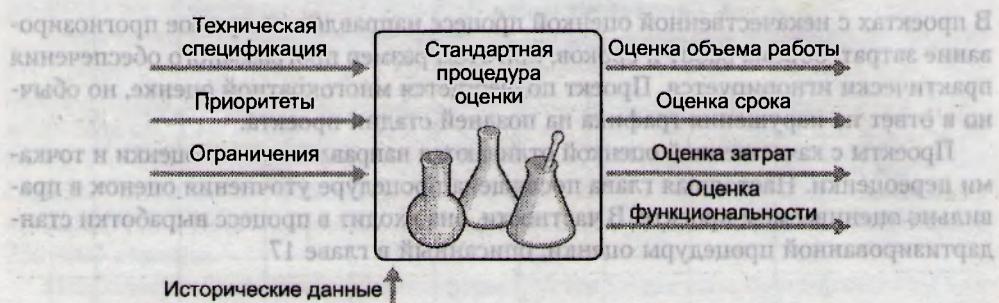


Рис. 16.2. Процесс создания оценки в правильно оцениваемых проектах. Входные данные и процесс четко определены. Процесс и результаты не оспариваются; тем не менее входные данные могут подвергаться итеративным изменениям вплоть до получения приемлемого вывода

В правильно оцениваемом проекте входные данные включают техническую спецификацию, приоритеты и ограничения. Эти входные данные регулируются

до тех пор, пока процесс оценки не даст приемлемый результат. Исторические данные также относятся к входным данным оценки и используются для калибровки предположений о производительности. Они расположены отдельно в нижней части диаграммы, потому что исторические данные не должны регулироваться в контексте конкретной оценки (и особенно для ее подгонки под нужный результат).

Сама процедура оценки *стандартизирована*. Другими словами, эта процедура определяется заранее, еще до того, как возникнет надобность в создании конкретной оценки. Благодаря стандартизации она не регулируется на уровне отдельных оценок (как и в предыдущем случае, особенно для подгонки под нужный результат). Некоторые стандартизованные процедуры оценки описаны в главе 17.

Выходные данные, то есть оценки объема работ, сроков, стоимости и функциональности, создаются следованием четко определенному процессу, использующему четко определенные входные данные. В правильно оцениваемом проекте результаты в принципе не могут оспариваться.

В этом контексте особенно полезны четкие различия между оценками, целями и обязательствами. Даже если оценка отлична от желаемой, у руководства проекта могут найтись достаточно убедительные причины для выбора цели, более агрессивной по сравнению с оценкой. Тем не менее сама оценка от этого не изменится.

Единственным фактором в процедуре оценки, который когда-либо приходится оценивать в традиционном смысле (то есть посредством суждения), является размер. Субъективное суждение применяется для оценки размера только на очень ранней стадии проекта, до появления требований, пользовательских историй, сценариев тестирования или других счетных показателей. Объем работы вычисляется по оценке размера с использованием исторических данных производительности. Сроки, затраты и функциональность вычисляются на основании оценки объема. Общая схема показана на рис. 16.3.



Рис. 16.3. Логика создания единой оценки в правильно оцениваемом проекте. Объем работ, срок, стоимость и функциональность определяются по оценке размера

СОВЕТ № 70

Сначала сконцентрируйтесь на оценке размера. Затем вычислите объем работ, сроки, стоимость и функциональность по оценке размера.

16.3. Хронологическая последовательность формирования оценки для всего проекта

Из-за присутствия конуса неопределенности многие проекты полезно переоценивать по несколько раз. Оценка, созданная на поздней стадии проекта, может оказаться более точной, чем ранние оценки. Повторная оценка проекта

178 Глава 16 • Формирование оценок в правильно оцениваемых проектах

с улучшением точности также помогает уточнить планы и порядок управления проектом.

СОВЕТ № 71

Проводите повторную оценку.

Повторная оценка не сводится к простому повторению процедуры оценки. В ней должны быть задействованы более точные методы, которые становятся доступными по мере продвижения проекта. На рис. 16.4 приведена сводка наиболее полезных методов оценки для разных стадий типичных проектов.

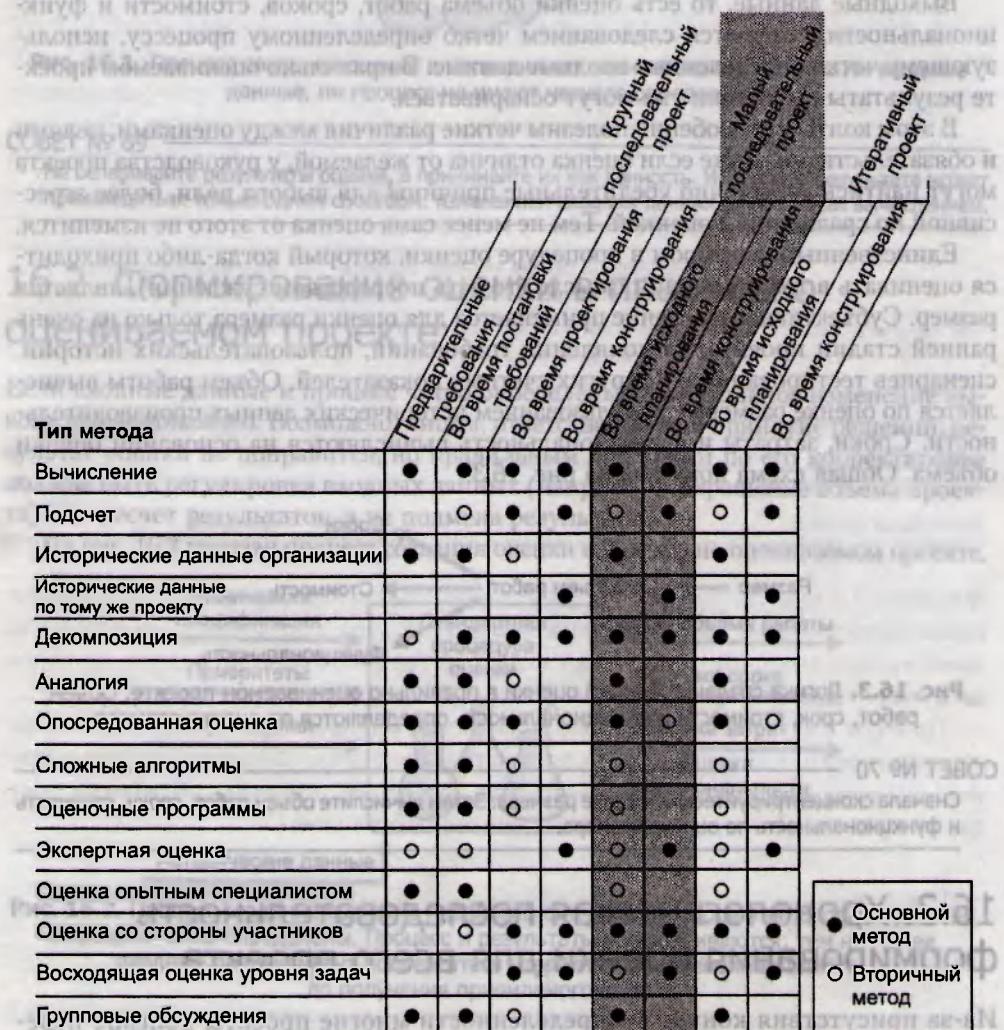


Рис. 16.4. Сводка применимости различных методов оценки в зависимости от типа и фазы проекта

Формирование оценки в крупных проектах

На очень ранней стадии крупного проекта счетные показатели могут оказаться недоступными, и вам придется прибегнуть к алгоритмам, оценочным программам и другим макрометодам. Далее эти ранние оценки улучшаются на групповых обсуждениях и с применением альтернативных методов.

При переходе к более поздним стадиям крупного проекта следует переходить к более точным, основанным на исторических данных счетным методам, и ориентироваться на микрометоды, обеспечивающие более высокую точность (Symons 1991).

СОВЕТ № 72

Переходите от неточных оценок к более точным по мере продвижения работы над проектом.

Формирование оценки в мелких проектах

Мелкие проекты с самого начала оцениваются по тем же принципам, по которым крупные проекты оцениваются в самом конце. Как только вы поближе познакомитесь с людьми, которым предстоит работать над проектом, и начнете раздавать задания, переходите от крупномодульных алгоритмических методов к восходящим методам, базирующимся на индивидуальных оценках заданий работниками. В мелких проектах это может происходить в первый день, а в крупном проекте — через несколько месяцев после начала работы над проектом.

СОВЕТ № 73

Когда проект будет готов к назначению индивидуальных заданий, переходите на восходящую оценку.

16.4. Уточнение оценок

При нарушении контрольных точек в проектах возникает вопрос о перекалибровке графика. Допустим, для завершения проекта был установлен 6-месячный срок. Вы намеревались достичь первой контрольной точки за 4 недели, но в действительности для этого потребовалось 6 недель. Как вы будете действовать дальше?

- Понадеетесь скомпенсировать две потерянные недели позднее?
- Прибавите две недели к общему сроку?
- Умножите весь срок на величину ошибки (в данном случае на 50 %)?

На практике чаще всего встречается вариант № 1. Обычно это обосновывают так: «Требования заняли чуть больше времени, чем предполагалось, но теперь они стабильны, и это позволит нам сэкономить время позднее. Наверстаем при программировании и тестировании». Тем не менее анализ более 300 проектов в 1991 году показал, что проекты практически никогда не наверстывают упущенное время — чаще отставание лишь увеличивается (van Genuchten 1991). Вариант № 1 почти никогда не бывает лучшим.

Вариант № 2 предполагает, что первый этап проекта занял на две недели больше, чем следовало, но все оставшиеся этапы уложатся в изначально отведенные сроки. «Ахиллесова пята» варианта № 2 заключается в том, что ошибки оценки обычно обусловлены системными причинами, распространяющимися на весь проект. Маловероятно, чтобы вся оценка оказалась точной, а нарушение встретилось в той единственной части, которую вы опробовали на практике. За редкими исключениями самой правильной реакцией на результаты, отклоняющиеся от оцениваемых, является вариант № 3.

Конечно, изменение оценки после нарушения контрольной точки — не единственный вариант. Также можно урезать часть функциональности, израсходовать часть буфера риска своего проекта или внести некоторую комбинацию регулировок. Наконец, можно отложить решение и собрать дополнительные данные по следующей контрольной точке. Но если и на следующей контрольной точке проект будет на 50 % отставать от срока, на корректировку останется меньше времени.

СОВЕТ № 74

Если оценка пересчитывается в результате нарушения промежуточных сроков, новая оценка должна базироваться на фактическом ходе проекта, а не на запланированных показателях.

16.5. Представление измененной оценки другим ключевым сторонам проекта

В неправильно управляемых проектах оценщики часто поддаются давлению и выдают точечную оценку на ранней стадии проекта. Далее им приходится нести ответственность за эту оценку на протяжении оставшейся части проекта. Например, предположим, что в ходе проекта был получен набор оценок, перечисленных в табл. 16.1.

Таблица 16.1. История изменения точечной оценки

Фаза проекта	Оценка (человеко-месяцы)
Исходная концепция	10
Согласованное определение продукта	10
Завершение требований	13
Завершение проектирования пользовательского интерфейса	14
Первая промежуточная версия	16
ИТОГО	17

При таком наборе оценок уже при первом возрастании оценки с 10 до 13 месяцев клиент считает, что проект нарушил бюджет и сроки. При каждой последующей переоценке будет казаться, что дела идут все хуже. Но в действительности проблема состояла в том, что первая оценка в 10 месяцев содержала высокую степень неточности и потребовала многократной переоценки. Вполне возможно,

окончательный показатель в 17 человеко-месяцев стал результатом хорошо реализованного проекта, но способ представления оценки создает иное впечатление.

Сравните со сценарием, в котором диапазонные оценки сужаются по мере продвижения проекта, как показано в табл. 16.2.

Таблица 16.2. История изменения диапазонной оценки

Фаза проекта	Оценка (человеко-месяцы)
Исходная концепция	3–40
Согласованное определение продукта	5–20
Завершение требований	9–20
Завершение проектирования пользовательского интерфейса	12–18
Первая промежуточная версия	15–18
ИТОГО	17

При пошаговом уточнении оценки ваш клиент будет считать, что проект остается в рамках предположений. Вместо того чтобы терять доверие клиента при нарушении одного срока за другим, вы улучшаете свою репутацию, последовательно удовлетворяя ожиданиям клиента сужением диапазонов.

Для применения диапазонов вместо точечных оценок существует еще одна причина: исследователи выяснили, что исходные оценки часто становятся «опорой» для будущих оценок, даже если они не были достаточно хорошо обоснованы (Lim and O'Connorg 1996, Jørgensen 2002). Таким образом, одна неудачная точечная оценка может испортить оценки для целого проекта. Использование диапазонных оценок вместо точечных помогает избежать этой проблемы.

СОВЕТ № 75

Представляйте свои оценки в таком виде, чтобы они уточнялись по мере продвижения проекта.

Когда представлять повторные оценки

Не существует единственно правильных моментов для составления повторной оценки. Точность оценки непрерывно улучшается на протяжении жизненного цикла проекта. Проекты обычно переоцениваются в основных контрольных точках, после выпуска новых версий, а также при серьезном изменении основных предположений проекта (например, при смене требований).

Сколько бы раз вы ни планировали повторную оценку, заранее сообщите свои планы другим сторонам, участвующим в проекте. Конус неопределенности освобождает вас от жестких обязательств на ранней стадии проекта, когда вероятность их выполнения невелика. Тем не менее вы обязаны периодически информировать другие стороны по мере того, как ваши представления о проекте становятся более четкими.

В табл. 16.3 приведен пример графика оценки, который можно опубликовать для последовательного проекта.

Таблица 16.3. Примерный график уточнения оценок для последовательного проекта

После контрольной точки	Точность оценки (для оставшейся части проекта)	Комментарии
Исходная концепция	–75 %, +300 %	Только для внутреннего использования; не публикуется за пределами группы разработки
Согласованное определение продукта	–50 %, +100 %	Исследовательская оценка. Используется только внутри компании и не публикуется за ее пределами
Завершение требований и проектирования пользовательского интерфейса (UIDC)	–20 %, +25 %	Оценка бюджета. Допускается публикация верхней границы диапазона. Нижняя граница и средняя точка не публикуются
Первая промежуточная версия	–10 %, +10 %	Предварительная оценка, отрегулированная по данным проекта. Не подлежит внешней публикации; это всего лишь информация к размышлению. Становится доступной приблизительно в UIDC+45 дней
Вторая промежуточная версия	–10 %, +10 %	Предварительная оценка для принятия обязательств. Допускается внешняя публикация верхней границы диапазона. Нижняя граница не публикуется
Третья промежуточная версия	–10 %, +10 %	Окончательная оценка для принятия обязательств. Допускается внешняя публикация средней точки
Промежуточные версии 4–X	–10 %, +10 %	Оценки обновляются только при одобрении новых требований Комитетом по изменениям
Завершение кода	–5 %, +5 %	То же

В зависимости от потребностей сторон, задействованных в проекте, оценки иногда приходится предоставлять чаще или реже, чем показано в таблице. Отрегулируйте детали, чтобы они соответствовали специфике вашей среды. Глава 17 развивает этот пример и предлагает дополнительный метод, хорошо подходящий для итеративных проектов.

СОВЕТ № 76

Сообщайте о своих планах повторного проведения оценки заранее.

Что делать, если начальство запрещает переоценку?

Ваше начальство действительно не разрешает проводить повторную оценку? Сомневаюсь. Оно может запретить изменять *обязательства*, но это совсем не

то же самое. Всегда планируйте периодический пересмотр оценок, хотя бы для собственных целей внутреннего планирования и управления проектами, независимо от того, примет ли начальство или клиент его результаты.

Во многих организациях пересмотр оценок планируется заранее. Эта тема более подробно обсуждается в разделе 17.2.

16.6. Критерии правильной оценки проекта

Во время работы над проектом трудно сказать, насколько хороши ваши оценки. Точность оценок становится очевидной только в ретроспективе. Тем не менее в ретроспективе можно отличить правильно оцениваемый проект от неправильного.

После того как проект будет завершен, проанализируйте историю оценок и определите, удалось ли спрогнозировать конечный итог. На рис. 16.5 показан пример правильно оцениваемого проекта.

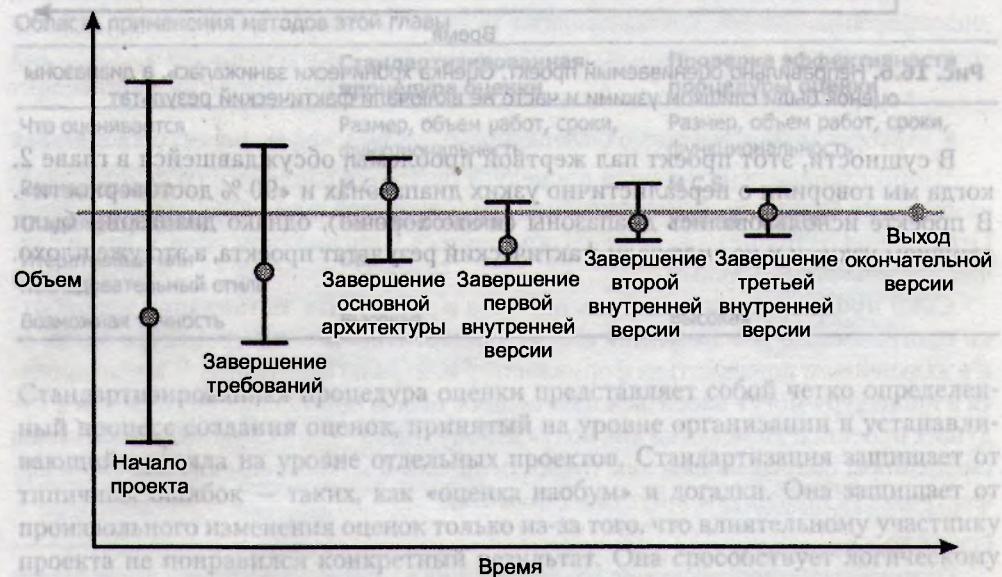


Рис. 16.5. Правильно оцениваемый проект. Точечные оценки не попадают в цель, но все диапазоны включают конечный итог

Вертикальные полосы на рисунке представляют диапазоны оценок. Светлые точки представляют точечные оценки ожидаемых случаев, а однородная синяя точка — фактический результат проекта. В этом проекте точечные оценки до самого конца проекта отличались от фактического результата. Тем не менее каждый из диапазонов, представленных в проекте, включал итоговый результат, поэтому я считаю, что этот проект оценивался правильно.

На рис. 16.6 изображен пример систематически недооцениваемого проекта.

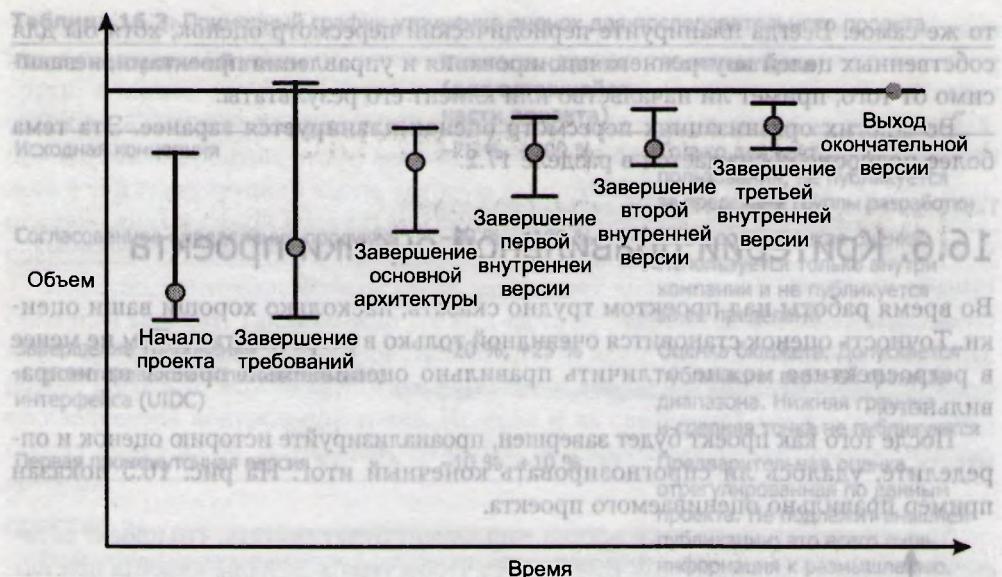


Рис. 16.6. Неправильно оцениваемый проект. Оценка хронически занижалась, а диапазоны оценок были слишком узкими и часто не включали фактический результат

В сущности, этот проект пал жертвой проблемы, обсуждавшейся в главе 2, когда мы говорили о нереалистично узких диапазонах и «90 % достоверности». В проекте использовались диапазоны (и это хорошо), однако диапазоны были слишком узкими и не включали фактический результат проекта, а это уже плохо.

Приоритетные версии 4-Х	-10 %, +10 %	Оценка	Комитет по изменениям
Завершение кода	-5 %, +5 %	То же	опытный выход
Завершение тестов	-10 %, +10 %	Оценка	предоставлена

В зависимости от потребностей сторон, задействованных в проекте, оценки могут приходиться предоставлять чаще или реже, чем показано в таблице. Отрегулируйте график, чтобы оно соответствовало специфике вашей группы. График 17 развивает этот принцип для этого типа групп. Синеконд 336 однажды подоходящий для итеративных проектов.

Что делать если начальство запрещает переоценку?

17

Стандартизованные процедуры оценки

Рис. 17.2 Типичный процесс разработки программного обеспечения в цикле «стадии разработки»

Процесс SDLC определяет операции по разработке программного обеспечения, обычно выполняемые на каждом этапе. Кроме того, он определяет критерии выхода, на основании которых можно оценить выполнение задачи и передать ее вперед. Важно отметить, что каждая стадия имеет свои специфические процедуры. Особенности применения методов этой главы

Стандартизированная процедура оценки	Проверка эффективности процедуры оценки
Что оценивается	Размер, объем работ, сроки, функциональность
Размер проекта	МСБ
Стадия разработки	Ранняя–поздняя
Итеративный или последовательный стиль	Оба
Возможная точность	Высокая

Стандартизированная процедура оценки представляет собой четко определенный процесс создания оценок, принятый на уровне организации и устанавливающий правила на уровне отдельных проектов. Стандартизация защищает от типичных ошибок — таких, как «оценка наобум» и догадки. Она защищает от произвольного изменения оценок только из-за того, что влиятельному участнику проекта не понравился конкретный результат. Она способствует логическому единству процесса оценки. Наконец, в случае особенно неудачной оценки стандартизация позволяет проанализировать свои действия и усовершенствовать процедуру оценки со временем.

Стандартизованные процедуры в равной степени полезны в любых проектах — в крупных и мелких, в итеративных и последовательных, однако специфика изменяется в зависимости от типа проекта.

17.1. Типичные элементы стандартизированной процедуры

В соответствии с описанием типичной последовательности формирования оценки, приведенным в главе 16, стандартизированная процедура оценки:

- по возможности базируется на подсчете и вычислениях, а не на субъективных оценках;
- поощряет применение нескольких альтернативных оценок и сравнение результатов;
- подразумевает план пересмотра оценки в заранее определенных точках проекта;
- определяет изменения метода оценки в ходе жизненного цикла проекта;
- содержит четкое описание неточности оценки;
- определяет, когда оценка может использоваться в качестве основы для формирования бюджета проекта;
- определяет, когда оценка может использоваться в качестве основы для внутренних и внешних обязательств;
- требует архивации оценочных данных и анализа эффективности процедуры.

Чтобы стандартизированная процедура оценки справлялась со своими задачами, очень важно, чтобы организация рассматривала процедуру как *стандарт*. Отклонения от процедуры должны объясняться в письменном виде, и такие случаи должны быть редкими.

Сама процедура должна быть описана в документе «Стандарты разработки программного обеспечения» или «Стандартизированная процедура оценки» и в дальнейшем подвергаться формальному контролю изменений. В конце проекта в процедуру могут вноситься изменения с целью повышения ее точности в будущих проектах. Изменения «на ходу» недопустимы — они слишком подвержены смещениям, подрывающим как точность оценки в конкретном случае, так и эффективность процедуры в будущих проектах.

СОВЕТ № 77

Определите стандартизированную процедуру оценки на уровне организации и применяйте ее на уровне отдельных проектов.

17.2. Адаптация оценки для процесса поэтапного контроля

В многих крупных организациях используется определенный жизненный цикл разработки программного обеспечения (SDLC, Software Development Life Cycle). Такие жизненные циклы обычно являются частью процессов поэтапного контроля — процессов, определяющих жизненный цикл продуктов в виде нескольких

«этапов» и «контрольных точек» (Cooper 2001). К числу компаний, использующих процессы поэтапного контроля, относятся 3M, Agilent, Corning, Exxon, GE, Guinness, Hewlett-Packard, Intel, Kodak, Proctor & Gamble и многие другие.

На рис. 17.1 показан типичный процесс поэтапного контроля.

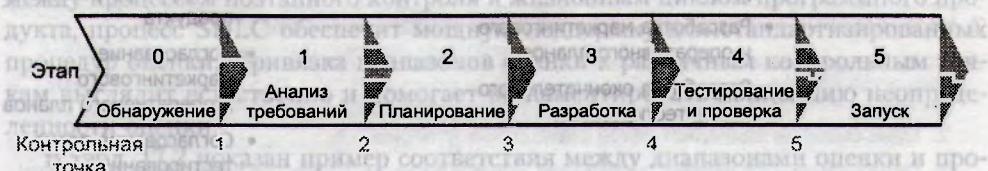


Рис. 17.1. Типичный процесс поэтапного контроля разработки программного продукта

Процесс SDLC определяет операции по разработке программного обеспечения, обычно выполняемые на каждом этапе. Кроме того, он определяет критерии выхода, на основании которых проекту разрешается завершить один этап и перейти к следующему (то есть сместиться к следующей контрольной точке). Специфика процесса SDLC сильно зависит от организации. В табл. 17.1 показано, как SDLC координируется с типичным процессом поэтапного контроля, ориентированным на программный продукт.

Таблица 17.1. Типичный процесс поэтапного контроля, ориентированный на программный продукт (сокращенный вариант)

Этап	Основные операции	Контрольная точка	Основной критерий выхода
0. Обнаружение	<ul style="list-style-type: none"> • Выявление рыночной возможности • Оценка технической приемлемости на высоком уровне • Разработка предварительной экономической модели 	1	<ul style="list-style-type: none"> • Согласование предварительной экономической модели
1. Анализ требований	<ul style="list-style-type: none"> • Формирование концепции продукта • Разработка рыночных требований • Согласование концепции с потребителями 	2	<ul style="list-style-type: none"> • Согласование концепции продукта • Согласование маркетинговых требований
2. Планирование	<ul style="list-style-type: none"> • Разработка подробных требований • Разработка подробных планов • Разработка оценки бюджета • Разработка окончательной экономической модели 	3	<ul style="list-style-type: none"> • Согласование планов разработки программного продукта • Согласование бюджета • Согласование окончательной экономической модели

продолжение ↗

Таблица 17.1 (продолжение)

Этап	Основные операции	Контрольная точка	Основной критерий выхода
3. Разработка	<ul style="list-style-type: none"> • Основной жизненный цикл разработки продукта • Разработка маркетингового и оперативного планов • Разработка окончательного плана тестирования 	4	<ul style="list-style-type: none"> • Согласование плана выпуска программного продукта • Согласование маркетингового и оперативного планов • Согласование плана тестирования программного продукта
4. Тестирование и контроль готовности	<ul style="list-style-type: none"> • Исполнение окончательного плана тестирования • Принятие решения о выпуске 	5	<ul style="list-style-type: none"> • Выполнение критериев выпуска
5. Запуск	<ul style="list-style-type: none"> • Исполнение плана развертывания • Проведение ретроспективного анализа проекта • Сбор мнения клиентов и сообщений о дефектах • Отслеживание экономических результатов 	—	—

С точки зрения оценки, поэтапный контроль создает некоторые проблемы, но и открывает ряд возможностей. Проблемы обусловлены тем фактом, что многие процессы поэтапного контроля изначально разрабатывались для оборудования, потребительских или других продуктов, не имеющих отношения к программам. Хотя основная структура таких процессов остается полезной, ее приходится адаптировать к специфике программного обеспечения, чтобы она работала с такой же эффективностью, как и при разработке других видов продуктов.

Одна из стандартных проблем заключается в том, что разработка часто представляется в виде одного этапа (этап 3 в табл. 17.1). Операции, выполняемые в процессе разработки, занимают от 75 до 90 % общего объема работы программного проекта, и в общем случае я бы предпочел видеть на этом этапе несколько промежуточных контрольных точек вместо одной точки, обозначающей конец этапа. Это одна из ситуаций, в которых следует периодически пересматривать оценки во время этапа. Тем самым обеспечивается эффективное планирование и управление проектом независимо от того, как относится организация к пересмотру оценок на середине этапа.

Вторая стандартная проблема: этапы анализа требований и планирования, определенные в табл. 17.1, часто объединяются в один этап. Фактически это означает, что контрольная точка 3 находится в фазе сужения конуса неопределенности до $\pm 25\%$, что может происходить когда угодно от 15 до 35 % календарного времени проекта (эти процентные показатели более подробно рассматриваются в главе 21). Участникам проекта, не связанным с технической стороной, часто приходится

объяснять, какие операции по разработке программного обеспечения должны быть завершены для получения осмысленного представления о «контрольной точке № 3». Количество и глубина таких операций часто превышают ожидания.

Но когда «не технические» участники проекта получат представление о связи между процессом поэтапного контроля и жизненным циклом программного продукта, процесс SDLC обеспечит мощную поддержку для стандартизованных процедур оценки. Привязка диапазонов оценки к различным контрольным точкам выглядит естественно и помогает регламентировать концепцию неопределенности оценки.

В табл. 17.2 показан пример соответствия между диапазонами оценки и процессами SDLC.

Таблица 17.2. Типичное соответствие между контрольными точками SDLC и оценками

Контрольная точка SDLC	Точность оценки (для оставшейся части проекта)	Использование оценки
1	-75 %, +300 %	Оценка общего представления. Предназначена исключительно для внутреннего использования; не публикуется за пределами группы разработки
2	-50 %, +100 %	Исследовательская оценка. Для внутреннего использования в границах компании; не публикуется за ее пределами
3	-20 %, +25 %	Бюджетная оценка. Допускается внешняя публикация верхней границы диапазона. Нижняя граница и среднее значение не публикуются
4	-10 %, +10 %	Оценка окончательных обязательств. Допускается внешняя публикация среднего значения

СОВЕТ № 78

Координируйте стандартизованную процедуру оценки с процессом SDLC, принятым в вашей организации.

В двух следующих разделах представлены примеры стандартизованных процедур оценки. В разделе 17.3 описан пример процедуры, используемой в последовательных проектах, а в разделе 17.4 — пример процедуры для итеративных проектов.

17.3. Пример стандартизированной процедуры оценки для последовательных проектов

В табл. 17.3 приведен пример стандартизированной процедуры оценки, которая может применяться в последовательных программных проектах. Процедура оценки предполагает, что главным приоритетом организации является функциональность,

а основная цель оценки заключается в повышении точности оценок стоимости и сроков.

Таблица 17.3. Пример стандартизированной процедуры оценки для последовательных проектов — с упором на оценку стоимости и сроков

I. Исследовательская оценка (согласованное определение продукта)

A. Создайте как минимум одну оценку, используя каждый из перечисленных методов.

1. Один оценщик применяет восходящий метод по структуре трудозатрат (WBS).
 2. Один оценщик применяет метод стандартных компонентов.
 3. Один оценщик применяет нисходящий метод и оценку по аналогии с похожими прошлыми проектами.
- B. Оценщики применяют широкополосный Дельфийский метод и вырабатывают точечную номинальную оценку (N).
- C. Оценки всегда должны выдаваться в виде диапазонов $0,5N$ – $2,0N$ (например, -50% , $+100\%$).
1. Точечная номинальная оценка, разработанная для использования в этих вычислениях, не публикуется.
 2. Полученные оценки не должны использоваться для формирования бюджета и внешних обязательств, кроме согласования бюджета с целью завершения стадии проектирования продукта.

II. Бюджетная оценка (завершение проектирования продукта)

A. Создайте новые оценки, используя два метода из этапа I.

1. Восходящий метод по структуре трудозатрат (WBS).
2. Метод стандартных компонентов.

B. Создайте оценку методом функциональных пунктов.

1. Вычислите функциональные пункты по спецификации требований.
2. Откалибруйте оценочную программу по историческим данным своей организации.
3. Оцените номинальный объем работ и сроки с использованием коммерческого оценочного программного пакета.

B. Проводите итеративный пересмотр оценок II.A(1), II.A(2) и II.Б до тех пор, пока не будет достигнуто схождение оценок в пределах 5 %. Используйте среднюю величину оценки как номинал N .

Г. Вычислите диапазонную оценку $0,8N$ – $1,25N$.

1. Бюджет определяется на основе показателя $1.0N$.
2. Резерв на непредвиденные расходы определяется на основе показателя $0,25N$.
3. Возможно выделение дополнительного резерва на основании исторических данных роста требований в данной организации.
4. Публикуется только верхняя граница диапазона ($1,25N$).
5. Оценка не должна использоваться во внешних обязательствах.

III. Оценка предварительных обязательств (после второй внутренней версии)

A. Постройте восходящую оценку.

1. Создайте подробный список задач.
- (а) Список задач должен быть согласован с руководителями разработки, тестирования и подготовки документации.

2. Поручите каждому разработчику, специалисту по тестированию и другим участникам проекта оценить объем работ, необходимых для реализации требований, за которые он отвечает.

- (а) Модули должны оцениваться по лучшему, худшему и ожидаемому случаю.
- (б) Номинальные оценки модулей вычисляются по формуле [ЛучшийСлучай + (4 × ОжидаемыйСлучай) + ХудшийСлучай]/6.

3. Просуммируйте номинальные оценки отдельных модулей.

Б. Сравните оценку II.Г с III.A(3). Вычислите номинальную оценку N по следующей формуле: $(2 \times \text{БольшаяОценка} + \text{МеньшаяОценка})/3$.

В. Вычислите диапазонную оценку $1,0N-1,1N$.

1. Верхняя граница диапазона ($1,1N$) может использоваться во внешних публикациях.

2. Внешние обязательства принимаются по оценке $1,1N$.

3. Диапазон $1,0N-1,1N$ может использоваться во внутренних публикациях.

IV. Оценка окончательных обязательств (после третьей внутренней версии)

А. Сравните оценку с фактическими результатами, полученными на этапе III.

1. Вычислите заново пересмотренное номинальное значение по формуле: $\text{ОставшийсяОбъем-Работ} = \text{ЗапланированныйОстаток}/(\text{ФактическийПредшествующийОбъем}/\text{ЗапланированныйПредшествующийОбъем})$

2. Добавьте все задачи, не учтенные на шаге III.

Б. Сумма IV.A.1 и IV.A.2 может использоваться в качестве нового номинального объема работ, N.

1. Номинал ($1,0N$) может использоваться во внешних публикациях.

2. Внешние обязательства могут делаться по оценке $1,0N$.

3. Диапазон $0,9N-1,1N$ может использоваться во внутренних публикациях.

V. Оценка проекта может пересматриваться в любой момент в ответ на значительные изменения в проектных предпосылках

А. Изменения в предпосылках включают в себя (хотя и не ограничиваются) расширение требований, изменения в определениях основных требований, изменения в доступности персонала и изменения в целевых сроках.

VI. Завершение проекта

А. Соберите и заархивируйте данные по фактическим результатам проекта для использования в будущем.

Б. Проанализируйте точность каждой оценки.

1. Проанализируйте глубинные причины всех основных ошибок.

2. Решите, можно ли было достичь той же точности с меньшими усилиями.

3. Предложите изменения в стандартизированной процедуре оценки.

Процедура оценки, показанная в табл. 17.3, содержит все элементы, обычно присутствующие в подобных процедурах. Она:

- **по возможности базируется на подсчете и вычислениях, а не на субъективных оценках.** Вычисление исследовательской оценки (оценка I в табл. 17.3) требует применения декомпозиции со структурой трудозатрат, оценки по аналогии и метода стандартных компонентов. Ни один из этих методов не обладает выдающейся точностью, но каждый из них по крайней мере

- не ограничивается субъективным суждением и сопряжен с определенными вычислениями;
- поощряет применение нескольких альтернативных оценок и сравнение результатов. В первых трех оценках (I, II и III) задействован метод сравнения альтернативных оценок. Альтернативные методы обычно используются на ранней стадии проекта, когда возможности применения вычислительных методов еще ограничены;
 - подразумевает план пересмотра оценки в заранее определенных точках проекта. В плане задействованы оценки с I по V, что указывает на намерение периодического пересмотра оценок. Каждая оценка связывается с определенными контрольными точками в проекте;
 - определяет изменения метода оценки в ходе жизненного цикла проекта. Шаги отличаются друг от друга и базируются на совершенствующихся данных, которые были получены в ходе проекта и становятся доступными на более поздних этапах. Ближе к концу проекта его исторические данные становятся основным материалом для формирования оценок;
 - содержит четкое описание неточности оценки. В каждом этапе процедуры заложено выражение неточности оценки. Скажем, оценка I.B требует диапазона $-50\%, +100$, а в оценке IV.B неточность сужается до $\pm 10\%$;
 - определяет, когда оценка может использоваться в качестве основы для формирования бюджета проекта. Оценка II названа «бюджетной оценкой». В описании оценки I явно указано, что она не может использоваться в качестве основы для формирования бюджета;
 - определяет, когда оценка может использоваться в качестве основы для внутренних и внешних обязательств. Оценка III называется «оценкой предварительных обязательств», а оценка IV — «оценкой окончательных обязательств». В описаниях более ранних оценок явно указано, что они не могут использоваться в качестве основы для принятия внешних обязательств.

17.4. Пример стандартизированной процедуры оценки для итеративных проектов

В табл. 17.3 приведен пример стандартизированной процедуры оценки, адаптированной для итеративных программных проектов. Такие процедуры обычно работают с наибольшей эффективностью в организациях с годичным бюджетным циклом. Бюджет фиксируется в начале цикла; это означает, что укомплектованность штата тоже фиксируется. Следовательно, подобные оценки не должны быть направлены ни на оценку стоимости (которая фиксируется бюджетом), ни на оценку сроков (которые для годичных бюджетных циклов составляют один год). Задача оценщика — оценить объем функциональности, которая может быть реализована с фиксированным штатом за фиксированный промежуток времени.

Процедура, описанная в табл. 17.4, предполагает, что итерации находятся под правильным управлением — каждая итерация доводится до уровня качества, обеспечивающего возможность выпуска версии, при выходе каждой версии выполняется необходимая «зачистка», отставания от графика не накапливаются и т. д.

Таблица 17.4. Пример стандартизированной процедуры оценки для итеративных проектов — с упором на оценку функциональности

I. Исследовательская оценка (для планирования первой итерации)

- Запланированная функциональность оценивается с применением абстрактных рейтингов.
- Первая итерация планируется с использованием исторических данных организации.
 - Длина итерации не должна превышать одного месяца.
 - Оценка для всего проекта не создается.
 - Никакие обязательства не принимаются.

II. Плановая оценка (планирование второй и третьей итераций)

- Вычислите среднее количество абстрактных пунктов на человека-неделю (для калибровки объема работ).
- Вычислите среднее количество абстрактных пунктов на календарную неделю (для калибровки сроков).
- Данные II.А и II.В используются для планирования второй и третьей итераций.
 - Длина итерации не должна превышать одного месяца.
 - Номинальная оценка всего проекта (N) составляется в форме абстрактных пунктов, которые могут быть реализованы при заданном периоде времени и численности штата.
 - Оценка может использоваться во внутренних публикациях в виде диапазона $0,75N-1,0N$.
 - Оценка не должна использоваться во внешних публикациях.
 - Никакие обязательства не принимаются.

III. Оценка для формирования обязательств (после третьей итерации)

- Вычислите среднее количество абстрактных пунктов на человека-неделю по данным первых трех итераций (для калибровки объема работ).
- Вычислите среднее количество абстрактных пунктов на календарную неделю по данным первых трех итераций (для калибровки сроков).
- Данные III.А и III.Б используются для планирования оставшейся части проекта.
 - Номинальная оценка всего проекта (N) составляется в форме абстрактных пунктов, которые могут быть реализованы при заданном периоде времени и численности штата.
 - Оценка может использоваться во внутренних публикациях в виде диапазона $0,9N-1,1N$.
 - Во внешних публикациях оценка может использоваться в виде диапазона $0,9N-1,0N$.
 - Обязательства принимаются на основе диапазона $0,9N-1,0N$.

IV. Оценка проекта может пересматриваться в любой момент в ответ на значительные изменения в проектных предпосылках

- Как минимум калибровка оценок проекта должна обновляться после каждой третьей итерации с учетом изменений в комплектации штата, повышения производительности и других факторов.
- V. Завершение проекта**
- Соберите и заархивируйте данные по фактическим результатам проекта для использования в будущем.

продолжение ↗

5. Проанализируйте точность каждой оценки.

1. Проанализируйте глубинные причины всех основных ошибок.
2. Решите, можно ли было достичь той же точности с меньшими усилиями.
3. Предложите изменения в стандартизированной процедуре оценки.

Процедура оценки, показанная в табл. 17.4, содержит многие элементы, обычно присутствующие в подобных процедурах. Она:

- по возможности базируется на подсчете и вычислениях, а не на субъективных оценках. В оценке II задействованы фактические данные, и дальнейшие оценки вычисляются на основании исторических данных проекта;
- поощряет применение нескольких альтернативных оценок и сравнение результатов. Данная процедура не требует альтернативных оценок. Впрочем, если вы решили использовать эту процедуру, но считаете, что исторические пункты не обеспечивают достаточной точности прогноза, измените процедуру и включите в нее дополнительные методы оценки;
- подразумевает план пересмотра оценки в заранее определенных точках проекта. В планировании задействованы оценки I–III, что указывает на намерение производить периодический пересмотр оценок;
- определяет изменения метода оценки в ходе жизненного цикла проекта. Как и в случае с последовательной процедурой, все этапы процедуры отличаются друг от друга на основании исторических данных, генерированных проектом;
- содержит четкое описание неточности оценки. В оценке I просто сказано, что оценка всего проекта не производится, а оценка II обеспечивает диапазон неопределенности от 75 до 100 % предполагаемой функциональности;
- определяет, когда оценка может использоваться в качестве основы для формирования бюджета проекта. В данном случае финансовый бюджет фиксирован;
- определяет, когда оценка может использоваться в качестве основы для внутренних и внешних обязательств. Оценка III называется «оценкой для формирования обязательств». В описаниях более ранних оценок явно сказано, что они не могут использоваться для принятия внешних обязательств.

17.5. Пример стандартизированной процедуры оценки от прогрессивной организации

В табл. 17.5 описана процедура оценки, используемая лабораторией разработки программного обеспечения NASA (NASA SEL), одной из самых передовых организаций по разработке программного обеспечения.

Таблица 17.5. Процедура оценки NASA SEL

Входные данные	Выходные данные					
	Фаза проекта	Входные данные для оценки	Оценка размера	Оценка объема работ	Оценка срока	Диапазон неопределенности ¹
Завершение анализа требований	Количество подсистем	11 000 строк кода ²	3000 часов на подсистему	Количество подсистем умножается на 83 недели и делится на численность персонала	+75 %	-43 %
Завершение предварительного проектирования	Количество модулей (функций и/или подпрограмм)	190 строк кода на модуль	52 часа на модуль	Количество модулей умножается на 1,45 недели и делится на численность персонала	+40 %	-29 %
Завершение подробного проектирования	Количество новых и существенно измененных модулей (N). Количество переработанных и незначительно измененных модулей (R)	Строки кода = 200 × (N + 0,2R)	0,31 часа на строку кода	Количество строк кода умножается на 0,0087 недели и делится на численность персонала	+25 %	-20 %
Завершение реализации	Текущий размер в строках кода. Объем работы, выполненной до настоящего момента. Время, затраченное до настоящего момента	Текущий размер увеличивается на 26 % (с учетом тестирования)	Объем выполненной работы увеличивается на 43 % (с учетом работы по завершению)	Затраченное время увеличивается на 54 %	+10 %	-9 %
Завершение тестирования системы	Общий объем выполненной работы	Окончательный размер программного проекта становится известным	Объем выполненной работы увеличивается на 13 % (с учетом работы по завершению)	Затраченное время увеличивается на 18 %	+5 %	-5 %

Источник: Адаптировано по материалам «Manager's Handbook for Software Development», Revision 1 (NASA SEL 1990).

¹ Чтобы сохранить допуск на текущесть кадров, рост требований и т. д., консервативные принципы управления рекомендуют использовать оценки, лежащие между прогнозируемым значением и верхней границей.

² К «строками кода» относятся все конструкции исходного кода, включая комментарии и пустые строки.

Самая замечательная особенность процедуры оценки NASA SEL заключается в том, что она требует меньшей работы для создания более точных оценок. Это характерно для общего правила: чем более изощренными становятся ваши оценки, тем меньше усилий требуется для создания точных оценок.

Конкретные числа в этой процедуре относятся к специфике NASA SEL. Они были откалиброваны за несколько десятилетий сбора и анализа данных и характерны для высокоразвитых организаций, занимающихся разработкой программного обеспечения. Для других организаций они не подходят.

Отличия от процедур, которые могут использоваться менее развитыми организациями, весьма поучительны. Впрочем, существует и определенное сходство. Процедура NASA SEL:

- **по возможности базируется на подсчете и вычислениях, а не на субъективных оценках.** Процедура NASA SEL интересна тем, что даже ранние проектные оценки базируются на подсчетах и вычислениях. Объем работ и сроки никогда не оцениваются напрямую;
- **поощряет применение нескольких альтернативных оценок и сравнение результатов.** Данная процедура не требует одновременного применения нескольких альтернативных методов в любой момент времени. Лаборатория NASA SEL достаточно долго собирала и анализировала исторические данные, что позволяет ей составлять точные оценки с малыми усилиями;
- **подразумевает план пересмотра оценки в заранее определенных точках проекта.** В табл. 17.5 отмечены точки проекта, в которых создаются новые оценки;
- **определяет изменения метода оценки в ходе жизненного цикла проекта.** Каждая строка таблицы представляет отдельный метод оценки для конкретной точки жизненного цикла проекта;
- **содержит четкое описание неточности оценки.** В крайнем правом столбце содержатся величины поправок номинальной оценки. Первая сноска содержит хорошую общую рекомендацию: «...консервативные принципы управления рекомендуют использовать оценки, лежащие между прогнозируемым значением и верхней границей»;
- **определяет, когда оценка может использоваться в качестве основы для формирования бюджета проекта.** В данном случае этот элемент не выражен;
- **определяет, когда оценка может использоваться в качестве основы для внутренних и внешних обязательств.** Эти элементы не выражены в процедуре напрямую. Обратите внимание: первая строка таблицы относится к «завершению анализа требований». В терминологии NASA SEL «анализом требований» называется операция, выполняемая после «постановки требований». Таким образом, из таблицы следует, что первая оценка проекта может строиться в относительно широкой части конуса неопределенности.

17.6. Улучшение стандартизированной процедуры

Если процедура оценки создавалась специально для конкретного случая, оценку трудно усовершенствовать, потому что вы никогда не знаете точно, какой из аспектов оценки привел к появлению неточности. При использовании стандартизованных процедур вам точно известно, каким образом строилась каждая оценка; это поможет в будущем повторить удачу и избежать неудач.

После каждого проекта эффективность оценок анализируется по нескольким направлениям.

- Насколько точными были ваши оценки? Входил ли окончательный результат во все диапазоны (см. раздел 16.6)?
- Достаточно ли широкими были ваши диапазоны? Нельзя ли было их сузить, не утрачивая выявленной изменчивости?
- В какую сторону обычно смещались ваши оценки — завышения или занижения? Или смещения не было и оценки были нейтральными?
- Какие источники стали причиной искажения оценки?
- Какими методами были получены самые точные оценки? Действительно ли эти методы обеспечивают самые точные оценки, или они просто случайно обеспечивали лучшую оценку в данном случае?
- Правильно ли было выбрано время для пересмотра оценок? Сколько было пересмотров — слишком много, слишком мало, в самый раз?
- Был ли процесс оценки более сложным, чем требовалось? Нельзя ли упростить его без потери точности?

СОВЕТ № 79

Анализируйте оценки и процедуры получения оценок в своих проектах; это поможет вам повысить точность оценок и свести к минимуму затраты на их создание.

Дополнительные ресурсы

Boehm, Barry W. «Software Engineering Economics». Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981. В главе 21 описана процедура оценки программных проектов, состоящая из семи этапов.

Cooper, Robert G. «Winning at New Products: Accelerating the Process From Idea to Launch». New York, NY: Perseus Book Group, 2001. Купер — отец процессов поэтапного контроля. В книге рассказано, как адаптировать процесс поэтапного контроля для ваших потребностей.

McGarry, John, et al. «Practical Software Management: Objective Information for Decision Makers», Boston, MA: Addison-Wesley, 2002. В разделе 5.21 обсуждаются факторы, включаемые в процедуру оценки.

NASA SEL. «Manager's Handbook for Software Development», Revision 1, Document Number SEL-84-101. Greenbelt, MD: Goddard Space Flight Center, NASA, 1990. В документе приводится более подробное описание методики оценки программных проектов в лаборатории NASA SEL.

Отличия от других методик оценки в том, что здесь не делается попытка определить точные оценки, а лишь оценка вида «затрачено времени и затрачено средств». Для этого используется методика оценки, основанная на оценке затраченного времени и затраченных средств.

Методика оценки затраченного времени и затраченных средств предполагает, что для каждого проекта определены оптимальные нормы затрат времени и затрат средств, необходимые для выполнения проекта.

Процедура NASA SEL:

- Подготовка к оценке:** определяются критерии оценки, которые могут быть выражены в количественных или качественных показателях. Помимо этого, определяется, что даже ранние результаты оценки должны быть использованы для дальнейшего уточнения оценки.
- Оценка:** определяются критерии оценки, которые должны быть учтены при оценке. Оценка проводится на основе имеющихся данных о проекте и его характеристиках. Оценка проводится на основе имеющихся данных о проекте и его характеристиках. Оценка проводится на основе имеющихся данных о проекте и его характеристиках.
- Результаты оценки:** результаты оценки представляют собой таблицу, в которой отражены оценки по различным критериям. Таблица содержит следующую информацию:
 - подразумевает назначение оценки в зависимости от времени и источника информации;
 - определяет назначение метода оценки в ходе жизненного цикла проекта.

Каждая строка таблицы представляет отдельный метод оценки для конкретной точки жизненного цикла проекта.

* Содержит четкое описание неточности оценки. В крайнем правом столбце таблицы неточность оценки выражена в виде коэффициента, который определяет вероятность того, что фактическая оценка будет отличаться от ожидаемой на величину, равную коэффициенту умноженному на величину неточности.

** Содержит хорошую общую рекомендацию: «...консервативные принципы управления рекомендуют использовать оценки, лежащие между диапазоном управляемым значением и верхней границей».

*** Содержит формулы, которые могут использоваться в качестве основы для формирования бюджета проекта. В данном случае формулы выражены в виде коэффициентов химической интенсивности, вычисляемой по формуле: $I = \frac{C}{W} \cdot \ln \left(\frac{C}{W} \right)$, где C — стоимость проекта, W — стоимость единицы времени.

Конкретные проблемы оценки

Глава 18. Специфические проблемы при оценке размера

Глава 19. Специфические проблемы при оценке объема работ

Глава 20. Специфические проблемы при оценке сроков

Глава 21. Оценка параметров планирования

Глава 22. Стили представления оценок

Глава 23. Политика, переговоры и решение проблем

Роль строк кода в оценке размеров

Оценка программных проектов в строках кода имеет как положительные, так и отрицательные стороны. С одной стороны, у них есть ряд преимуществ. Это облегчение отладки и модификации в том смысле, что строки кода являются языковыми единицами, имеющими ясную структуру и логическую связь между собой. С другой стороны, у них есть ряд недостатков. Одним из них является то, что строки кода не всегда являются языковыми единицами, имеющими ясную структуру и логическую связь между собой. Вторым недостатком является то, что строки кода не всегда являются языковыми единицами, имеющими ясную структуру и логическую связь между собой. Третьим недостатком является то, что строки кода не всегда являются языковыми единицами, имеющими ясную структуру и логическую связь между собой.

Строки кода различно изменяются в зависимости от языка программирования и языка отладки. Поэтому для каждого языка программирования и языка отладки существует свой набор правил, определяющих, каким образом строки кода должны изменяться. Эти правила могут отличаться от языка к языку, но общим принципом является то, что строки кода должны изменяться в соответствии с правилами языка, на котором написаны программы. Для этого необходимо использовать специальные инструменты, такие как компиляторы, интерпретаторы и т.д.

Специфические проблемы при оценке размера

18

Область применения методов этой главы

	Функциональные пункты	Голландский метод	Элементы GUI
Что оценивается	Размер, функциональность	Размер, функциональность	Размер, функциональность
Размер проекта	М С Б	М С Б	М С Б
Стадия разработки	Ранняя–средняя	Ранняя	Ранняя
Итеративный или последовательный стиль	Последовательный	Последовательный	Последовательный
Возможная точность	Высокая	Низкая	Низкая

После перехода от прямой оценки объема работ и сроков к их вычислению по историческим данным наибольшие трудности возникают при оценке размера. Итеративные проекты могут использовать оценку размера для определения количества функций, выдаваемых за итерацию, но обычно они в большей степени концентрируются на методах, предназначенных для оценки функциональности. Оценка на поздних стадиях последовательных проектов часто оказывается направленной на восходящие оценки объема работ, созданные людьми, которым предстоит выполнять эту работу. Таким образом, оценка размера оказывается наиболее применимой на ранних и средних этапах последовательных проектов. Целью оценки размера является поддержка долгосрочной прогнозируемости в широкой части конуса неопределенности.

Стандартные метрики оценки размера — строки программного кода и функциональные пункты — обладают разными достоинствами и недостатками. То же можно сказать о специализированных методах, созданных организациями для собственного применения. Как правило, создание оценок по разным метрикам

размера и проверка их сходства/расхождения обеспечивает самые точные результаты.

В этой главе описано, как создать оценку размера. В главе 19 объясняется, как преобразовать оценку размера в оценку объема работ, а в главе 20 — как преобразовать оценку объема работ в оценку срока.

18.1. Проблемы при оценке размера

Существуют различные показатели размера программных проектов, в том числе:

- функции;
- требования;
- сценарии использования;
- функциональные пункты;
- веб-страницы;
- компоненты GUI (окна, диалоговые окна, отчеты и т. д.);
- таблицы баз данных;
- определения интерфейсов;
- классы;
- строки программного кода.

На практике для оценки чаще всего используются строки программного кода (LOC), поэтому я начну именно с них.

Роль строк кода в оценке размеров

Оценка программных проектов в строках кода имеет как положительные, так и отрицательные стороны. С одной стороны, у них есть ряд преимуществ.

- Данные по количеству строк кода в прошлых проектах легко собираются при помощи служебных программ.
- Во многих организациях уже наработан большой объем исторических данных, выраженных в строках кода.
- Объем работы на строку кода остается более или менее постоянным для разных языков программирования или, во всяком случае, достаточно близким для практических целей. (Как объясняется в главе 5, объем работы на строку кода в большей степени зависит от размера проекта и типа программы, нежели от языка программирования. А вот *функциональность* одной строки кода радикально изменяется в зависимости от языка.)
- Измерения в строках кода позволяют выполнять межпроектные сравнения и оценивать будущие проекты по данным прошлых проектов.
- В большинстве коммерческих оценочных программ оценки объема работ и сроков в конечном счете основываются на строках кода.

С другой стороны, строки кода также создают определенные трудности при оценке размера.

- Упрощенные модели вида «количество строк кода на человека-месяц» подвержены ошибкам из-за издержек масштаба и заметных различий в скорости кодирования для разных типов программного обеспечения.
- Строки кода не могут использоваться в качестве основы для оценки задач, порученных отдельным программистам, из-за огромных различий в производительности.
- Если проект требует более сложного кода по сравнению с проектом, использовавшимся для калибровки предположений о производительности, это может нарушить точность оценки.
- Применение метрики LOC при оценке работы по постановке требований, проектированию и других действий, предшествующих созданию кода, выглядит противоестественно.
- Строки кода трудно оценивать напрямую и их обычно приходится оценивать опосредованно.
- Вы должны заранее тщательно определить, что именно следует считать строкой кода (это необходимо для предотвращения проблем, описанных в разделе 8.2).

Некоторые эксперты возражают против использования строк кода в качестве метрики размера из-за проблем, возникающих при попытке анализа производительности в проектах с разными типами, размерами, языками программирования и программистами (Jones 1997). Другие эксперты указывают, что аналогичные проблемы встречаются и при использовании других метрик размеров, включая функциональные пункты (Putnam and Myers 2003).

Общая проблема оценки в строках кода, функциональных пунктах и других простых метриках заключается в том, что измерение чего-либо столь многогранного, как размер программного проекта, в одномерных метриках неизбежно порождает аномалии, по крайней мере в отдельных ситуациях (Gilb 1988, Gilb 2005).

Одномерные метрики не применяются для описания экономических моделей или других сложных сущностей. Они даже не применяются для определения лучшего подающего в бейсбольной команде — мы учитываем среднее количество поданий, пробежек и других факторов, но даже после этого продолжаем спорить о смысле этих чисел. Если даже уровень подающего нельзя оценить по одному простому показателю, почему можно полагать, что простая метрика подойдет для оценки чего-то настолько сложного, как размер программного проекта?

Мое личное отношение относительно оценки проектов по строкам кода напоминает высказывание Уинстона Черчилля по поводу демократии. Метрика LOC — очень плохой способ оценки размера программного проекта, но все остальные способы еще хуже. В большинстве организаций, несмотря на все свои недостатки, метрика LOC остается основным рабочим инструментом для измерения размера прошлых проектов и создания ранних оценок новых проектов.

Метрика LOC – это *lingua franca* оценки программных проектов и обычно хорошая отправная точка (при условии, что вы не забываете о ее ограничениях).

Возможно, ваша среда достаточно сильно отличается от типичной среды программирования, и количество строк кода в ней слабо коррелируется с размером проекта. В этом случае найдите другой показатель, в большей степени связанный с размером проекта, подсчитайте его и заложите в основу своих оценок размера, как обсуждалось в главе 8. Ищите метрику, легко подсчитываемую, коррелированную с объемом работ и достаточно универсальную для использования в нескольких проектах.

СОВЕТ № 80

Используйте строки программного кода для оценки размеров, но помните об общих ограничениях простых метрик, а также специфических опасностях метрики LOC.

18.2. Функциональные пункты

Одной из альтернатив метрики LOC являются функциональные пункты. Это синтетическая метрика размера программы, которая может применяться для оценки размера проекта на его ранних стадиях (Albrecht 1979). Функциональные пункты проще вычислять по спецификациям требований, чем строки кода; кроме того, они формируют основу для вычисления размера в строках кода. Существует много разных методов для вычисления функциональных пунктов. Стандарт подсчета функциональных пунктов поддерживается группой International Function Point Users Group (IFPUG) и размещается на веб-сайте по адресу www.ifpug.org.

Размер программы в функциональных пунктах базируется на количестве и сложности следующих элементов.

Внешние входные элементы — экраны, формы, диалоговые окна или управляющие сигналы, при помощи которых пользователь или внешняя программа добавляет, удаляет и изменяет данные программы. К этой категории относятся все входные элементы, обладающие уникальным форматом или уникальной логикой обработки.

Внешние выходные элементы — экраны, отчеты, диаграммы или управляющие сигналы, генерируемые программой для пользователя или внешних программ. К этой категории относятся все выходные элементы, отличающиеся по формату или логике обработки от других типов вывода.

Внешние запросы — комбинации входных/выходных элементов, в которых входному элементу ставится в соответствие простая выходная форма. Термин происходит из мира баз данных и относится к прямому поиску данных (обычно по уникальному ключу). В современных графических и веб-приложениях граница между запросами и выходными элементами размыта, но в общем случае запросы производят выборку данных непосредственно из базы и ограничиваются минимальным форматированием, а выходные элементы поддерживают обработку, комбинирование и обобщение сложных данных с широкими возможностями форматирования.

Внутренние логические файлы — основные логические группы пользовательских или управляющих данных, находящиеся под полным контролем программы.

204 Глава 18 • Специфические проблемы при оценке размера

Логический файл представляет собой один неструктурированный файл или одну таблицу в реляционной базе данных.

Внешние интерфейсные файлы — файлы, находящиеся под контролем других программ, с которыми взаимодействует измеряемая программа. К этой категории относятся все основные группы логических или управляющих данных, принимаемых или передаваемых программой.

В табл. 8.1 показано, как счетчики входных элементов, выходных элементов и т. д. преобразуются в нескорректированные функциональные пункты. Количество входных элементов низкой сложности умножается на 3, количество выходных элементов низкой сложности — на 4 и т. д. Сумма полученных чисел дает метрику проекта в нескорректированных функциональных пунктах.

Таблица 18.1. Множители для вычисления нескорректированных функциональных пунктов

Характеристика программы	Функциональные пункты		
	Низкая сложность	Средняя сложность	Высокая сложность
Внешние входные элементы	$\times 3$	$\times 4$	$\times 6$
Внешние выходные элементы	$\times 4$	$\times 5$	$\times 7$
Внешние запросы	$\times 3$	$\times 4$	$\times 6$
Внутренние логические файлы	$\times 4 + ?$	$\times 10$	$\times 15$
Внешние интерфейсные файлы	$\times 5$	$\times 7$	$\times 10$

После суммы нескорректированных функциональных пунктов вычисляется коэффициент влияния; он основывается на влиянии, оказываемом на программу 14 факторами. Среди примеров таких факторов можно назвать передачу данных, оперативный ввод данных, сложность обработки и простоту установки. Коэффициент влияния лежит в диапазоне от 0,65 до 1,35. После умножения нескорректированной суммы на коэффициент влияния вы получаете скорректированную величину в функциональных пунктах.

Если вы читали мои предшествующие комментарии о «субъективных регуляторах», вероятно, вы уже догадываетесь, что я думаю о коэффициенте влияния и его 14 регуляторах. Два исследования показали, что нескорректированные функциональные пункты в большей степени коррелируются с итоговым размером, чем скорректированные (Kemerer 1987, Gaffney and Werling 1991). Некоторые эксперты также рекомендуют исключить оценки «низкой сложности» и «высокой сложности» и классифицировать все счетные показатели как «средние» — тем самым исключается еще один источник субъективизма (Jones 1997). Стандарт ISO/IEC 20926:2003 базируется на нескорректированных функциональных пунктах.

В табл. 18.2 показан пример составления итоговой метрики в скорректированных функциональных пунктах. Конкретные значения входных и выходных элементов, запросов, внутренних логических и внешних интерфейсных файлов представлены исключительно для наглядности.

В показанном примере размер проекта оценивается в 284 функциональных пункта. Полученное значение можно напрямую преобразовать в оценку объема работы (см. главу 19) или сначала преобразовать в оценку в строках кода, а от нее перейти к оценке объема работы.

Таблица 18.2. Пример вычисления размера в функциональных пунктах

Характеристика программы	Функциональные пункты	Низкая сложность	Средняя сложность	Высокая сложность
Внешние входные элементы	$6 \times 3 = 18$	$2 \times 4 = 8$	$3 \times 6 = 18$	
Внешние выходные элементы	$7 \times 4 = 28$	$7 \times 5 = 35$	$0 \times 7 = 0$	
Внешние запросы	$0 \times 3 = 0$	$2 \times 4 = 8$	$4 \times 6 = 24$	
Внутренние логические файлы	$0 \times 4 = 0$	$2 \times 10 = 20$	$3 \times 15 = 45$	
Внешние интерфейсные файлы	$2 \times 5 = 10$	$0 \times 7 = 0$	$7 \times 10 = 70$	
Нескорректированная сумма в функциональных пунктах				284
Коэффициент влияния				1,0
Скорректированная сумма в функциональных пунктах				284

Терминология метода функциональных пунктов заметно ориентирована на базы данных, но группа IFPUG неуклонно обновляет правила подсчета функциональных пунктов, и эта методика хорошо работает для всех типов программного обеспечения. Исследования показали, что сертифицированные специалисты по оценке функциональных пунктов обычно выдают показатели, отличающиеся не более чем на 10 %, так что подсчет функциональных пунктов открывает реальную возможность сокращения неопределенности, связанной с объемом проекта, в ко-
нусе неопределенности на ранней стадии проекта (Stutzke 2005).

СОВЕТ № 81

Воспользуйтесь методом функциональных пунктов, чтобы получить относительно точную оценку размера на ранней стадии проекта.

Преобразование функциональных пунктов в строки кода

Если потребуется перейти от функциональных пунктов к строкам кода, в табл. 18.3 перечислены коэффициенты преобразования для некоторых распространенных языков программирования.

Скажем, если ваша программа на 285 функциональных пунктов будет реализована на Java, из таблицы берется диапазон от 40 до 80 LOC на функциональный пункт; умножив его на 284 функциональных пункта, мы получаем оценку размера от 11 360 до 22 720 LOC, с ожидаемым значением $55 \times 284 = 15675$ LOC. Чтобы не создавать ложного впечатления точности, эти числа можно упростить до диапазона 11 000–23 000 LOC с ожидаемым значением 16 000 LOC.

Коэффициенты, представленные в таблице, используют широкие диапазоны — обычно верхняя граница отличается от нижней в 2–3 раза. Как и во многих других оценках, если вам удастся собрать исторические данные о соответствии между функциональными пунктами и строками кода в вашей организации, это повысит точность оценок и, вероятно, сузит диапазоны оценок по сравнению со среднеотраслевыми данными.

Таблица 18.3. Количество команд на функциональный пункт в разных языках программирования

Язык	Команд на функциональный пункт	Минимум (-1 стандартное отклонение)	Номинал (наиболее вероятное значение)	Максимум (+1 стандартное отклонение)
Ada 83	45	80	125	
Ada 95	30	50	70	
C	60	128	170	
C#	40	55	80	
C++	40	55	140	
Cobol	65	107	150	
Fortran 90	45	80	125	
Fortran 95	30	71	100	
Java	40	55	80	
Макроассемблер	130	213	300	
Perl	10	20	30	
Языки второго поколения (Fortran 77, Cobol, Pascal и т. д.)	65	107	160	
Smalltalk	10	20	40	
SQL	7	13	15	
Языки третьего поколения (Fortran 90, Ada 83 и т. д.)	45	80	125	
Microsoft Visual Basic	15	32	41	

Источник: Адаптировано по данным «Estimating Software Costs» (Jones 1998), «Software Cost Estimation with Cocomo II» (Boehm 2000) и «Estimating Intensive Systems» (Stutzke 2005).

Описание вычисления функциональных пунктов, приведенное в этом разделе, дает лишь весьма поверхностное представление об этой сложной методике. Хотя эксперты в области вычисления функциональных пунктов могут выдавать результаты, отличающиеся в пределах 10 %, результаты вычислений неопытных оценщиков различаются от 20 до 25 % (Kemerer and Porter 1992, Stutzke 2005). За дополнительной информацией об этом методе обращайтесь к разделу «Дополнительные ресурсы» в конце этой главы.

18.3. Упрощенные методы вычисления функциональных пунктов

При вычислении функциональных пунктов необходимо строку за строкой перебрать спецификацию требований и буквально подсчитать все входные и выходные элементы, файлы и т. д. На это может потребоваться много времени.

Эксперты в области оценки предложили ряд упрощенных методов вычисления функциональных пунктов. С учетом других источников изменчивости, при-

существующих в программных проектах на ранней стадии, когда функциональные пункты еще играют важную роль, стремление к минимизации работы, необходимой для получения не очень точных оценок, выглядит уместным.

Голландский метод

Ассоциация метрик программного обеспечения Нидерландов (NESMA) предлагает «индикативный» метод для вычисления функциональных пунктов на ранней стадии проекта (Stutzke 2005). В этом методе вместо всех входных/выходных элементов и запросов учитываются только внутренние логические файлы и внешние интерфейсные файлы. Затем индикативная метрика вычисляется по следующей формуле:

ФОРМУЛА № 8

$$\text{ИндикативныеФункциональныеПункты} = (35 \times \text{ВнутренниеЛогическиеФайлы}) + (15 \times \text{ВнешниеИнтерфейсныеФайлы}).$$

Коэффициенты 35 и 15 были получены посредством калибровки. В конечном итоге их рекомендуется заменить калибровочными коэффициентами, полученными по данным вашей среды.

Метрики, созданные этим методом, уступают по точности метрикам, полученным с использованием полной методики вычисления функциональных пунктов, описанной в разделе 18.2. Однако объем работы также получается существенно меньшим, поэтому такие приближения могут принести пользу при грубой оценке.

СОВЕТ № 82

Используйте голландский метод для получения приближенной оценки с минимальными затратами на ранней стадии проекта.

Элементы GUI

Вместо прямого вычисления функциональных пунктов можно начать с подсчета элементов графического интерфейса (GUI); это пример опосредованных методов оценки, о которых было рассказано в главе 12. Процесс состоит из следующих шагов.

1. Подсчитайте количество элементов GUI по категориям из табл. 18.4.
2. Преобразуйте количество элементов GUI в количество функциональных пунктов; для этого данные, полученные по табл. 18.4, подставляются в табл. 18.1.
3. Вычислите размер в строках кода по отношениям, показанным в табл. 18.3.

При использовании этого подхода следует понимать, какая неопределенность закладывается в вашу оценку. Вероятно, некоторая определенность существует уже в исходных количествах элементов GUI или ваших оценках. Дополнительная неопределенность вводится при преобразовании элементов GUI в функциональные

208 Глава 18 • Специфические проблемы при оценке размера

Таблица 18.3. Количество функциональных пунктов в строках кода

пункты. И наконец, при преобразовании функциональных пунктов в строки кода ее становится еще больше.

Таблица 18.4. Опосредованный подсчет элементов GUI вместо функциональных пунктов

Элемент GUI	Эквивалент в функциональных пунктах
Простое клиентское окно	Один внешний входной элемент низкой сложности для операций добавления, изменения и удаления (если они поддерживаются), плюс один внешний запрос низкой сложности
Среднее клиентское окно	Один внешний входной элемент средней сложности для операций добавления, изменения и удаления (если они поддерживаются), плюс один внешний запрос средней сложности
Сложное клиентское окно	Один внешний входной элемент высокой сложности для операций добавления, изменения и удаления (если они поддерживаются), плюс один внешний запрос высокой сложности
Средний отчет	Один внешний входной элемент средней сложности
Сложный отчет	Один внешний входной элемент высокой сложности
Любой файл	Один внутренний логический файл низкой сложности
Простой интерфейс	Один внешний входной элемент низкой сложности для получения данных; один внешний выходной элемент низкой сложности для выдачи данных
Средний интерфейс	Один внешний входной элемент средней сложности для получения данных; один внешний выходной элемент средней сложности для выдачи данных
Сложный интерфейс	Один внешний входной элемент высокой сложности для получения данных; один внешний выходной элемент высокой сложности для выдачи данных
Окно сообщения или диалоговое окно	Не учитываются по отдельности (а только в составе экрана, с которым они связаны)

СОВЕТ № 83

Используйте подсчет элементов GUI для получения приближенной оценки с минимальным объемом работы на ранней стадии проекта.

18.4. Сводка методов оценки размера

В этой главе и в других главах книги были представлены многочисленные методы оценки размера, в том числе и некоторые методы, оценивающие размер в строках кода. В табл. 18.5 приведена сводка методов, представлявшихся до настоящего момента.

Содержимое последнего столбца в действительности зависит от имеющихся у вас калибровочных данных. Самыми распространенными (и чаще всего применимыми на практике) показателями размера являются функциональные пункты и строки программного кода.

Таблица 18.5. Методы оценки размера

Метод	Глава	Оцениваемый показатель
Метод аналогий	11	Функциональность, функциональные пункты, веб-страницы, компоненты GUI, таблицы баз данных, определения интерфейсов, строки программного кода
Декомпозиция	10	Функциональность, функциональные пункты, веб-страницы, компоненты GUI, таблицы баз данных, определения интерфейсов, строки программного кода
Голландский метод	18	Функциональные пункты, строки программного кода
Оценочные программы	14	Функциональные пункты, строки программного кода
Метод функциональных пунктов	18	Функциональные пункты, строки программного кода
Нечеткая логика	12	Функциональные пункты, строки программного кода
Групповые обсуждения	13	Функциональность, пользовательские истории, требования, сценарии использования, функциональные пункты, веб-страницы, компоненты GUI, таблицы баз данных, определения интерфейсов, классы, функции/подпрограммы, строки программного кода
Элементы GUI	18	Функциональные пункты, строки программного кода
Стандартные компоненты	12	Функциональные пункты, строки программного кода
Широкополосный Дельфийский метод	13	Функциональность, пользовательские истории, требования, сценарии использования, функциональные пункты, веб-страницы, компоненты GUI, таблицы баз данных, определения интерфейсов, классы, функции/подпрограммы, строки программного кода
Что оценивается	Объем работ	Функциональные пункты, строки программного кода

Как упоминалось в главе 15, лучшие оценщики обычно применяют несколько методов оценки, а затем проверяют схождение или расхождение между оценками. Методы, перечисленные в табл. 18.5, предоставляют широкие возможности для составления альтернативных оценок с последующим сравнением результатов.

СОВЕТ № 84

При правильной методологии оценка размера становится основой для всех остальных оценок. Размер создаваемой системы является важнейшим фактором, определяющим ее стоимость. Для повышения точности используйте альтернативные оценки со сравнением результатов.

Дополнительные ресурсы

Garmus, David and David Herron. «Function Point Analysis: Measurement Practices for Successful Software Projects». Boston, MA: Addison-Wesley, 2001. В книге описана процедура вычисления функциональных пунктов, а также представлен ряд упрощенных методов.

Jones, Capers. «Applied Software Measurement: Assuring Productivity and Quality, 2d Ed.». New York, NY: McGraw-Hill, 1997. Джонс в подробностях описывает историю функциональных пунктов и представляет аргументы против метрики LOC.

Stutzke, Richard D. «Estimating Software-Intensive Systems». Upper Saddle River, NJ: Addison-Wesley, 2005. В главах 8 и 9 описаны дополнительные методы оценки размера, в том числе пункты пользовательских сценариев, прикладные пункты, веб-объекты и упрощенные методы функциональных пунктов. Также Стуцкे описывает оценку размеров для коммерческих продуктов.

www.construx.com/resources/surveyor/. На сайте имеется бесплатная утилита для подсчета строк кода Construx Surveyor.

www.ifpug.com. Группа IFPUG является наиболее авторитетным источником

информации по действующим правилам вычисления функциональных пунктов. www.nesma.nl. На сайте ассоциации NESMA (Netherlands Software Metrics Users Association) представлена информация о голландском методе.

Сложность: **Сложно** (один из основных элементов сложности) — это один из основных элементов сложности. Сложно означает, что для выполнения задачи требуется более высокий уровень знаний и навыков, чем для выполнения простых задач. Сложно означает, что для выполнения задачи требуется более высокий уровень знаний и навыков, чем для выполнения простых задач.

Сложные глагольные конструкции с вспомогательными глаголами и аффиксами являются высокой линейной единицей языка.

Любий обійтися без цієї книжки, якщо він хоче знати, що відбувається в Україні та як вона відповідає на сучасні виклики.

Модуль ПДЭ является дополнительной сложностью для выдачи данных Средний экран, являющийся базой для вывода информации, является основным элементом средней сложности.

Geert-Jan Oerdingen and David H. Egelman, "Protein Functionality in the Nucleoplasmic Membrane," *Journal of Cell Biology*, 1999, 145(3), 621-632.

Специфические проблемы при оценке объема работ

19

Вычисление оценки объема работ по предыдущим проектам

Несформализованная оценка с помощью методов IBSG

Область применения методов этой главы

	Неформальное сравнение с прошлыми проектами	Оценочные программы	Диаграммы средне- отраслевых объемов работ	Метод IBSG
Что оценивается	Объем работ	Объем работ	Объем работ	Объем работ
Размер проекта	М С –	М С Б	М С –	М С –
Стадия разработки	Ранняя–средняя	Ранняя–средняя	Ранняя	Ранняя
Итеративный или последовательный стиль	Оба	Оба	Последова- тельный	Последова- тельный
Возможная точность	Средняя	Высокая	Низкая–средняя	Низкая–средняя

В большинстве проектов объем работ оценивается по подробному списку задач. Однако на ранней стадии проекта наиболее точными оказываются оценки объема работ, вычисленные по оценкам размера. В этой главе описаны некоторые способы вычисления этих ранних оценок.

19.1. Факторы, влияющие на объем работ

Наибольшее влияние на объем работ оказывает размер создаваемой программы. Вторым по важности фактором является производительность вашей организации.

В табл. 19.1 представлены сведения о диапазонах производительности между разными программными проектами. Данные в таблице демонстрируют опасности, возникающие при использовании среднеотраслевых данных и при игнорировании эффекта издержек размера. Во встроенных программных проектах (таких, как проекты Lincoln Continental и IBM Checkout Scanner) код обычно генерируется

медленнее, чем в коммерческих проектах вроде Microsoft Excel. При использовании «средних» данных производительности для неподходящего типа проекта оценка может оказаться ошибочной на порядок и более.

В пределах одной отрасли также возможны значительные различия в производительности. Так, для Microsoft Excel 3.0 код генерировался примерно в 10 раз быстрее, чем для Lotus 123 v.3, хотя оба проекта были направлены на создание схожих продуктов и проводились примерно в один период времени.

Даже в рамках одной организации производительность может различаться из-за влияния издержек масштаба и других факторов. Так, в проектах Microsoft Windows NT код генерировался гораздо медленнее, чем в других проектах Microsoft, потому что это были системные, а не прикладные проекты, и они отличались гораздо большими размерами.

Таблица 19.1. Примеры различий в производительности среди разных типов программных проектов (* = Оценка)

Продукт	Эквивалент в новых строках кода	Человеко- лет	Год созда- ния	Приблизи- тельная стоимость в долларах	\$/LOC	LOC/ чело- веко- год
IBM Chief Programmer Team Project	83 000	9	1968	1 400 000*	\$17	9200
Lincoln Continental	83 000	35	1989	2 900 000*	\$35	2400
IBM Checkout Scanner	90 000	56	1989	4 900 000	\$55	1600
Microsoft Word for Windows 1.0	249 000	55	1989	8 500 000*	\$34	4500
NASA SEL Project	249 000	24	2002	3 700 000*	\$15	10 000
Lotus 123 v.3	400 000	263	1989	36 000 000	\$90	1500
Microsoft Excel 3.0	649 000	50*	1990	7 700 000	\$12	13 000
Citibank Teller Machine	780 000	150	1989	22 000 000	\$28	5200
Windows NT 3.1 (первая версия)	2 880 000	2000*	1994	200 000 000	\$70	1400
Космический шаттл	25 600 000	22 096	1989	2 000 000 000	\$77	1200

Источники: «Chief Programmer Team Management of Production Programming» (Baker 1972), «Microsoft Corporation: Office Business Unit» (Iansiti 1994), «How to Break the Software Logjam» (Schlender 1989), «Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules» (NASA, 1991), «Microsoft Secrets» Cusumano and Selby 1995).

Согласно табл. 19.1, наименьшая производительность в строках кода на чело-веко-год наблюдалась при разработке программного обеспечения космического шаттла, но называть эту группу разработчиков непроизводительной было бы ошибкой. Для проектов такого размера вероятность полного провала превышает 50 % (Jones 1998). Тот факт, что проект был завершен, сам по себе является крупным достижением. По производительности он всего на 15 % уступал проекту Windows NT, несмотря на то, что программное обеспечение шаттла в 10 раз превышало по размеру проект Windows NT.

Если вы не располагаете историческими данными о производительности вашей организации, можно использовать среднеотраслевые цифры для разных типов

программ: внутренних бизнес-систем, критических систем, игр, драйверов устройств и т. д. При этом следует остерегаться 10-кратных различий в производительности разных организаций в пределах одной отрасли. Если вы располагаете историческими данными производительности в вашей организации, лучше используйте их для преобразования оценки размеров в оценку объема работ, вместо того, чтобы использовать среднеотраслевые данные.

19.2. Вычисление объема работ по размеру

Занявшись вычислением оценки объема работ по оценке размера, мы начнем сталкиваться с некоторыми слабостями неформальных методов и будем вынуждены в большей степени задействовать научные методы.

Вычисление оценки объема работ посредством неформального сравнения с прошлыми проектами

Если исторические данные проектов лежат в узком диапазоне размеров (скажем, верхняя граница отличается от нижней не более чем в 3 раза), вероятно, вы можете воспользоваться линейной моделью для вычисления оценки объема работ нового проекта по результатам похожих прошлых проектов. В табл. 19.2 представлен пример данных прошлых проектов, которые могут стать основой для такой оценки.

Таблица 19.2. Пример данных о производительности прошлых проектов, заложенных в основу для оценки объема работ

Проект	Размер (LOC)	Срок (календарные месяцы)	Объем работ (человеко-месяцы)	Производительность (LOC/человеко-месяц)	Комментарии
Проект А	33 842	8,2	21	1612	
Проект В	97 614	12,5	99	986	
Проект С	7444	4,7	2	3722	Не используется — слишком мал для сравнения
Проект D	54 322	11,3	40	1358	
Проект Е	340 343	24,0	533	639	Не используется — слишком велик для сравнения

Предположим, вы оцениваете эффективность новой бизнес-системы. По вашей оценке, размер нового продукта составит от 65 000 до 100 000 строк кода Java, с наиболее вероятным размером в 80 000 строк. Проект С слишком мал, чтобы использоваться в целях сравнения, потому что его размер составляет менее 1/3 от нижней границы диапазона. Проект Е слишком велик для использования в целях сравнения, потому что он более чем в 3 раза превышает верхнюю границу. Таким образом, ваши исторические данные производительности лежат в диапазоне от

986 строк кода на человека-месяц (проект В) до 1612 строк кода на человека-месяц (проект А). Деление нижней границы диапазона размера на максимальную производительность дает нижнюю оценку в 40 человеко-месяцев, а деление на минимальную производительность — верхнюю оценку в 101 человеко-месяц. Таким образом, оценка объема работ представляет собой диапазон от 40 до 101 человека-месяца.

Хорошее рабочее предположение заключается в том, что диапазон включает 68 % возможных результатов (то есть ± 1 стандартное отклонение, если только у вас нет причин предполагать иное). По табл. 10.6 можно проверить другие вероятности, входящие в диапазон от 40 до 101 человека-месяца.

Какой объем работ включается в эту оценку?

Поскольку оценка создавалась по историческим данным, в нее включен тот объем работ, который задействован в исторических данных. Если в исторические данные включались работы только по разработке и тестированию и только для части проекта от завершения требований до тестирования системы, — именно они будут отражены в оценке. Если в исторические данные также вошли работы по постановке требований, управлению проектом и подготовке документации — значит, они тоже будут присутствовать в оценке.

В общем случае оценки, основанные на среднеотраслевых данных, включают всю техническую работу, но не работу по управлению и все операции разработки, кроме требований. На практике данные, включаемые в вычисления среднеотраслевых показателей, не всегда следуют этим предположениям; отчасти это объясняет такой разброс в среднеотраслевых данных.

19.3. Вычисление оценки объема работ научными методами

Научные методы оценки дают несколько иные результаты, чем неформальные сравнения с прошлыми проектами. Если ввести те же предположения в Construx Estimate (то есть воспользоваться приведенными историческими данными для калибровки оценки), вы получите ожидаемый результат в 80 человеко-месяцев; он лежит в середине диапазона, полученного менее формальным методом. Construx Estimate дает оценку лучшего случая (достоверность 20 %) в 65 человеко-месяцев и оценку худшего случая (достоверность 80 %) в 94 человека-месяца.

Если откалибровать Construx Estimate среднеотраслевыми данными вместо исторических, он выдает номинальную оценку в 84 человека-месяца и гораздо более широкий диапазон 20–80 % от 47 до 216 человеко-месяцев. Это лишний раз показывает, как важно использовать исторические данные там, где это возможно.

СОВЕТ № 85

Используйте оценочные программы, основанные на формальных методах оценки, для более точного вычисления оценки объема работ по имеющейся оценке размера.

19.4. Среднеотраслевые графики объема работ

Если вы не располагаете собственными историческими данными, для грубой оценки объема работ можно воспользоваться графиком — примеры таких графиков приведены на рис. 19.1–19.9. Светлая линия на этих рисунках представляет общий объем технических работ проекта (включая разработку, контроль качества и тестирование) при среднеотраслевом уровне производительности. Верхняя черная линия представляет объем работ, на единицу стандартного отклонения превышающий средний. Я не стал приводить аналогичную линию на единицу стандартного отклонения ниже среднего. Если вы не располагаете собственными историческими данными и используете эти графики, это говорит о том, что разработка организована в лучшем случае на среднем уровне. В таких случаях лучше проявить умеренность в оценке и использовать среднеотраслевые или более низкие показатели производительности.

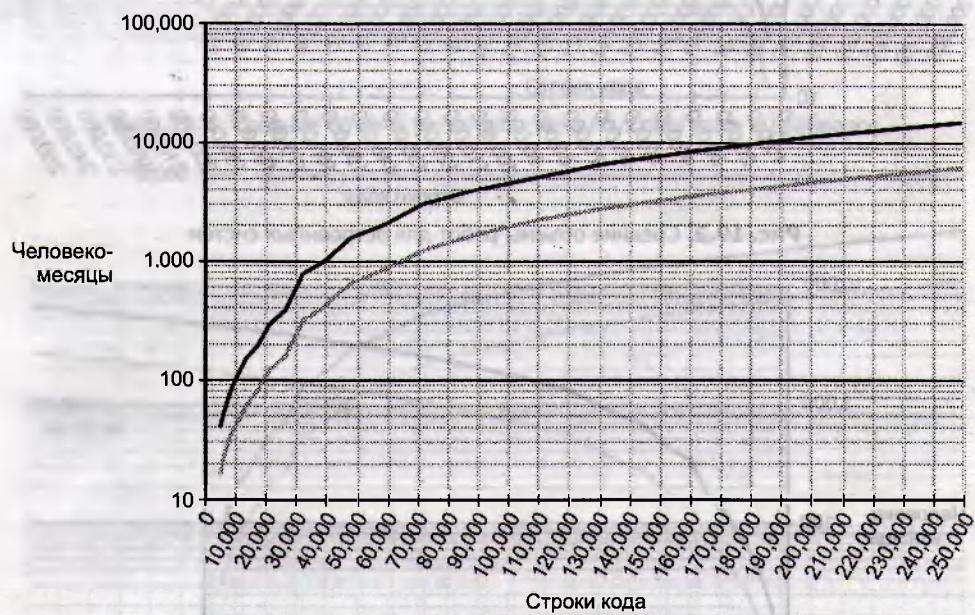


Рис. 19.1. Средние объемы работ для проектов реального времени

На графиках представлены проекты размером до 250 000 строк кода, а максимальный объем работ по некоторым из них превышает 10 000 человеко-месяцев. Для проектов такого размера следует понимать, что применение более мощных и точных методов оценки способно улучшить планирование и обеспечить экономию сотен тысяч долларов. Эксперт в области оценки Кейперс Джонс неоднократно отмечал, что применение ручных методов для оценки проектов свыше 1000 функциональных пунктов или 100 000 строк кода создает существенную ошибку, а отказ от применения оценочных программ в проектах свыше

216 Глава 19 • Специфические проблемы при оценке объема работ

5000 функциональных пунктов или 500 000 строк кода должно рассматриваться как признак некомпетентности в управлении (Jones 1994, Jones 2005).

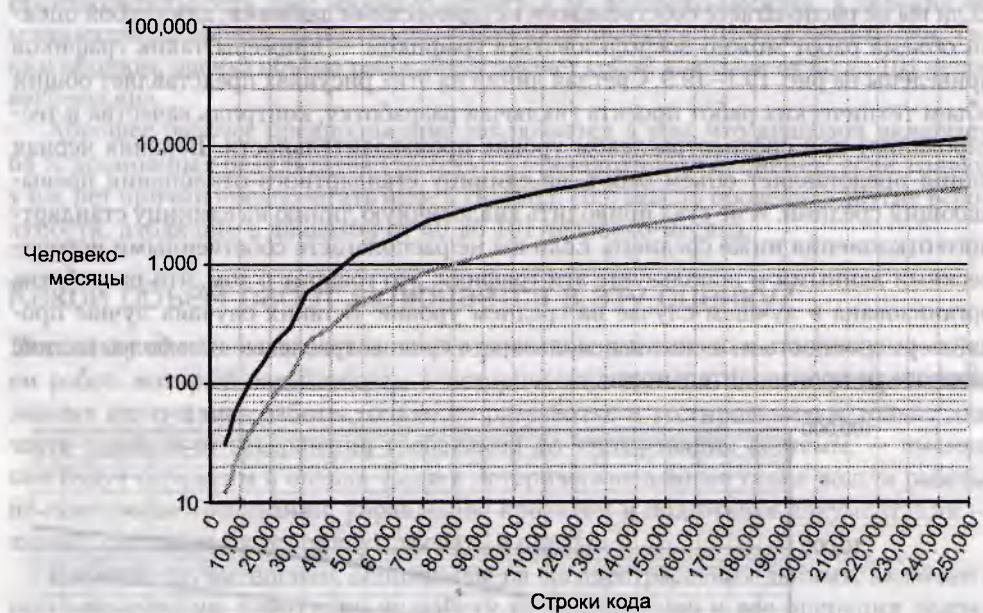


Рис. 19.2. Средние объемы работ для встроенных систем

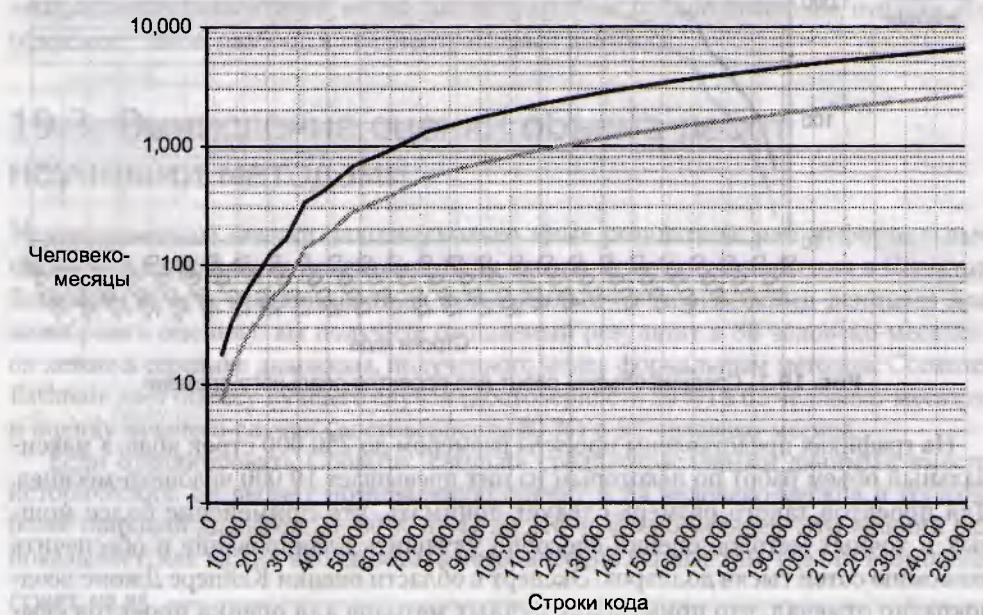


Рис. 19.3. Средние объемы работ для телекоммуникационных проектов

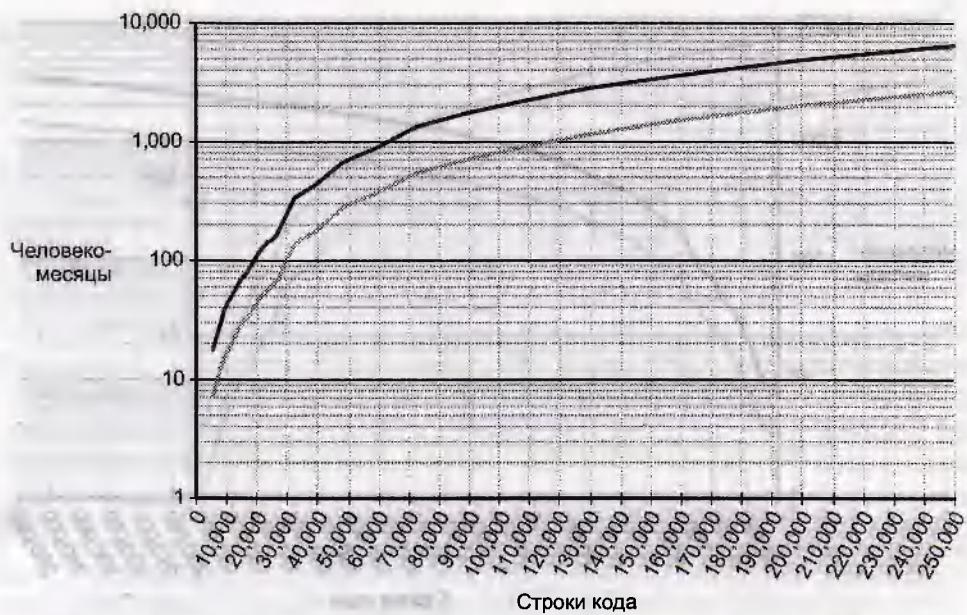


Рис. 19.4. Средние объемы работ для системных программ и драйверов

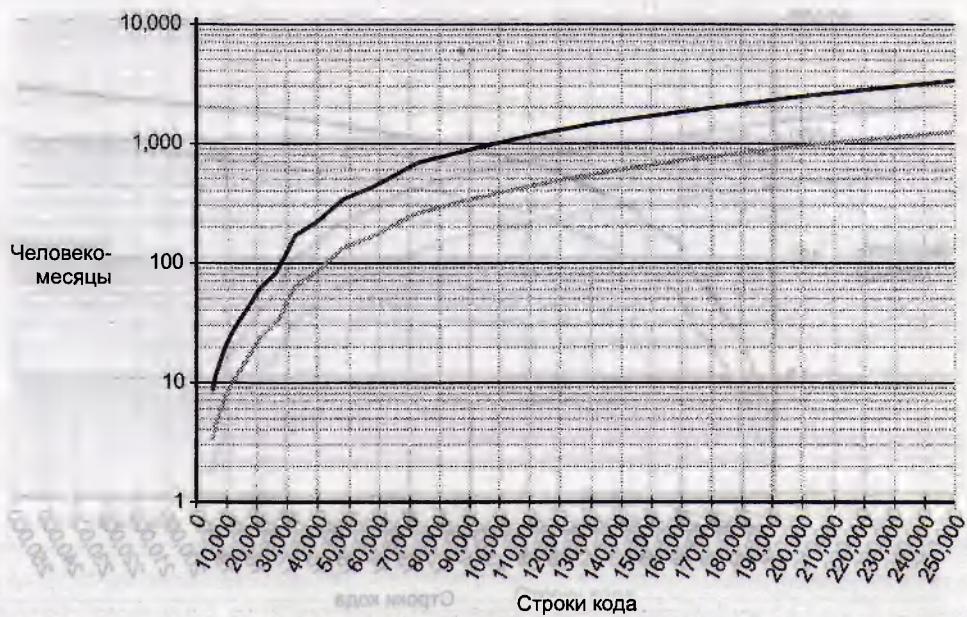


Рис. 19.5. Средние объемы работ для научных систем и инженерных исследовательских проектов

218 Глава 19 • Специфические проблемы при оценке объема работ

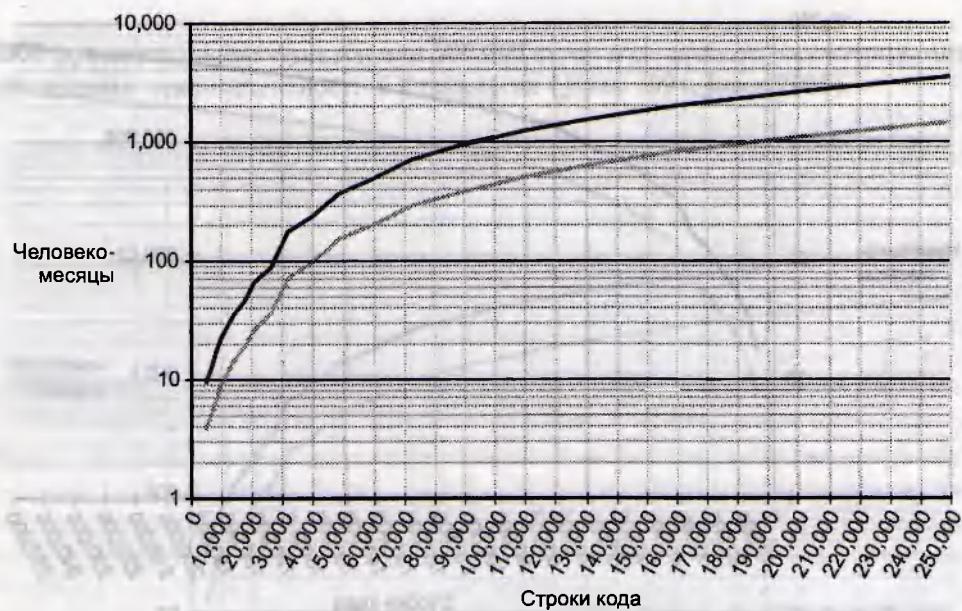


Рис. 19.6. Средние объемы работ для коммерческих программных проектов

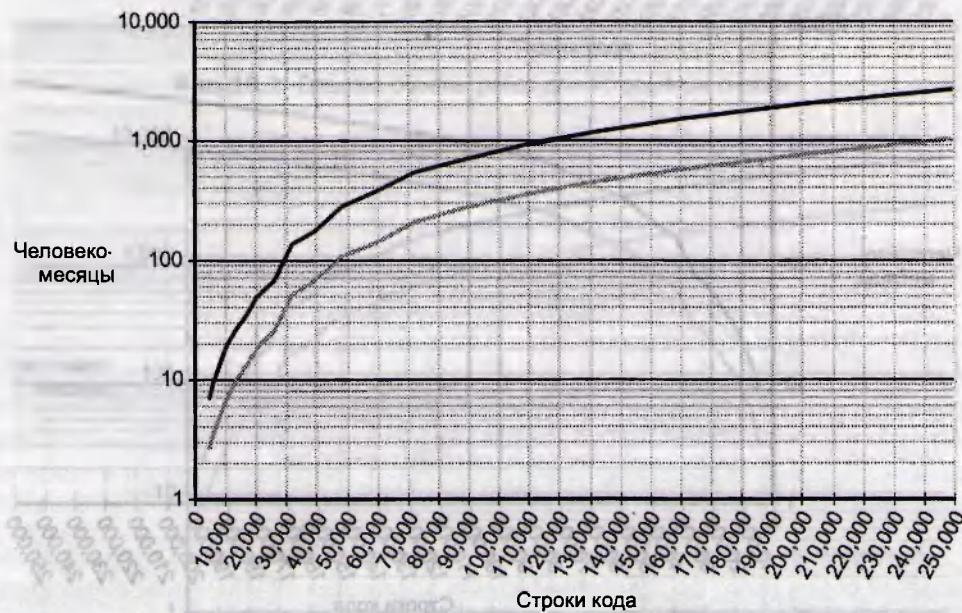


Рис. 19.7. Средние объемы работ для публичных проектов интернет-систем

Рис. 19.3. Средние объемы работ для телекоммуникационных проектов

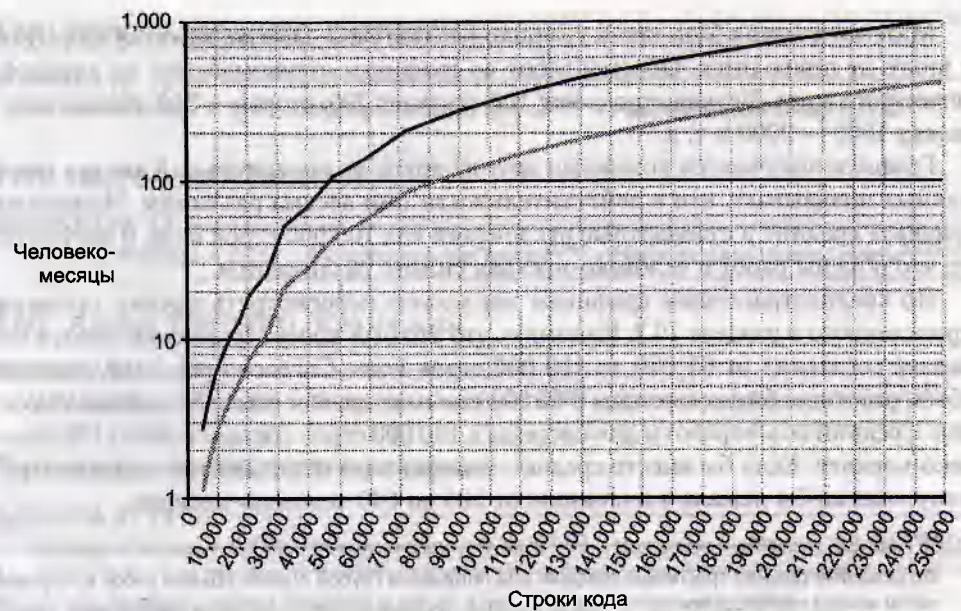


Рис. 19.8. Средние объемы работ для интрасетевых проектов

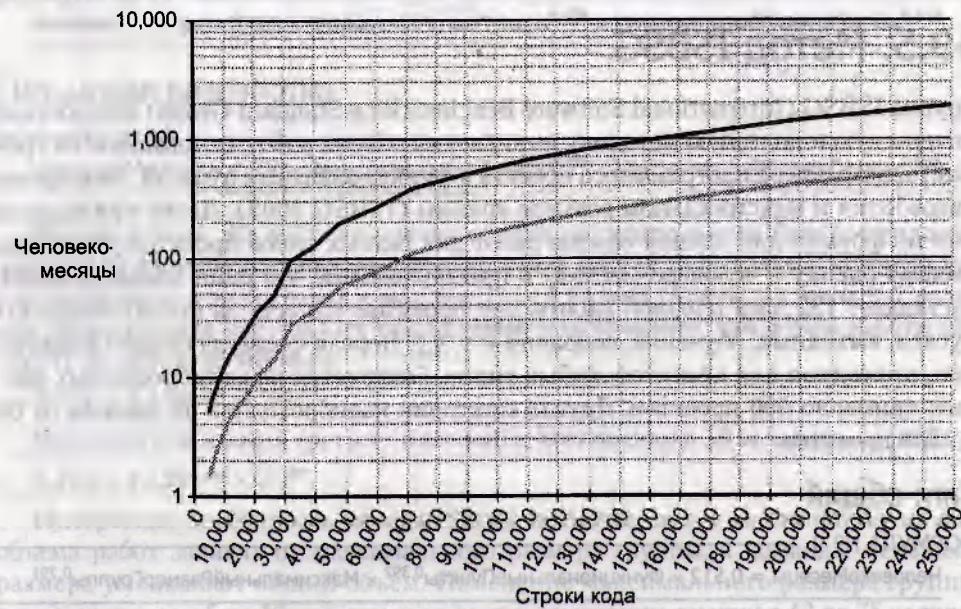


Рис. 19.9. Средние объемы работ для бизнес-систем

220 Глава 19 • Специфические проблемы при оценке объема работ

Математическая база таких графиков достаточно сложна, поэтому формулы в книге не приводятся. Объемы работ по графикам отсчитываются по логарифмической шкале. Первая строка над 100 означает 200, вторая — 300, первая строка над 1000 — 2000 и т. д.

Графики получаются похожими друг на друга, но внимательный анализ точек данных показывает, что в действительности они весьма различны. Например, сравните средние и стандартные отклонения над 100 000 строк кода, и вы увидите, что объемы работ в человеко-месяцах сильно различаются.

По среднеотраслевым графикам мы можем пересмотреть оценку примера, приведенного в разделе 19.2. Напомню, что это был проект бизнес-системы, а его размер составлял от 65 000 до 100 000 строк кода. Согласно рис. 19.9, средний объем работ для бизнес-системы в 65 000 строк составляет около 85 человеко-месяцев. Средний объем работы для системы в 100 000 строк составит около 170 человеко-месяцев. Если бы вместо среднего значения мы использовали верхнюю границу, оценка бы лежала в диапазоне от 300 до 600 человеко-месяцев.

СОВЕТ № 86

Используйте среднеотраслевые графики для получения грубой оценки объема работ в широкой части конуса неопределенности. Помните, что в крупных проектах затраты, связанные с применением более мощных методов оценки, быстро окупаются.

19.5. Метод ISBSG

Группа ISBSG (International Software Benchmarking Standard Group) разработала интересную и полезную методику вычисления объема работ, основанную на трех факторах: размере программного проекта в функциональных пунктах, типа среди разработки и максимальном размере группы (ISBSG 2005). Далее приводятся восемь формул для оценки объема работ для разных типов проектов. Формулы выдают оценку в человеко-месяцах в предположении, что один человеко-месяц составляет 132 часа плотной работы над проектом (то есть за исключением отпусков, выходных, обучения, собраний и т. д.). Первая формула общего назначения, пригодная для проектов любых типов, базируется на калибровочных данных примерно 600 проектов. Другие категории калибровались по данным от 63 до 363 проектов.

Тип: общий

ФОРМУЛА № 9

$$\text{ЧеловекоМесяцы} = 0,512 \times \text{ФункциональныеПункты}^{0,392} \times \text{МаксимальныйРазмерГруппы}^{0,791}.$$

Тип проекта: проект для больших компьютеров (мейнфреймов)

ФОРМУЛА № 10

$$\text{ЧеловекоМесяцы} = 0,685 \times \text{ФункциональныеПункты}^{0,507} \times \text{МаксимальныйРазмерГруппы}^{0,464}.$$

Тип: проект среднего диапазона**ФОРМУЛА № 11**

$$\text{ЧеловекоМесяцы} = 0,472 \times \text{Функциональные Пункты}^{0,375} \times \text{Максимальный Размер Группы}^{0,882}.$$

Тип: настольная система**ФОРМУЛА № 12**

$$\text{ЧеловекоМесяцы} = 0,157 \times \text{Функциональные Пункты}^{0,591} \times \text{Максимальный Размер Группы}^{0,810}.$$

Тип: языки третьего поколения**ФОРМУЛА № 13**

$$\text{ЧеловекоМесяцы} = 0,425 \times \text{Функциональные Пункты}^{0,488} \times \text{Максимальный Размер Группы}^{0,697}.$$

Тип: языки четвертого поколения**ФОРМУЛА № 14**

$$\text{ЧеловекоМесяцы} = 0,317 \times \text{Функциональные Пункты}^{0,472} \times \text{Максимальный Размер Группы}^{0,784}.$$

Тип: доработка существующих проектов**ФОРМУЛА № 15**

$$\text{ЧеловекоМесяцы} = 0,669 \times \text{Функциональные Пункты}^{0,338} \times \text{Максимальный Размер Группы}^{0,758}.$$

Тип: новая разработка**ФОРМУЛА № 16**

$$\text{ЧеловекоМесяцы} = 0,520 \times \text{Функциональные Пункты}^{0,385} \times \text{Максимальный Размер Группы}^{0,866}.$$

Предположим, вы создаете оценку объема для настольного бизнес-приложения в 1450 функциональных пунктов на языке Java (та же система, которую мы уже оценивали в этой главе), а максимальный размер группы составляет 7 человек. Формула для настольных приложений подсказывает, что объем работы составит 56 человеко-месяцев:

$$0,157 \times 1,450^{0,591} \times 7^{0,810}.$$

Формула для языков третьего поколения дает оценку в 58 человеко-месяцев:

$$0,425 \times 1,450^{0,488} \times 7^{0,697}.$$

Интересная особенность метода ISBSG заключается в том, что формулы для объема работ зависят от максимального размера группы, а команды меньшего размера уменьшают общий объем. Изменение максимального размера группы в этом примере с 5 до 10 человек приводит к изменению оценки с 43 до 75 человеко-месяцев. С точки зрения оценки это создает неопределенность. С точки зрения управления проектом эти различия могут заставить вас использовать группу меньшего размера вместо большей (эта тема дополнительно рассматривается в подразделе «Сокращение срока и размера группы» раздела 20.6).

СОВЕТ № 87

Используйте метод ISBSG для получения грубой оценки объема работ. Сравните его с другими методами и проанализируйте сходжение или расхождение оценок.

19.6. Сравнение оценок объема работ

Для проверки реалистичности этих оценок можно сравнить четыре разных метода оценки объема работ, представленные в этой главе:

- неформальные сравнения с прошлыми проектами;
- использование оценочных программ;
- использование среднеотраслевых графиков;
- метод ISBSG.

На рис. 19.10 показано, как выглядят диапазоны оценок, полученных этими методами.

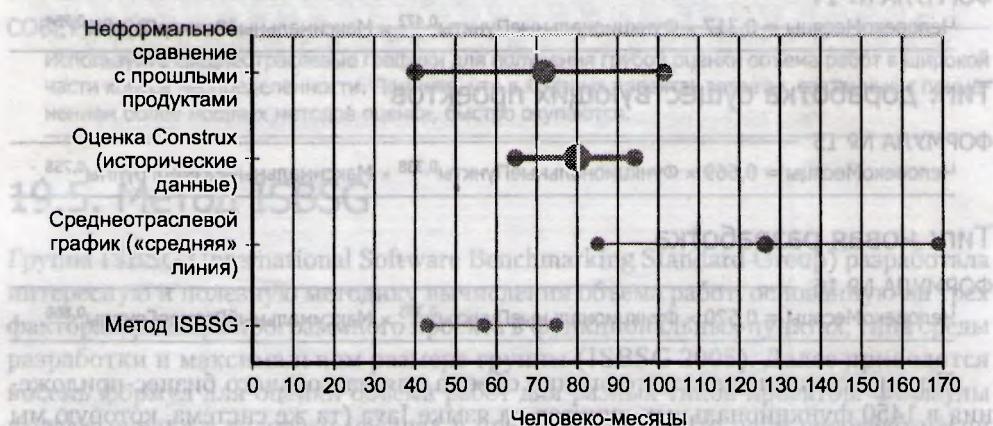


Рис. 19.10. Диапазоны оценок, полученных методами, рассматриваемыми в этой главе. Относительные размеры точек и толщина линий представляют весовые коэффициенты, которые я присвоил каждому из представленных методов

На графике показан диапазон оценок от 40 до 110 человеко-месяцев. Как в методе ISBSG, так и на среднеотраслевых графиках используются среднеотраслевые данные, поэтому я присвоил им меньший вес, чем методам, основанным на исторических данных. При неформальном сравнении с прошлыми проектами я присваивал наибольший вес самым похожим проектам (включая максимальное сходство размеров).

С учетом всех факторов в данном случае я бы представил оценку от 65 до 100 человеко-месяцев с ожидаемым результатом в 80 человеко-месяцев. Можно было бы предположить, что ожидаемый результат попадет в середину диапазона 65–100, но объем работ часто смещается к нижней границе диапазона из-за воздействия факторов, описанных в разделе 1.4.

СОВЕТ № 88

Не все методы оценки равны. При поиске схождения или расхождения между оценками присваивайте большие весовые коэффициенты методам, дающим более точные результаты.

Дополнительные ресурсы

Boehm, Barry, et al. «Software Cost Estimation with Cocomo II». Reading, MA: Addison Wesley, 2000. Модель оценки Cocomo II, описанная в книге, содержит формулы для преобразования оценки размеров, выраженной в строках программного кода, в объем работ. Не забывайте о проблемах субъективных «регуляторов».

ISBSG. «Practical Project Estimation, 2nd Edition: A Toolkit for Estimating Software Development Effort and Duration». Victoria, Australia: International Software Benchmarking Standards Group, February 2005. В книге приведено множество полезных формул для вычисления оценки объема работ по оценке размера. Книга приятно удивляет объективными характеристиками точности своих формул; в нее включены примеры размеров и тестовых значений, которые могут использоваться для проверки точности формул.

Putnam, Lawrence H. and Ware Myers. «Measures for Excellence: Reliable Software On Time, Within Budget». Englewood Cliffs, NJ: Yourdon Press, 1992. Модель Путнэма, описанная в книге, преобразует оценку размера, выраженную в строках кода, в оценку объема работ.

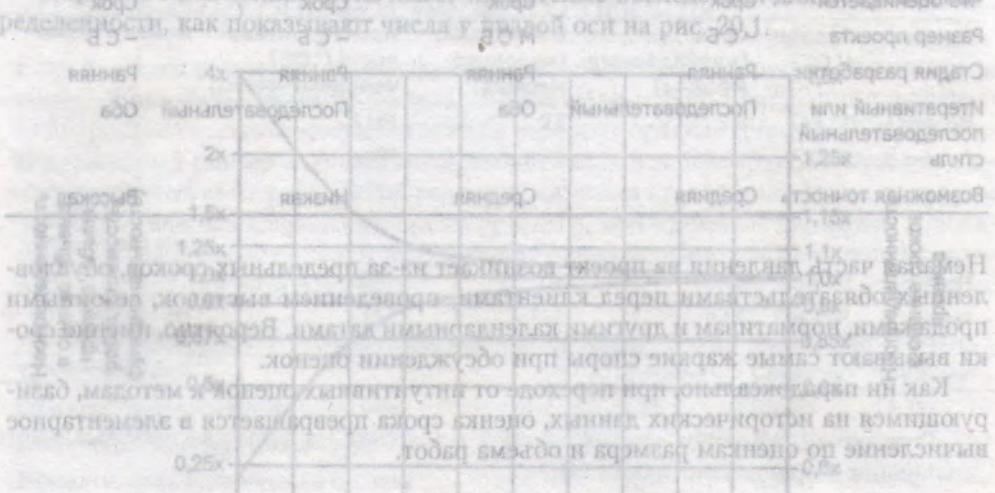


Рис. 20.1. Базовая формула для вычисления срока

Формула графа является линией на логарифмической оси, из-за которой диапазон неопределенности для объема работ на рис. 20.1 гораздо шире, чем на рис. 18.2. Объем работ возрастает пропорционально кубику проекта, тогда как срок возрастает пропорционально кубическому корню из объема работ.

Специфические проблемы при оценке сроков

20

Область применения методов этой главы

	Базовая формула для вычисления срока	Неформальное сравнение с прошлыми проектами	Метод оценки первого порядка	Оценочные программы
Что оценивается	Срок	Срок	Срок	Срок
Размер проекта	– С Б	М С Б	– С Б	– С Б
Стадия разработки	Ранняя	Ранняя	Ранняя	Ранняя
Итеративный или последовательный стиль	Последовательный	Оба	Последовательный	Оба
Возможная точность	Средняя	Средняя	Низкая	Высокая

Немалая часть давления на проект возникает из-за предельных сроков, обусловленных обязательствами перед клиентами, проведением выставок, сезонными продажами, нормативами и другими календарными датами. Вероятно, именно сроки вызывают самые жаркие споры при обсуждении оценок.

Как ни парадоксально, при переходе от интуитивных оценок к методам, базирующимся на исторических данных, оценка срока превращается в элементарное вычисление по оценкам размера и объема работ.

20.1. Базовая формула для вычисления срока

Приближенная оценка срока на ранней стадии проекта может производиться по базовой формуле:

ФОРМУЛА № 17

$$\text{Срок в месяцах} = 3,0 \times \text{Человеко-месяцы}^{1/3}$$

Если вы подзабыли математику, на всякий случай напомню, что показатель степени $1/3$ означает кубический корень.

Иногда $3,0$ заменяется на $2,0, 2,5, 3,5, 4,0$ или другое число, но основная идея, по которой срок вычисляется как кубический корень из объема работ, признается почти всеми экспертами в области оценки (конкретный множитель может быть получен посредством калибровки по историческим данным организации). Барри Бем заметил в 1981 году, что эта формула стала одним из самых растиражированных результатов в программотехнике (Boehm 1981). Анализ, проведившийся на протяжении нескольких прошедших десятилетий, подтвердил состоятельность формулы (Boehm 2000, Stutzke 2005).

Допустим, на построение проекта вам потребуется 80 человеко-месяцев. Вычисленные по формуле сроки лежат в диапазоне от $8,6$ до $17,2$ месяца, в зависимости от выбора коэффициента в диапазоне от $2,0$ до $4,0$. Номинальный срок будет равен ($3,0 \times 80^{1/3}$), то есть $12,9$ месяца. (Я не рекомендую представлять оценку срока с такой четкостью и привожу ее лишь для того, чтобы вам было проще следить за вычислениями.)

СОВЕТ № 89

Используйте базовую формулу для ранней оценки срока в средних и крупных проектах.

Формула вычисления срока имеет интересные последствия для конуса неопределенности, как показывают числа у правой оси на рис. 20.1.

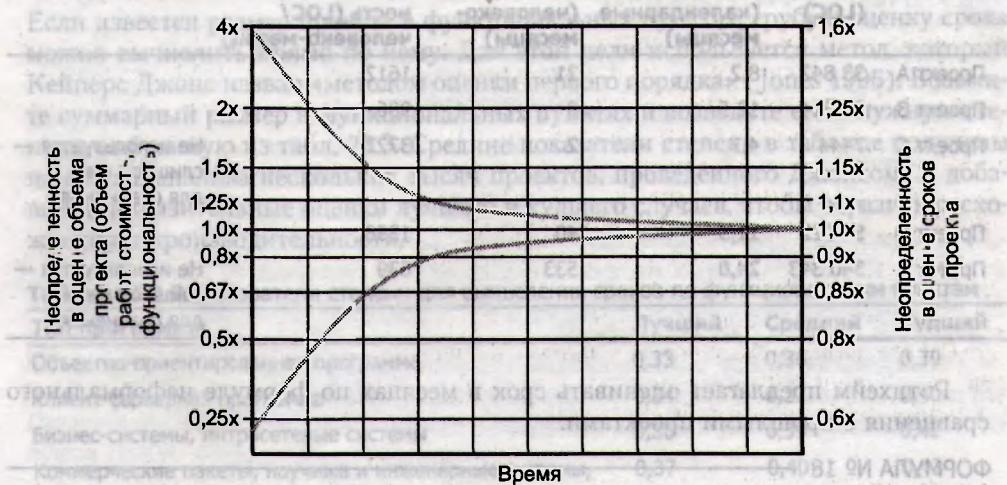


Рис. 20.1. Конус неопределенности с поправками сроков на правой оси. Неопределенность сроков гораздо меньше неопределенности объема, потому что срок вычисляется как кубический корень из объема работ

Формула срока является одной из причин, из-за которых диапазоны неопределенности для объема работ на рис. 20.1 гораздо шире, чем для сроков. Объем работ возрастает пропорционально объему проекта, тогда как срок возрастает пропорционально кубическому корню из объема работ.

Формула срока неявно подразумевает, что размер группы может регулироваться до размера, следующего из уравнения. При фиксированном размере группы срок не будет меняться пропорционально кубическому корню объема; он будет меняться более значительно в зависимости от ограничений размера группы. Эта проблема более подробно рассматривается в разделе 20.7.

Базовая формула для вычисления срока не предназначена для малых проектов или на поздних фазах больших проектов. Когда становятся известными имена конкретных людей, работающих над проектом, переключайтесь на другой метод.

20.2. Вычисление срока посредством неформальных сравнений с прошлыми проектами

Уильям Ротцхейм (William Roetzheim) предложил оценивать сроки новых проектов по соотношению сроков и объемов работ в прошлых проектах (Roetzheim 1988, Stutzke 2005). Возьмем те же проекты, которые были использованы в главе 19; для удобства их данные повторены в табл. 20.1.

Таблица 20.1. Пример данных о производительности прошлых проектов, заложенных в основу для оценки сроков

Проект	Размер (LOC)	Срок (календарные месяцы)	Объем работ (человеко-месяцы)	Производительность (LOC/человеко-месяц)	Комментарии
Проект А	33 842	8,2	21	1612	
Проект В	97 614	12,5	9	986	
Проект С	7444	4,7	2	3722	Не используется — слишком мало для сравнения
Проект D	54 322	11,3	40	1358	
Проект Е	340 343	24,0	533	639	Не используется — слишком велик для сравнения

Ротцхейм предлагает оценивать срок в месяцах по формуле неформального сравнения с прошлыми проектами:

ФОРМУЛА № 18

$$\text{ОценкаСрока} = \text{ПрошлыйСрок} \times (\text{ОценкаОбъемаРабот}/\text{ПрошлыйОбъемРабот})^{1/3}.$$

Показатель степени $1/3$ используется в тех проектах, которые в книге отнесены к категории средних и крупных (более 50 человеко-месяцев). Для более мелких проектов следует использовать показатель степени $1/2$.

В главе 19 объем работ оценивался от 65 до 100 человеко-месяцев при наиболее вероятной оценке в 80 человеко-месяцев. Таким образом, мы получаем оцениваемые сроки из прошлых проектов, приведенные в табл. 20.2.

Таблица 20.2. Оцениваемые сроки из прошлых проектов

Проект	Исторические данные		Оценки		
	Прошлый срок (календарные месяцы)	Прошлый объем работ (человеко- месяцы)	Низкая оценка (65 человеко- месяцев)	Номинальная оценка (80 человеко- месяцев)	Высокая оценка (100 человеко- месяцев)
Проект А	8,2	21	12,0	12,8	13,8
Проект В	12,5	99	10,8	11,6	12,5
Проект D	11,3	40	13,2	14,2	15,3

По методике Ротцхайма диапазон «лучший-худший случай» составляет 10,8–17,3 месяца. Я бы оценил ожидаемый случай простым усреднением трех номинальных оценок из таблицы, после чего вычислил верхнюю и нижнюю границы диапазона усреднением верхних и нижних оценок из таблицы. Вычисления дают номинальный срок в 12,9 месяца с диапазоном 12,0–13,9 месяца.

СОВЕТ № 90

Используйте формулу неформального сравнения с прошлыми проектами для ранней оценки срока в проектах любого размера.

20.3. Метод оценки первого порядка

Если известен размер проекта в функциональных пунктах, грубую оценку срока можно вычислить прямо по нему. Для этой цели используется метод, который Кейперс Джонс назвал «методом оценки первого порядка» (Jones 1996). Возьмите суммарный размер в функциональных пунктах и возведите его в нужную степень, выбранную из табл. 20.3. Средние показатели степени в таблице получены на основе анализа нескольких тысяч проектов, проведенного Джонсом. Я добавил приблизительные оценки лучшего и худшего случаев, чтобы отразить расхождения в производительности.

Таблица 20.3. Показатели степени для вычисления сроков по функциональным пунктам

Тип программы	Лучший	Средний	Худший
Объектно-ориентированная программа	0,33	0,36	0,39
Клиент-серверная программа	0,34	0,37	0,40
Бизнес-системы, интрасетевые системы	0,36	0,39	0,42
Коммерческие пакеты, научные и инженерные системы, публичные интернет-системы	0,37	0,40	0,43
Встроенные системы, телекоммуникации, драйверы устройств, системные программы	0,38	0,41	0,44

Источник: Адаптировано по материалам «Determining Software Schedules» (Jones 1995c) и «Estimating Software Costs» (Jones 1996).

Если общее количество функциональных пунктов проекта равно 1450, а ваша организация занимается разработкой бизнес-систем со средней производительностью,

возведение 1450 в степень 0,39 ($1450^{0,39}$) дает грубую оценку в 17 календарных месяцев. Если же ваша организация занимается разработкой бизнес-систем и является лучшей в своем классе, то 1,450 возводится в степень 0,36, что дает срок в 14 месяцев. Если же вы разрабатываете объектно-ориентированные бизнес-системы, показатели степени для объектно-ориентированного программного обеспечения дают диапазон от 11 до 17 месяцев. Таким образом, можно сделать вывод, что реальный срок лежит где-то в диапазоне от 11 до 17 месяцев.

Эта методика не заменит более тщательной оценки срока, но она предоставляет простое средство получения грубой оценки, которое все же лучше простых догадок. Кроме того, она позволяет быстро проверить задачи на реалистичность. Если вы хотите разработать бизнес-систему в 1450 пунктов за 9 месяцев, подумайте заново. Для самых передовых организаций срок составит от 11 до 14 месяцев, а большинство организаций к передовым не относятся. Метод оценки первого порядка поможет вам быстрее узнать, не нужно ли внести изменения в ваши предположения о функциональности и/или сроках.

СОВЕТ № 91

Используйте метод оценки первого порядка для получения неточной (но требующей крайне малых усилий) оценки срока на ранней стадии проекта.

20.4. Вычисление оценки срока научными методами

В конечном итоге неформальные методы плохо подходят для получения точной оценки. Сроки слишком сильно изменяются в зависимости от множества факторов. Самый простой и точный способ вычисления оценки основан на использовании таких вспомогательных инструментов, как Construx Estimate¹.

Что произойдет, если мы используем Construx Estimate для вычисления оценки срока в рассматриваемом примере? При калибровке историческими данными объемов работы и сроков, представленными в главе 19, Construx Estimate вычисляет срок в 12,2 календарных месяца, с диапазоном достоверности 20–80 % от 11,6 до 12,9 месяца. При среднеотраслевых данных номинал составляет 15,8 месяца, а диапазон – от 13,2 до 21,5 месяца!

Расхождения между номинальными значениями и диапазоном, вычисленными по историческим и среднеотраслевым данным, демонстрируют преимущества от использования исторических данных.

20.5. Сжатие графика и размер группы

После того как номинальная оценка будет вычислена, часто возникает вопрос: «На сколько можно будет сократить срок в случае необходимости?» Ответ зависит от гибкости набора функций. Если из проекта можно исключать те или иные

¹ Программу Construx Estimate можно бесплатно загрузить по адресу www.construx.com/estimate.

функции, срок можно сократить насколько потребуется, в зависимости от вашей готовности к усечению функций. На выполнение меньшей работы требуется меньше времени, что вполне логично.

Если функциональность жестко фиксирована, сокращение срока может быть достигнуто только за счет добавления персонала, который бы выполнял больше работы за меньшее время... что до определенного момента тоже вполне логично.

За несколько прошедших десятилетий многие исследователи, занимающиеся оценкой программных проектов, анализировали последствия попыток сжатия номинальных сроков. Результаты их исследований показаны на рис. 20.2.

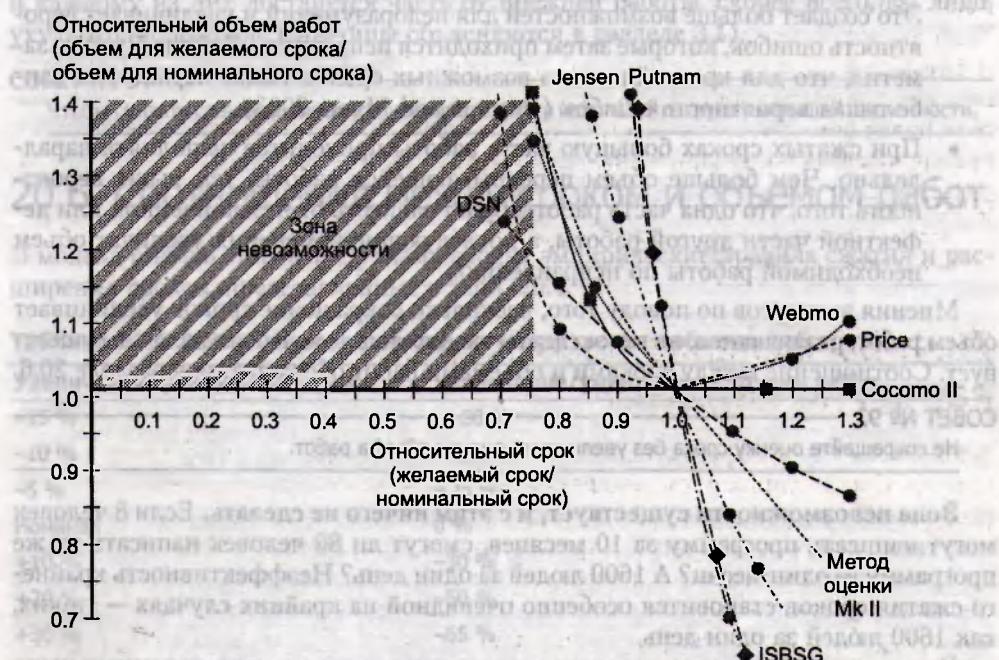


Рис. 20.2. Последствия сжатия или расширения номинальных сроков и зона невозможности.
Все исследователи обнаружили, что возможности сжатия срока ограничены

Источник: Адаптировано по материалам «Software Sizing and Estimating: Mk II» (Symons 1991), «Software Cost Estimation with Cocomo II» (Boehm et al. 2000), «Estimating Web Development Costs: There Are Differences» (Reifer 2002) и «Practical Project Estimation», 2nd Edition (ISBSG 2005).

По горизонтальной оси отсчитывается отношение между номинальным и сжатым сроками. Скажем, отметка 0,9 означает, что сжатый срок занимает 0,9 от времени, соответствующего номинальному сроку (то есть 90 % от номинала). Вертикальная ось представляет общий объем работ, необходимый при сжатии или расширении срока, по сравнению с номиналом. Значение 1,3 указывает, что сжатый срок требует в 1,3 раза больше работы, чем номинальный.

Из графика на рис. 20.2 можно сделать ряд выводов.

Сокращение номинального срока приводит к увеличению объема работ. Все исследователи сошлись на том, что сокращение номинального срока увеличивает общий объем работ. Если номинальный срок составляет 12 месяцев для группы из 7 человек, то простое увеличение персонала до 12 человек не позволит сократить срок до 7 месяцев.

Сокращение сроков приводит к увеличению объема работы по нескольким причинам.

- В больших группах увеличиваются издержки на координацию и управление.
- В больших группах увеличивается количество коммуникационных путей. Это создает больше возможностей для недоразумений и увеличивает вероятность ошибок, которые затем приходится исправлять. Лоренс Путнэм заметил, что для кратчайшего из возможных сроков также характерна наибольшая вероятность ошибок (Putnam and Myers 2003).
- При сжатых сроках большую часть работы приходится выполнять параллельно. Чем больше объем перекрывающейся работы, тем выше вероятность того, что одна часть работы будет зависеть от незаконченной или дефектной части другой работы, а последующие изменения увеличат объем необходимой работы по исправлению.

Мнения экспертов по поводу того, насколько сокращение сроков увеличивает объем работ, различаются, но все эксперты сходятся на том, что это явление существует. Соотношение между сроками и объемами работ обсуждается в разделе 20.6.

СОВЕТ № 92

Не сокращайте оценку срока без увеличения оценки объема работ.

Зона невозможности существует, и с этим ничего не сделать. Если 8 человек могут написать программу за 10 месяцев, смогут ли 80 человек написать ту же программу за один месяц? А 1600 людей за один день? Неэффективность крайнего сжатия сроков становится особенно очевидной на крайних случаях — таких, как 1600 людей за один день.

Вычисление пределов менее радикального сжатия сроков является нетривиальной проблемой, но все исследователи сошлись на том, что существует некая «зона невозможности» — точка, за пределами которой сжать номинальный срок уже не удастся. По их общему мнению, сокращение срока более 25 % номинала невозможно.

«Зона невозможности» показана на рис. 20.2. Сокращение срока разработки попросту невозможно. Ни за счет более усердной работы. Ни за счет повышения ее эффективности. Ни за счет поиска более современных решений или увеличения размера группы. Просто невозможно, и все (Symons, 1991, Boehm 2000, Putnam and Myers 2003).

СОВЕТ № 93

Не сокращайте номинальный срок более чем на 25 %. Иначе говоря, не пытайтесь ввести оценки в «зону невозможности».

Увеличение срока по отношению к номиналу обычно приводит к сокращению общего объема работ при сокращении размера группы. Эксперты также

сделали общий вывод, что увеличение срока за пределы номинала способствует уменьшению общего объема работ по тем же причинам, по которым сокращение срока увеличивает объем работ. При увеличении срока появляется возможность использования группы меньшего размера, что снижает проблемы с коммуникациями и координацией работ. Также снижается перекрытие операций, благодаря чему больше дефектов исправляется своевременно, пока они еще не успели проникнуть в другие компоненты и увеличить объем переработки.

Чтобы увеличение срока привело к снижению объема работы, необходимо уменьшить размер группы. Если тот же проект будет поручен тем же участникам и каждому из них достанется часть от прежней работы, скорее всего, вы лишь ухудшите положение (причины объясняются в разделе 3.1).

СОВЕТ № 94

Чтобы снизить стоимость проекта, увеличьте срок и используйте группу меньшей численности.

20.6. Соотношения между сроком и объемом работ

В модель оценки Лоренса Путнэма входят эмпирические правила сжатия и расширения сроков, представленные в табл. 20.4.

Таблица 20.4. Рекомендуемые соотношения между сроком и объемом работ

Увеличение/уменьшение срока	Увеличение/уменьшение объема работ
-15 %	+100 %
-10 %	+50 %
-5 %	+25 %
Номинал	0 %
+10 %	-30 %
+20 %	-50 %
+30 %	-65 %
Более 30 %	Не практично

Источник: По данным «Measures for Excellence» (Putnam and Myers 1992).

Путнэм предупреждает, что при увеличении срока более чем на 30 % начинает снижаться эффективность работы, а это, в свою очередь, приводит к увеличению затрат.

Некоторые специалисты критиковали модель Путнэма за преувеличение последствий увеличения и уменьшения сроков, но данные International Software Benchmarking Standards Group от 2005 года демонстрируют очень похожие результаты (ISBSG 2005).

Сокращение срока и размер группы

При снижении срока ниже номинала возникает еще одна потенциальная проблема: даже если вы не попадете в «зону невозможности», может оказаться, что размер

группы придется увеличить выше экономически эффективного максимума. Лоренс Путнэм провел интереснейшие исследования связей между размером группы, сроками и производительностью для бизнес-систем среднего размера. Полученные им результаты показаны на рис. 20.3 (Putnam and Myers 2003).

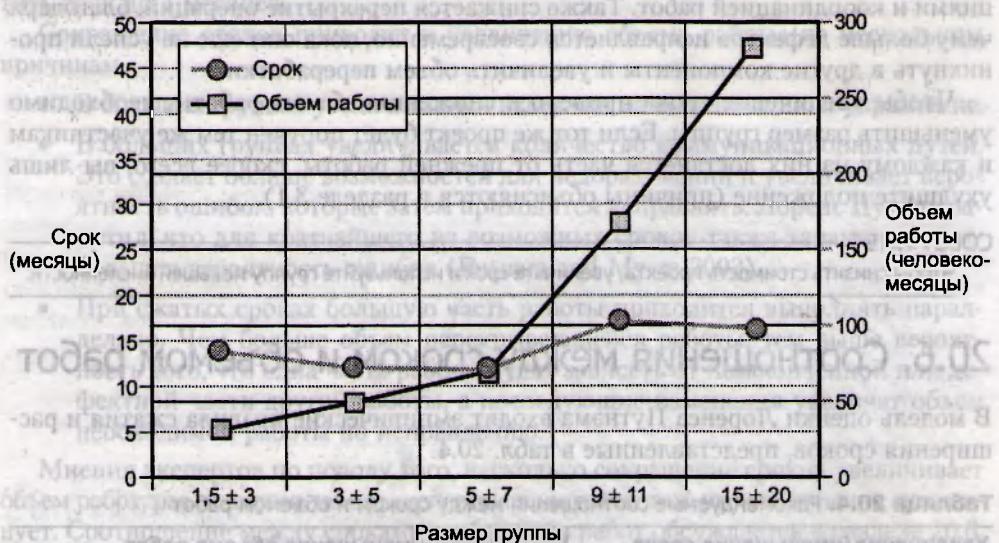


Рис. 20.3. Связь между размером группы, сроком и объемом работ в бизнес-системах, состоящих приблизительно из 57 000 строк кода. В группах, состоящих более чем из 5–7 человек, увеличивается как сроки, так и объем работ

Путнэм проанализировал около 500 бизнес-проектов в диапазоне от 35 000 до 95 000 строк кода (LOC), со средним размером 57 000 строк. Он разделил эти проекты на 5 категорий в зависимости от размера группы разработки. Различия в средних размерах проектов для каждого размера группы лежали в пределах 3000 строк. Путнэм обнаружил, что увеличение размера группы от диапазона 1,5–3 до 3–5 приводило к уменьшению срока и увеличению объема работ, чего и следовало ожидать. При увеличении размера группы с 3–5 до 5–7 срок снова сокращался, а объем работ снова возрастал. Но с увеличением размера группы с 5–7 до 9–11 *увеличивался как срок, так и объем работ*. А при увеличении размера группы с 15 до 20 срок оставался прежним, но объем работ радикально возрастал.

Подозреваю (хотя это всего лишь мое личное мнение), что данные Путнэма доказывают, что издержки масштаба в программных проектах представляют собой не плавную и постепенно возрастающую, а скорее ступенчатую функцию с большими потерями, резко проявляющимися на определенных размерах.

Путнэм еще не обобщил свои результаты для других типов или размеров программных проектов, но в области бизнес-систем среднего размера его результаты весьма важны. Похоже, для таких проектов экономически оптимальной является группа от 5 до 7 человек. В группах большего размера ухудшаются как сроки, так и объемы работ.

СОВЕТ № 95

В бизнес-системах среднего размера (от 35 000 до 100 000 строк кода) не рекомендуется использовать группы численностью более 7 человек.

20.7. Оценка срока и ограничения численности группы

В общем случае средний размер группы вычисляется простым делением оценки объема работ на срок. Если проект в 80 человеко-месяцев оценивается 12-месячным сроком, то средний размер группы равен количеству человеко-месяцев, деленному на срок, или $80/12$, то есть от 6 до 7 участников.

Оценки сроков, представленные в этой главе, выдают номинальный срок для проекта при известном объеме работ. Эти методы предполагают, что независимо от номинального срока вы сможете изменить размер команды до нужного уровня. Если на оцениваемый проект можно будет выделить группу из 6–7 участников, вы сможете обеспечить объем работ в 80 человеко-месяцев при сроке в 12 календарных месяцев.

Но что делать, если вы располагаете всего 4 работниками? А если проект уже достиг той стадии, на которой назначаются индивидуальные задания? А если доступны 10 человек, каждый из которых свободен на 2/3 времени? А если группа уже сформирована и период комплектования штата не потребуется? Все перечисленные факторы не учитываются в формулах этой главы; эти формулы представляют собой методы макрооценки, действующие только на ранних стадиях проектов в диапазоне от среднего до крупного размера.

В средних и крупных проектах численность группы обычно доводится до номинальной от начала до середины проекта, а в отдельных случаях изменения в группе продолжаются до завершающих стадий. В проекте среднего размера могут работать в среднем 15 человек, но начаться проект может с 5 человек, достигнуть 20 на максимуме и закончиться с 10 участниками.

В меньших проектах чаще используется модель «постоянной комплектации» — группа в полном составе начинает работу в день 1, и это продолжается до конца проекта. Если срок оценивается в 12 месяцев, но ваши планы показывают, что при фактической доступности персонала для реализации 80 человеко-месяцев потребуется 15 месяцев, то исходная оценка замещается вашими планами.

Основной целью методов оценки сроков, описанных в этой главе, является не предсказание дня завершения работы, а проверка реалистичности ваших планов в отношении сроков.

После того как вы оцените сроки и убедитесь в разумности своих планов, детальные аспекты планирования (кто и когда доступен) имеют более высокий приоритет по сравнению с исходной оценкой срока.

СОВЕТ № 96

Используйте оценку срока для проверки реалистичности своих планов. Для формирования окончательного графика следует применять детальное планирование.

20.8. Сравнение результатов, полученных разными методами

В этой главе мы использовали пять разных методов оценки срока:

- базовая формула для вычисления срока;
- формула неформальной оценки с прошлыми проектами;
- метод оценки первого порядка;
- оценочные программы, откалиброванные среднеотраслевыми данными;
- оценочные программы, откалиброванные историческими данными.

На рис. 20.4 представлено наглядное сравнение оценок, полученных разными методами.



Рис. 20.4. Диапазоны оценок, которые были получены методами, описанными в этой главе. Относительные размеры точек представляют весовые коэффициенты, присвоенные мной каждой из оценок. Внешний вид оценок (в том числе и недостаточно хорошо обоснованных) маскирует фактическое схождение оценок

На первый взгляд может показаться, что схождение оценки срока в этом примере оставляет желать лучшего. Одно из возможных улучшений предназначено для базовой формулы, по которой строилась верхняя линия. Общий диапазон, показанный на рис. 20.4, предназначен для коэффициентов базовой формулы в диапазоне от 2,0 до 4,0. Анализ исторических данных позволяет оценить действительный диапазон значений коэффициента. В примере, рассмотренном в этой главе, коэффициенты прошлых проектов лежали в диапазоне от 2,7 до 3,7, в результате чего диапазон сроков, получаемых по базовой формуле, сужается до 11,6–14,1 месяца.

Я снова присвоил высокий весовой коэффициент оценке, полученной в Construx Estimate, потому что этот метод относится к категории научных, и что еще важнее — базируется на исторических данных. Оценки базовой формулы и неформального сравнения с прошлыми проектами я бы поставил на второе место, а при наличии более надежных данных метод оценки первого порядка не использовал бы вообще.

На рис. 20.5 показано схождение оценок срока после удаления излишне обобщенных данных.

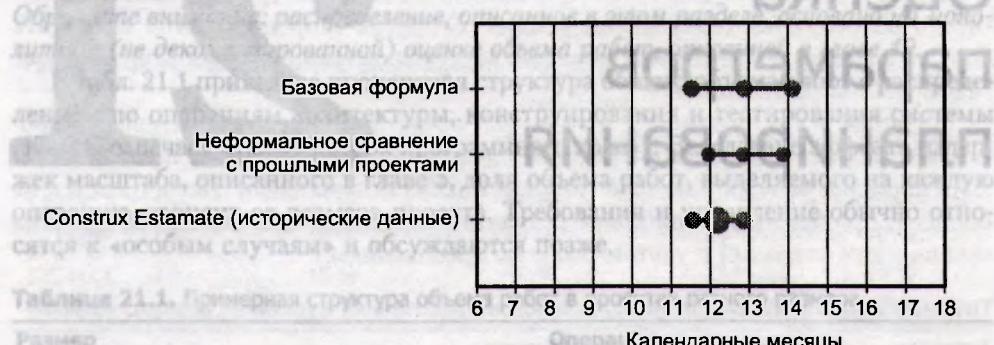


Рис. 20.5. Диапазоны оценок, которые были получены наиболее точными методами. После удаления оценок, полученных излишне обобщенными способами, схождение оценок становится очевидным

Руководствуясь полученными оценками, я бы представил диапазон от 11,5 до 14 месяцев; вероятно, в этом конкретном случае я бы не стал предоставлять оценку ожидаемого случая. Все методы оценки сроков, представленные в этой главе, применяются в широкой части конуса неопределенности, и отсутствие точечной оценки на столь ранней стадии проекта будет приемлемым.

СОВЕТ № 97

Прежде чем искать схождение или расхождение между оценками, исключите из набора данных результаты, полученные слишком общими методами.

Дополнительные ресурсы

Putnam, Lawrence H. and Ware Myers. «Five Core Metrics». New York, NY: Dorset House, 2003. В главе 11 подробно описаны потери производительности, возникающие при разработке бизнес-систем среднего размера в группах, численность которых превышает 7 человек.

Stutzke, Richard D. «Estimating Software-Intensive Systems». Upper Saddle River, NJ: Addison-Wesley, 2005. Стутцке описывает несколько дополнительных методов для оценки сроков; как правило, для них требуются более обширные математические вычисления, чем для методов, представленных в этой главе.

Оценка параметров планирования

21

- метод оценки рискового портфеля
- оценочная программа, отслеживающая среднестатистические данные
- оценочные программы, использующие историческими данными

Граница между оценкой проекта и планированием проекта широка и расплывчата. Многие параметры планирования нуждаются в оценке: какой объем работы следует выделить на конструирование, тестирование, постановку требований и проектирование; сколько тестеров должно приходиться на одного разработчика; сколько человеко-часов может быть выделено тому или иному проекту за календарную неделю или месяц; каким должен быть буфер риска; и множество других показателей, необходимых для целей планирования.

Когда проект выходит на уровень планирования, рассматриваемый в этой главе, цели планирования обычно начинают конфликтовать с целями оценки. Например, после того как вы оцените размер буфера риска, все последующее планирование управления буфером риска будет направлено на снижение его величины (то есть фактически на то, чтобы сделать его оценку недействительной).

Оценку параметров планирования трудно назвать оценкой в чистом виде: взаимосвязь между мелкоструктурными целями и оценками должна быть интенсивной и итеративной. Целью оценки в этом контексте является проверка реалистичности исходных планов. Начиная с этого момента, на первый план выходят планирование и управление проектом, а оценки уступают им.

Короче говоря, планирование решает, *как* вести проект, а оценка — *какую величину* следует заложить в план. Именно этой теме посвящена настоящая глава.

21.1. Оценка операционной структуры проекта

Одним из важных решений из области планирования должно стать распределение объема работ по требованиям, разработке архитектуры, конструированию, тестированию и сопровождению системы. Эти решения должны приниматься как для последовательных, так и для итеративных проектов. Другими словами, вопрос не в том, сколько времени следует отвести на ту или иную фазу, а в том, какой объем работ следует закрепить за операциями при их выполнении.

Оценка объема работ по разным техническим операциям

Обратите внимание: распределение, описанное в этом разделе, основано на монолитной (не декомпозированной) оценке объема работ, описанной в главе 19.

В табл. 21.1 приведена процентная структура общего объема работ с распределением по операциям архитектуры, конструирования и тестирования системы (KLOC означает «тысячи строк программного кода»). Вследствие эффекта издержек масштаба, описанного в главе 5, доля объема работ, выделяемого на каждую операцию, зависит от размера проекта. Требования и управление обычно относятся к «особым случаям» и обсуждаются позже.

Таблица 21.1. Примерная структура объема работ в проектах разного размера

Размер	Операция		
	Архитектура	Конструирование	Тестирование системы
1 KLOC	11 %	70 %	19 %
25 KLOC	16 %	57 %	27 %
125 KLOC	18 %	53 %	29 %
500 KLOC	19 %	44 %	37 %

Источники: Albrecht 1979; Boehm 1981; Glass 1982; Boehm, Gray, and Seewaldt 1984; Boddie 1987; Card 1987; Grady 1987; McGarry, Waligora, and McDermott 1989; Putnam and Myers 1992; Brooks 1995; Jones 1998; Jones 2000; Boehm et al. 2000; Putnam and Myers 2003; Boehm and Turner 2004; Stutzke 2005.

Данные, приведенные в табл. 21.1, являются *приближенными*. Они зависят от применяемых методов, от выбора модели жизненного цикла, от эффективности работы по контролю качества и множества других факторов. В конечном итоге вы должны построить собственную таблицу по историческим данным своей организации. Впрочем, пока это не будет сделано, используйте данные из табл. 21.1 как отправную точку и отрегулируйте оценку для конкретного типа оцениваемого проекта при помощи коэффициентов из табл. 21.5.

Оценка объема работ по требованиям

В табл. 21.1 не включены работы, связанные с требованиями. Если оценка объема работ производилась по среднеотраслевым данным производительности, учтите, что в этих данных операции, связанные с требованиями, обычно не учитываются (впрочем, бывают и исключения; это одна из причин, из-за которых в среднеотраслевых данных встречается такой разброс).

Оценка, созданная вами по собственным историческим данным, может включать или не включать данные требований в зависимости от того, учитываются ли они в использованных исторических данных.

Модели оценки, в том числе Cocomo II и модель Путнэма, предполагают, что полученные с их помощью монолитные оценки не включают работу по требованиям. Одна из причин заключается в том, что доля работы по требованиям изменяется

в большей степени, чем доля других операций. Вы можете бегло пройтись по требованиям и очень приблизительно определить большой набор, для реализации которого потребуются геркулесовы усилия. С другой стороны, можно потратить больше времени и определить меньший набор качественных требований с более эффективной реализацией.

С учетом всего сказанного, в табл. 21.2 представлены приблизительные проценты от общего объема работ, которые следует отводить на работу по требованиям в проектах разных размеров. Базовая оценка объема работ увеличивается на процент, приведенный в таблице; результат определяет суммарный объем всех технических операций с учетом работ по требованиям.

Таблица 21.2. Приблизительная доля работ по требованиям в проектах разных размеров

Размер	Приращение базовой оценки
1 KLOC	5 %
25 KLOC	5 %
125 KLOC	8 %
500 KLOC	10 %

Источники: те же, что для табл. 21.1.

Оценка объема работ по управлению

Как и в случае с требованиями, монолитная оценка объема работ не учитывает работы, относящиеся к управлению, если только они не учитываются в используемых вами исторических данных. В табл. 21.3 представлены приблизительные проценты от общего объема работ, которые следует отводить на управляющие операции в проектах разных размеров. Базовая оценка объема работ также увеличивается на процент, приведенный в таблице; результат определяет суммарный объем всех технических операций с учетом управления.

Таблица 21.3. Приблизительная доля управляющих операций в проектах разных размеров

Размер	Приращение базовой оценки из табл. 21.1 (без учета работы по требованиям)
1 KLOC	10 %
25 KLOC	12 %
125 KLOC	14 %
500 KLOC	17 %

Источники: те же, что для табл. 21.1.

Оценка общего объема работ

Для простоты вычислений в табл. 21.4 приведены распределения работ по требованиям, архитектуре, конструированию, тестированию системы и управлению для проектов разных размеров.

Таблица полезна в том случае, когда данные, которые использовались для калибровки вашей монолитной оценки объема работ, включали работы по требованиям и управлению.

Таблица 21.4. Распределение объема работ в проектах разного размера

Операция					
Требования	Архитектура и планирование	Конструирование	Тестирование системы	Управление	
1 KLOC	4 %	10 %	61 %	16 %	9 %
25 KLOC	4 %	14 %	49 %	23 %	10 %
125 KLOC	7 %	15 %	44 %	23 %	11 %
500 KLOC	8 %	15 %	35 %	29 %	13 %

Источники: те же, что для табл. 21.1.

Поправки для типов проектов

Как было указано в главе 5, объем работы зависит от типа проекта. Кроме того, тип влияет на процентное распределение объемов работ по разным типам операций. В табл. 21.5 перечислены поправки, которые необходимо внести в номинальное распределение в зависимости от типа проекта.

Таблица 21.5. Приблизительные поправки для распределения операций в зависимости от типа проекта

Операция	Бизнес-системы, внутренние интрасетевые системы	Встроенные системы, телекоммуникации, драйверы устройств, системные программы	Коммерческие программы, научные и инженерные пакеты, публичные интернет-системы
Требования	-3 %	+20 %	-20 %
Архитектура	-7 %	+10 %	-5 %
Конструирование	+5 %	-10 %	+2 %
Тестирование системы	-7 %	+6 %	+9 %
Управление	+3 %	+3 %	-15 %

Источники: Putnam and Myers 1992; Jones 1998; Jones 2000; Boehm et al, 2000; Putnam and Myers 2003; Boehm and Turner 2004; Stutzke 2005.

СОВЕТ № 98

При распределении объема работы проекта следует учитывать размер проекта, тип проекта, а также операции, заложенные в калибровочных данных, использованных для создания исходной монолитной оценки.

Пример распределения объема работы

Предположим, вы разрабатываете бизнес-систему, которая, согласно вашей оценке, будет состоять из 80 000 строк кода (1450 функциональных пунктов) и займет около 80 человеко-месяцев. В основном распределении из табл. 21.1 представлены

проценты для проектов в 25 KLOC и 125 KLOC. Проект размером 80 KLOC находится примерно посередине между этими двумя размерами, поэтому мы будем использовать средние арифметические процентов для 25 и 125 KLOC. В соответствии с полученными данными, 17 % объема выделяется на архитектуру (13,6 человека-месяца), 55 % — на конструирование (44,0 человека-месяца), а 28 % — на тестирование системы (22,4 человека-месяца). Таблица 21.2 рекомендует увеличить работу по требованиям на 6,5 % (5,2 человека-месяца), а в соответствии с табл. 21.3, работа по управлению проектом возрастает на 13 % (10,4 человека-месяца). Затем в табл. 21.6 базовое распределение объема работ умножается на поправочные коэффициенты для проектов бизнес-систем.

Таблица 21.6. Применение поправочных коэффициентов к номинальному распределению объема работ в зависимости от типа проекта

Операция	Номинальное распределение (человеко-месяцы)	Поправка для бизнес-систем	Итоговое распределение (в человеко-месяцах)	Итоговое распределение (в процентах)
Требования	5,2	-3 %	5,0	4 %
Архитектура	13,6	-7 %	12,6	13 %
Конструирование	44,0	+5 %	46,6	51 %
Тестирование системы	22,4	-7 %	20,8	22 %
Управление	10,4	+3 %	10,7	10 %
ИТОГО	95,6	—	95,7	100 %

В данном примере номинальная и итоговая оценки объема работы составляют 95,6 и 95,7 человека-месяца соответственно. Небольшое несовпадение результатов обусловлено ошибками округления.

Следует хорошо понимать, что это распределение работ по фазам является приближенным, а проценты лишь представляют отправные точки для планирования. После того как оценка выведет вас на правильный путь планирования, исходные оценки уступают детальному планированию.

Соотношения между численностью разработчиков и тестеров

Один из самых частых вопросов из области планирования: «Сколько разработчиков должно приходиться на одного тестера?» Некоторые типичные соотношения перечислены в табл. 21.7.

Данные в таблице были получены организациями, с которыми моя компания и я сотрудничали последние 10 лет.

Как видно из таблицы, соотношения очень сильно изменяются даже в пределах одного вида программных проектов. Это вполне объяснимо, потому что соотношение, хорошо подходящее для конкретной компании или проекта, зависит от стиля разработки, сложности тестируемой программы, соотношения

объемов старого и нового кода, квалификации тестеров относительно квалификации разработчиков, степени автоматизации тестирования и множества других факторов.

Таблица 21.7. Примеры соотношений между численностью разработчиков и тестеров

Среда	Типичные наблюдаемые соотношения
Типичные бизнес-системы (внутренние интрасети, информационно-управляющие системы и т. д.)	3:1 — 20:1 (часто специалистов по тестированию вообще нет)
Типичные коммерческие системы (открытые интернет-системы, коммерческие программные продукты и т. д.)	1:1 — 5:1
Научные и инженерные проекты	5:1 — 20:1 (часто специалистов по тестированию вообще нет)
Типичные системные проекты	1:1 — 5:1
Системы, критические по безопасности	5:1 — 1:2
Microsoft Windows 2000	1:2
Программное обеспечение космического шаттла NASA	1:10

В конечном итоге соотношение численности разработчиков и тестеров определяется скорее планированием, а не оценкой, то есть тем, что вы *намерены* сделать, а не прогнозами того, что нужно сделать.

21.2. Оценка срока для разных операций

Распределение календарного времени по разным операциям и фазам проекта также в большей степени определяется факторами планирования, а не оценками. В табл. 21.8 представлено примерное распределение сроков по основным техническим операциям для разных размеров проектов. Конечно, было бы удобнее, если бы числа в таблице не были выражены в виде диапазонов. График выполнения этих операций зависит от того, когда освободятся те или иные работники, в какой степени им приходится совмещать работу над оцениваемым проектом с другими задачами, и от других факторов. В этом отношении распределение сроков подвержено большей изменчивости, чем распределение объема работ.

Таблица 21.8. Примерное распределение сроков в проектах разного размера

Размер	Операция		
	Архитектура	Конструирование	Тестирование системы
1 KLOC	15–25 %	50–65 %	15–20 %
25 KLOC	15–30 %	50–60 %	20–25 %
125 KLOC	20–35 %	45–55 %	20–30 %
500 KLOC	20–40 %	40–55 %	20–35 %

Источники: Boehm 1981; Putnam and Myers 1992; Boehm et al., 2000; Putnam and Myers 2003; Stutzke 2005.

Как и в случае с объемом работ, сроки для работ по требованиям обычно оцениваются в процентах от базовой оценки срока. В табл. 21.9 представлены проценты, прибавляемые к базовому сроку для работ по требованиям.

Таблица 21.9. Приблизительное увеличение срока для работ по требованиям в проектах разных размеров

Размер	Приращение базовой оценки
1 KLOC	10–16 %
25 KLOC	12–20 %
125 KLOC	13–22 %
500 KLOC	24–30 %

Источники: Boehm 1981; Putnam and Myers 1992; Boehm et al., 2000; Putnam and Myers 2003; Stutzke 2005.

В высокоитеративных проектах распределение сроков производится при каждой итерации. Если же проект относится к последовательному типу, сроки распределяются для фаз всего проекта.

СОВЕТ № 99

При распределении сроков между различными операциями следует учитывать размер проекта, его тип и методологию разработки.

Как и в случае с распределением объема работ по операциям, распределение сроков проще всего выполняется при наличии исторических данных.

21.3. Преобразование оценки в планируемый объем работы

Оценки объема работы обычно выражаются в «человеко-месяцах», «человеко-днях» или других сходных показателях. Такие показатели выражают *идеальный* объем работы, когда каждый месяц работы соответствует одному календарному месяцу.

Майк Кон (Mike Cohn) описывает различия между идеальным и планируемым временем, сравнивая их с игровым и реальным временем в американском футболе (Cohn 2006). Нормальная игра в американском футболе продолжается 60 минут игрового времени. По настенным часам она может занять от 2 до 4 часов.

Аналогично, планировщик программного проекта не должен предполагать, что один человек способен выполнить объем работы в один человеко-месяц за один календарный месяц. Его «рабочий месяц» разбавляется отпусками, выходными и обучением; он может одновременно работать на несколько проектов; на конец, возможны и другие причины.

Рассматривая преобразование идеального объема работы в планируемый, необходимо учитывать следующие факторы.

- Какое время заложено в калибровочные данные, использованные при создании оценки объема работ? Включены ли в него работы, связанные

с управлением, требованиями и тестированием или только непосредственная разработка? Учтены ли сверхурочные? Предположения, включенные в калибровочные данные, автоматически переходят в оценку работы.

- В каком количестве проектов будут одновременно работать программисты? Если программист делится между двумя проектами, то для выдачи одного месяца «целенаправленной» работы ему потребуется два и более месяца.
- Учтены ли в калибровочных данных выходные, отпуска, болезни, время обучения, поддержка торговых выставок, работа с клиентами, поддержка эксплуатируемых систем и т. д.? Если нет, придется включить эти «отвлекающие факторы» при преобразовании оцениваемого объема работы в планируемый.

Перечисленные факторы сильно зависят от организации. Если вы работаете в организации, в которой группа может сосредоточиться на одном проекте, в предположения можно заложить от 40 до 50 часов «целенаправленного» рабочего времени в неделю. Я видел некоторые компании, в которых эта цель была достигнута. Обычно в таких компаниях внутренняя мотивация участников чрезвычайно сильна; группы невелики; участники групп молоды и не перегружены семейными обязательствами; компания предлагает значительные финансовые стимулы, а рабочая среда не отягощена бюрократизмом и корпоративными мероприятиями.

Если вы работаете в большой, устоявшейся организации, в которой часто происходят корпоративные собрания, а большинство людей работает по 40 часов в неделю, вероятно, только от 20 до 30 часов будет направлено на работу над проектом, причем эти часы могут делиться между двумя и более проектами.

Информация о том, сколько в среднем времени в день работник может посвятить конкретному проекту, различается. Кейперс Джонс сообщает, что в среднем технические работники в день выделяют около 6 часов на целенаправленную работу над своими проектами, или 132 часа в месяц (Jones 1998). Модель Сосомо II предполагает 152 часа целенаправленной работы в месяц (Boehm et al. 2000). Конкретное количество часов существенно зависит от специфики организации; как и в предыдущих случаях, старайтесь получить собственные данные, основанные на прошлых проектах вашей организации.

За ссылками на дополнительную информацию по проблемам планирования обращайтесь к разделу «Дополнительные ресурсы» в конце главы.

21.4. Оценка стоимости

Теоретически оценка стоимости представляет собой тривиальную функцию от объема работы. Тем не менее получение оценки стоимости проекта усложняется целым рядом факторов.

Сверхурочные работы

Допускает ли ваша организация некомпенсируемые сверхурочные работы? Если допускает, то некоторый процент оцениваемого объема работ может не учитываться в оценке стоимости. Использует ли ваша организация временных работников

или подрядчиков, получающих более высокую оплату за переработку? В таких случаях вклад части оцениваемого объема работы в формирование стоимости может оказаться выше среднего.

Выбор типа затрат

В некоторых организациях стоимость проекта вычисляется на базе «прямых затрат», то есть затрат, напрямую приписываемых конкретному работнику (зарплата, налоги, премии и т. д.). В других организациях при расчете стоимости проекта используются «обременяющие затраты» с учетом полных корпоративных затрат, которые не ассоциируются с конкретными работниками (аренда, корпоративные налоги, затраты служб кадров, продаж, маркетинга и т. д.). В зависимости от размера организации, объема невозмещаемой инфраструктуры, стоимости офисного пространства и других факторов, бремя расходов в процентах от зарплаты работника может составлять от 30 до 125 % и выше.

Другие прямые затраты

Некоторые проекты также требуют затрат на командировки, специализированный инструментарий, оборудование и т. д. Эти затраты тоже должны включаться в оценку стоимости.

За ссылками на дополнительную информацию по этой теме обращайтесь к разделу «Дополнительные ресурсы» в конце главы.

21.5. Оценка дефектообразования и исправления

Дефектообразование в программном проекте является функцией объема работы и размера проекта, поэтому количество дефектов тоже может оцениваться заранее. Информация о вероятном количестве дефектов пригодится для планирования объема работ, направленных на их исправление.

Кейперс Джонс описывает один из способов анализа дефектообразования, основанный на размере программы в функциональных пунктах (Jones 2000). Как показано в табл. 21.10, из данных Джонса следует, что в типичном проекте на один функциональный пункт создается около 5 дефектов. Этот показатель соответствует примерно 50 дефектам на 1000 строк программного кода (в зависимости от используемого языка программирования).

Таблица 21.10. Типичная частота дефектообразования в различных операциях

Операция	Среднее количество создаваемых дефектов
Требования	Один дефект на функциональный пункт
Архитектура	1,25 дефекта на функциональный пункт
Конструирование	1,75 дефекта на функциональный пункт
Документация	0,60 дефекта на функциональный пункт
Некорректные исправления	0,40 дефекта на функциональный пункт
ИТОГО	5,0 дефекта на функциональный пункт

Другой фактор, вносящий свой вклад в издержки масштаба, заключается в том, что в крупных проектах на строку кода обычно генерируется больше дефектов; это повышает объем работы по исправлению ошибок, что, в свою очередь, повышает затраты проекта. В табл. 21.11 представлена зависимость плотности ошибок для разных размеров проекта.

Таблица 21.11. Размер проекта и плотность ошибок

Размер проекта (в строках кода)	Типичная плотность ошибок
< 2K	0–25 ошибок на KLOC
2K–16K	0–40 ошибок на KLOC
16K–64K	0,5–50 ошибок на KLOC
64K–512K	2–70 ошибок на KLOC
>512K	4–100 ошибок на KLOC

Источник: «*Program Quality and Programmer Productivity*» (Jones 1977), «*Estimating Software Costs*» (Jones 1998).

Среднеотраслевые частоты дефектообразования изменяются более чем на порядок. Исторические данные прошлых проектов повысят точность оценки.

СОВЕТ № 100

Используйте среднеотраслевые или исторические данные для оценки предполагаемого количества дефектов в вашем проекте.

Оценка работы по исправлению дефектов

Дефектообразование — лишь одна из частей формулы планирования. Другой частью является исправление дефектов. В программной отрасли накопился достаточно большой объем данных по эффективности основных методов исправления дефектов. В табл. 21.12 представлены данные об эффективности исправления дефектов¹ для анализа, обсуждения, модульного тестирования, тестирования системы и других распространенных методов.

Таблица 21.12. Эффективность исправления дефектов

Фаза удаления	Минимальная эффективность	Модальная эффективность	Максимальная эффективность
Неформальный обзор архитектуры	25 %	35 %	40 %
Формальный анализ архитектуры	45 %	55 %	65 %
Неформальный обзор кода	20 %	25 %	35 %
Формальный анализ кода	45 %	60 %	70 %
Моделирование или макетирование	35 %	65 %	80 %

продолжение ↗

¹ Defect Removal Rate (Efficiency) = (Количество исправленных дефектов)/(Количество исправленных дефектов + Количество дефектов, обнаруженных после начала поставок продукта). — Примеч. перев.

Таблица 21.12 (продолжение)

Фаза удаления	Минимальная эффективность	Модальная эффективность	Максимальная эффективность
Персональная проверка кода	20 %	40 %	60 %
Модульное тестирование	15 %	30 %	50 %
Тестирование новых функций (компонентов)	20 %	30 %	35 %
Интеграционное тестирование	25 %	35 %	40 %
Регрессионное тестирование	15 %	25 %	30 %
Системное тестирование	25 %	40 %	55 %
Бета-тестирование в малом масштабе (< 10 мест)	25 %	35 %	40 %
Бета-тестирование в большом масштабе (> 1000 мест)	60 %	75 %	85 %

Источник: По материалам «Programming Productivity» (Jones 1986a), «Software Defect-Removal Efficiency» (Jones 1996) и «What We Have Learned About Fighting Defects» (Shull et al 2002).

Важную роль здесь играет диапазон от минимальной до максимальной эффективности; как обычно, исторические данные вашей организации помогут получить более точную оценку.

Пример оценки эффективности исправления дефектов

Объединение информации из таблиц дефектообразования и исправления позволяет оценить количество дефектов, которые останутся в окончательной версии вашей программы (и конечно, помогает спланировать действия по удалению большего или меньшего количества дефектов, в зависимости от ваших критериев качества).

Предположим, имеется система в 1000 функциональных пунктов. Оценка по данным Джонса из табл. 21.10 показывает, что проект будет содержать около 5000 дефектов. В табл. 21.13 описаны результаты их поэтапного устранения с применением стандартной стратегии, состоящей из персональной проверки кода, модульного тестирования, интеграционного тестирования, системного тестирования и бета-тестирования в малом масштабе.

Таблица 21.13. Пример типичной процедуры образования и устранения дефектов (для системы размером в 1000 функциональных пунктов)

Операция	Изменения в количестве дефектов	Общее количество внесенных дефектов	Остается дефектов
Требования	+1000 дефектов	1000	1000
Архитектура	+1250 дефектов	2250	2250
Конструирование	+1750 дефектов	4000	4000
Персональная проверка кода	-40 %	4000	2400
Документация	+600 дефектов	4600	3000
Модульное тестирование	-30 %	4600	2100

Операция	Изменения в количестве дефектов	Общее количество внесенных дефектов	Остается дефектов
Интеграционное тестирование	-35 %	4600	2100
Системное тестирование	-40 %	4600	1365
Некорректные исправления	+400 дефектов	5000	1219
Бета-тестирование в малом масштабе	-35 %	5000	792
Дефектов остается в окончательной версии	-84 %	5000	72 (16 %)

Получается, что типичный подход к исправлению дефектов устранит из программы только около 84 % ошибок, что примерно соответствует стандартам отрасли (Jones 2000). Как обычно, числа, полученные этим способом, являются приближенными.

В табл. 21.14 показано, как может проходить исправление дефектов в лучших организациях. Предполагается, что группа внесет в проект те же 5000 дефектов. Однако на этот раз в число методов удаления дефектов войдет моделирование требований, формальный анализ архитектуры, персональная проверка кода, модульное, интеграционное и системное тестирование, а также бета-тестирование в большом масштабе. Как видно из таблицы, комбинация этих методов должна привести к удалению из программного продукта около 95 % дефектов.

Таблица 21.14. Пример самой надежной процедуры образования и устранения дефектов (для системы размером в 1000 функциональных пунктов)

Операция	Изменения в количестве дефектов	Общее количество внесенных дефектов	Остается дефектов
Требования	+ 1000 дефектов	1000	1000
Моделирование требований	- 65 %	1000	350
Архитектура	+ 1250 дефектов	2250	1600
Формальный анализ архитектуры	- 55 %	2250	720
Конструирование	+ 1750 дефектов	4000	2470
Документация	+ 600 дефектов	4600	3070
Персональная проверка кода	- 40 %	4600	1842
Модульное тестирование	- 30 %	4600	1289
Интеграционное тестирование	- 35 %	4600	838
Системное тестирование	- 40 %	4600	503
Некорректные исправления	+ 400 дефектов	5000	903
Бета-тестирование в малом масштабе	- 735 %	5000	226
Дефектов остается в окончательной версии	- 95 %	5000	226 (5 %)

Как и в предыдущем примере, излишняя четкость оценки в 226 дефектов не поддерживается используемыми данными.

СОВЕТ № 101

Данные эффективности исправления дефектов позволяют оценить количество дефектов, которые будут удалены из программы вашими методами контроля качества перед выпуском окончательной версии.

Лоренс Путнэм приводит два дополнительных эмпирических правила исправления дефектов. Если вы хотите перейти от 95 % надежности к 99 % надежности, долю «основной сборки» в сроке следует увеличить на 25 %. Чтобы перейти от 99 к 99,9 % надежности, заложите в срок еще 25 % (Putnam and Myers 2003). (В терминологии Путнэма понятия «надежность» и «процент исправления дефектов перед выпуском» эквивалентны.)

Дальнейшая оценка атрибутов качества — весьма непростая тема, в значительной мере опирающаяся на научные методы оценки. В секции «Дополнительные ресурсы» в конце главы рассказано, где найти дополнительную информацию.

21.6. Оценка риска и резервные буферы

На интуитивном уровне все мы понимаем, что в проектах с повышенным риском следует предусмотреть увеличенные буферы для непредвиденных обстоятельств, а в проектах с малым риском можно обойтись небольшим буфером. Но каким именно должен быть размер буфера?

При анализе рисков обычно учитываются их возможные последствия и вероятности. В табл. 21.15 показан пример таблицы рисков с указанием вероятности, влияния риска и причиняемого ущерба.

Таблица 21.15. Пример таблицы рисков для сроков проекта

Риск	Вероятность	Влияние (срок)	Ущерб (срок)
1	5 %	15 недель	0,75 недели
2	25 %	2 недели	0,5 недели
3	25 %	8 недель	2 недели
4	50 %	2 недели	1 неделя
ИТОГО			4,25 недели

Влияние риска, умноженное на его вероятность, обычно называется ущербом (Risk Exposure, RE). Со статистической точки зрения ущерб представляет собой «ожидаемое значение» риска, или ту величину, на которую, как предполагается, увеличится срок из-за наличия риска. Для рисков, перечисленных в табл. 21.15, базовый срок проекта увеличивается на 4,25 недели. С вероятностью 50 % срок возрастет на большую величину и с вероятностью 50 % — на меньшую.

Суммарный ущерб является хорошей отправной точкой для количественного буферного планирования. Если вам нужна более твердая уверенность в том, что проект будет выдан вовремя, запланируйте буфер, размер которого превышает

величину суммарного ущерба. Если повышенный риск нарушения сроков вас не пугает, планируйте буфер меньшего размера.

Впрочем, ущерб — это лишь часть общей картины. Если в табл. 21.15 риск 1 или 3 реализуется, проект выйдет за пределы своего ожидаемого буфера в 4,25 недели. Эти события маловероятны, однако вы должны проанализировать последствия конкретных рисков перед тем, как остановиться на итоговом размере буфера.

Риски в табл. 21.15 представлены только с точки зрения возможного нарушения срока. Любой риск также может создавать угрозу для объема работ, стоимости, функциональности, качества или прибыли. В табл. 21.16 показан пример таблицы рисков, включающей сроки, стоимость и прибыль.

Таблица 21.16. Пример расширенной таблицы рисков

Риск	Вероятность	Влияние (срок)	Ущерб (срок)	Влияние (стоимость)	Ущерб (стоимость)	Влияние (прибыль)	Ущерб (прибыль)
1	5 %	15 недель	0,75 недели	\$150 000	\$7500	\$10 000 000	\$500 00
2	25 %	2 недели	0,5 недели	\$20 000	\$5000	\$0	\$0
3	25 %	8 недель	2 недели	\$80 000	\$20 000	\$500 000	\$125 000
4	50 %	2 недели	1 неделя	\$20 000	\$10 000	\$0	\$0
ИТОГО		—	4,25 недели	—	\$42 000	—	\$625 000

При планировании необходимо выделить отдельные буфера для срока, объема работ, стоимости, функциональности и качества. Эти буфера лишь отдаленно связаны друг с другом.

Помните, что влияние и вероятности рисков — величины оцениваемые, а точность сводного ущерба не превышает точности данных, которые использовались при ее вычислении.

СОВЕТ № 102

Используйте суммарный ущерб рисков вашего проекта в качестве отправной точки при планировании буферов. Проанализируйте отдельные аспекты конкретных рисков и постарайтесь понять, не должен ли запланированный буфер быть больше или меньше суммарного ущерба.

Управление рисками — довольно сложная область, в которой научные методы оценки могут сыграть важную роль. В секции «Дополнительные ресурсы» в конце главы рассказано, где найти дополнительную информацию.

21.7. Другие эмпирические правила

Приведу еще несколько эмпирических правил, которые могут использоваться для других аспектов планирования.

- Административная и бюрократическая поддержка увеличивают базовую оценку объема работ на 5–10 % (Stutzke 2005).
- Техническая поддержка (настройка лабораторного оборудования, установка новых программ и т. д.) увеличивает базовую оценку объема работ на 2–4 % (Stutzke 2005).

- Поддержка управления конфигурациями/сборкой увеличивает базовую оценку объема работ на 2–8 % (Stutzke 2005).
- Резервируйте от 1 до 4 % в месяц на расширение требований (Jones 1998).
- Переход от разработки, производимой одной компанией на одной площадке, к совместным распределенным разработкам увеличивает объем работ на 25 % (Boehm et al 2000).
- Переход от разработки, производимой одной компанией на одной площадке, к международной удаленной (outsourcing) разработке увеличивает объем работ на 40 % (Boehm et al 2000).
- Если разработчики впервые имеют дело с новым языком программирования и инструментарием, объем работ увеличивается на 20–40 % по сравнению со знакомым языком и инструментарием (Boehm et al 2000).
- Если разработка впервые производится в новой среде, объем работ увеличивается на 20–40 % по сравнению с разработкой в знакомой среде (Boehm et al 2000).

Дополнительные ресурсы

Boehm, Barry W. «Software Engineering Economics». Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981. Хотя это издание отчасти устарело с выходом «Software Cost Estimation with Cocomo II» (см. ниже), в нем содержатся интересные, подробные таблицы объемов работ и распределения сроков по различным операциям.

Boehm, Barry et al. «Software Cost Estimation with Cocomo II». Reading, MA: Addison-Wesley, 2000. В приложении А книги Бема описано распределение объема работ и сроков для каскадных проектов, а также проектов на базе MBASE и Rational Unified Process. В таблице A.10 (которая в действительности состоит из шести таблиц) приводится подробное распределение сроков и объемов по различным операциям.

Cohn, Mike. «Agile Estimating and Planning», Englewood Cliffs, NJ: Prentice Hall PTR, 2006. В главе 5 книги Коня хорошо объясняются различия между идеальным и плановым объемом работ.

DeMarco, Tom, and Timothy Lister. «Waltzing with Bears: Managing Risks on Software Projects», New York, NY: Dorset House, 2003. Доступное введение в управление рисками в программных проектах.

Fenton, Norman E., and Shari Lawrence Pfleeger. «Software Metrics: A Rigorous and Practical Approach», Boston, MA: PWS Publishing Company, 1997. В главе 10 подробно обсуждается оценка надежности программного продукта. Если вы не любите уравнения с греческими символами и математическими знаками, эта книга не для вас.

Jones, Capers. «Estimating Software Costs». New York, NY: McGraw-Hill, 1998. Глава 14 книги Джонса содержит подробное, снабженное примерами описание структуры расходов в разных типах организаций. Глава 21 объясняет, как неоплаченные сверхурочные работы отражаются на оценке стоимости.

Jones, Capers. «Software Assessments, Benchmarks, and Best Practices». Reading, MA: Addison-Wesley, 2000. Некоторые данные, представленные в этой книге, обновлены или расширены по сравнению с теми данные, которые были представлены в «Estimating Software Costs».

Putnam, Lawrence H. and Ware Myers. «Measures for Excellence: Reliable Software On Time, Within Budget». Englewood Cliffs, NJ: Yourdon Press, 1992. Путнэм и Майерс приводят ряд полезных эмпирических правил. Общий контекст книги — подробное математическое объяснение модели оценки Путнэма.

Stutzke, Richard D. «Estimating Software-Intensive Systems». Upper Saddle River, NJ: Addison-Wesley, 2005. В главе 12 описаны методы распределения объема работ, основанные на моделях Cocomo 81 и Cocomo II. Главы 15 и 23 фокусируются на подробной оценке стоимости. Основной темой книги Стузке является оценка стоимости проекта и другие вопросы, относящиеся к затратам, и в ней встречаются многочисленные полезные советы по этой теме. В разделах 12.1 и 12.2 обсуждаются связи между объемом работ, продолжительностью и доступностью персонала.

Tockey, Steve. «Return on Software». Boston, MA: Addison-Wesley, 2005. В главе 15 книги Токи хорошо описана методика определения стоимости, включая распределение непроизводительных затрат разными методами калькуляции и потенциальные опасности, связанные с некоторыми методами.

СОВЕТ № 103

Планирование и оценка — взаимосвязанные темы, а тема планирования слишком обширна, чтобы ее можно было изложить в одной главе книги, посвященной оценке программных проектов. Читайте специализированную литературу по планированию.

4. Бизнес-предметы не являются объектом оценки.
5. Степень необходимости интеграции с системой Foodat неизвестна. Для оценки требуется уточнение. Если потребуется базовый объем работы, оценки для всего проекта также должны быть уточнены.
6. Количество этих ошибок неизвестно и необходимо учесть, потому что большая часть проекта выполняется в лестничном порядке.
7. После наступления кризиса в 2008 году оценки должны меняться в соответствии с изменениями в бизнес-процессах.
8. На краине мере 80 % кода базы данных версии 2.0 удастся использовать без значительных модификаций.
9. После наступления кризиса в 2008 году оценки должны меняться в соответствии с изменениями в бизнес-процессах.
10. Этот подход также может применяться в ситуациях, когда вас заставляют сформировать оценку на базе предположений, используя формулы и стандартные статистическими (таких, как предположения 7–9). Вы беретесь за дело и составляете оценку, но документируете заложенные в нее предположения. Если работы над проектом пойдет так, что оценка окажется недостоверной, вы всегда сможете сослаться на предположения как на основу для формирования оценки в будущем.

Стили представления оценок

22

Область применения методов этой главы

Выбор стиля представления в соответствии с точностью оценки

Что оценивается	Размер, объем работы, срок, функциональность
Размер проекта	М С Б
Стадия разработки	Ранняя–поздняя
Итеративный или последовательный стиль	Оба
Возможная точность	—

Стиль представления оценки создает определенное впечатление относительно ее точности. Если стиль представления подразумевает необоснованную точность, вы сами закладываете предпосылки для непростого обсуждения самой оценки. В этой главе представлено несколько способов представления оценок.

22.1. Представление предположений, заложенных в оценку

Важнейшей практикой представления оценки является документирование предположений, заложенных в этой оценке. Предположения делятся на несколько категорий:

- обязательные функции;
- необязательные функции;
- глубина проработки тех или иных функций;
- доступность ключевых ресурсов;

- зависимость от третьих сторон;
- основные неизвестные факторы;
- основные факторы влияния и чувствительность оценки;
- качество оценки;
- предполагаемое использование оценки.

Ниже показан пример оценки, представленной с документированными предположениями. Документирование предположений помогает сформировать впечатление об изменчивости, заложенной в программном проекте.

Пример документирования предположений оценки

Оценка проекта

Базовая оценка срока составляет 6 календарных месяцев; по нашему мнению, ее точность лежит в пределах 25 %. Эта оценка может быть взята за основу формирования бюджета, но не может использоваться для принятия внешних обязательств. Оценка базируется на следующих предположениях.

1. Три ключевых технических руководителя приступят к работе исключительно над проектом с 15 марта.
2. Все разработчики и специалисты по тестированию приступят к работе исключительно над проектом с 15 апреля.
3. Подсистема форматирования графики будет получена от субподрядчика с приемлемым качеством к 1 августа.
4. Бизнес-правила не потребуют обновления.
5. Степень необходимой интеграции с системой FooBar неизвестна. В данную оценку заложено 250 человеко-часов работы по интеграции. Если потребуется больший объем работы, оценку для всего проекта также придется пересмотреть.
6. В проекте потребуется не более 5 новых отчетов.
7. Новый инструментарий разработки обеспечит прирост производительности на 20 % и более по сравнению с прошлыми проектами.
8. Количество дней отпусков по болезни будет меньше среднего, потому что большая часть проекта выполняется в летнее время.
9. После наступления дат, указанных в пунктах 1 и 2, персонал не будет отвлекаться на поддержку предыдущих версий программы.
10. По крайней мере 80 % кода базы данных из версии 2.0 удастся использовать повторно без модификаций.

В случае нарушения этих предположений оценку придется пересмотреть.

Этот подход также может пригодиться в ситуациях, когда вас заставляют сформировать оценку на базе предположений, которые вам кажутся нереалистичными (таких, как предположения 7–9). Вы беретесь за дело и составляете оценку, но документируете заложенные в нее предположения. Если работа над проектом пойдет так, что оценка станет недействительной, вы всегда сможете ссытаться на предположения как на основу для пересмотра оценки.

СОВЕТ № 104

Документируйте и сообщайте предположения, заложенные в оценке.

22.2. Выражение неопределенности

Ключевой проблемой в представлении оценки является документирование ее неопределенности способом, который бы ясно выражал неопределенность и повышал до максимума вероятность конструктивного, корректного использования оценки. В этом разделе описано несколько способов передачи неопределенности в оценках.

Квалификаторы «плюс/минус»

Оценка с квалификатором «плюс/минус» имеет вид «6 месяцев \pm 2 месяца» или «\$600 000 \pm \$200 000 – \$100 000». В подобных обозначениях указывается как величина, так и направление неопределенности. Формулировка «6 месяцев + 1/2 месяца – 1/2 месяца» говорит о том, что оценка весьма точна, и существует достаточно высокая вероятность ее достижения. Формулировка «6 месяцев +4 месяца – 1 месяц» означает, что точность оценки оставляет желать лучшего, а шансы на ее реализацию невелики.

При выражении оценки с квалификатором «плюс/минус» следует учитывать величину квалификаторов и их смысл. На практике квалификаторы часто выбираются достаточно большими для того, чтобы они включали одно стандартное отклонение с каждой стороны базовой оценки. При таком подходе все равно остается 16 % вероятность того, что фактический результат превысит верхнюю границу оценки, и 16 % вероятность того, что он окажется под нижней границей. Если вам нужно обеспечить более 68 % вероятности в середине диапазона в одно стандартное отклонение, используйте соответствующие квалификаторы (в табл. 10.6 приведен список стандартных отклонений и связанных с ними вероятностей).

Подумайте, должен ли «минусовой» квалификатор совпадать с «плюсовым». Для объемов работ и сроков «минусовой» квалификатор обычно меньше «плюсового» по причинам, объясненным в разделе 1.4.

Впрочем, у квалификаторов «плюс/минус» имеются и свои слабости: по мере прохождения оценки по организации она обычно усекается до минимума. В отдельных случаях менеджеры упрощают оценку из нежелания учитывать изменчивость, заложенную в оценке. Чаще оценка упрощается из-за того, что их начальники или корпоративная бюджетная система принимают только точечные оценки, выраженные в виде отдельных чисел. Используя эту методику, убедитесь в том, что вас устроит точечная оценка, оставшаяся после преобразования диапазона в упрощенную форму.

Квантификация рисков

Квантификация рисков представляет собой комбинацию квалификаторов «плюс/минус» и предположений проекта. В результате квантификации риски ассоциируются с конкретными последствиями, как показано в табл. 22.1.

Таблица 22.1. Пример оценки с квантификацией рисков**Оценка: 6 месяцев, +5 месяцев, -1 месяц****Последствия Описание риска**

+ 1,5 месяца	Версия потребует более 20 % новых возможностей по сравнению с версией 2.0
+1 месяц	Подсистема форматирования графики будет доставлена позднее, чем планировалось
+1 месяц	Новый инструментарий разработки будет работать не так хорошо, как планировалось
+1 месяц	Не удастся заново использовать 80 % кода из предыдущей версии
+0,5 месяца	Средняя заболеваемость персонала в летние месяцы будет средней (а не пониженной)
-0,5 месяца	100%-е комплектование разработчиков завершено к 1 апреля
-0,5 месяца	Новый инструмент разработки будет работать лучше, чем планировалось

В этом относительно простом примере внимание направлено исключительно на риски для сроков. В реальной ситуации, помимо последствий рисков для сроков, могут быть перечислены основные последствия рисков для объемов и функциональности. Следует помнить, что речь идет о методике *представления* оценки. Ее целью является передача информации о рисках нетехническим ключевым участникам проекта. Чтобы не перегружать их подробной информацией о рисках, постарайтесь сосредоточиться на монолитных, укрупненных рисках.

При документировании источников неопределенности предоставьте ответственным участникам проекта информацию, которая может использоваться для сокращения рисков, и заложите основу для объяснения изменений в оценках в случае реализации каких-либо из рисков.

Если проект уже достиг достаточно зрелого состояния для принятия *обязательств*, риски из табл. 22.1 могут оказаться рисками для выполнения обязательств, а не рисками для оценки.

Представленный пример не отражает общей неопределенности проекта, обусловленной воздействием конуса неопределенности. Если обязательства еще не принимались, возможно, вам также придется учесть и этот вид неопределенности.

СОВЕТ № 105

Вы должны четко понимать, какая неопределенность отражается в представлении оценки: неопределенность самой оценки или же неопределенность, влияющая на вашу возможность выполнения обязательств.

Коэффициенты достоверности

Один из вопросов, которые часто задаются о сроках — «Какова вероятность того, что мы управимся к этой дате?» Коэффициенты достоверности позволяют вам ответить на этот вопрос и привести оценку вроде той, что представлена в табл. 22.2.

Для приближенной оценки интервалов можно воспользоваться оценкой «наиболее вероятного» результата и множителями из табл. 4.1, соответствующими фазе проекта.

Как упоминалось в главе 2 и в других местах книги, избегайте излишне увереных («достоверных на 90 %») оценок, если только вы не располагаете количественными показателями, подтверждающими столь высокие проценты.

256 Глава 22 • Стили представления оценок

Таблица 22.2. Пример оценки с коэффициентами достоверности

Дата выдачи	Вероятность выдачи не позднее запланированной даты
15 января	20 %
1 марта	50 %
1 ноября	80 %

Также подумайте, стоит ли представлять оценки с низкой достоверностью. Теоретическая возможность результата еще не означает, что его нужно включать в таблицу. Вряд ли кому-нибудь придет в голову приводить варианты, вероятные на 1 % или 0,0001 %. Представление только тех вариантов, вероятность которых составляет не менее 50 % — вполне разумная стратегия оценки.

СОВЕТ № 106

Не представляйте ответственным сторонам проекта маловероятные результаты.

Некоторые воспринимают визуальные данные лучше, чем данные в табличной форме, поэтому вы также можете создать более наглядное представление — такое, как показано на рис. 22.1.

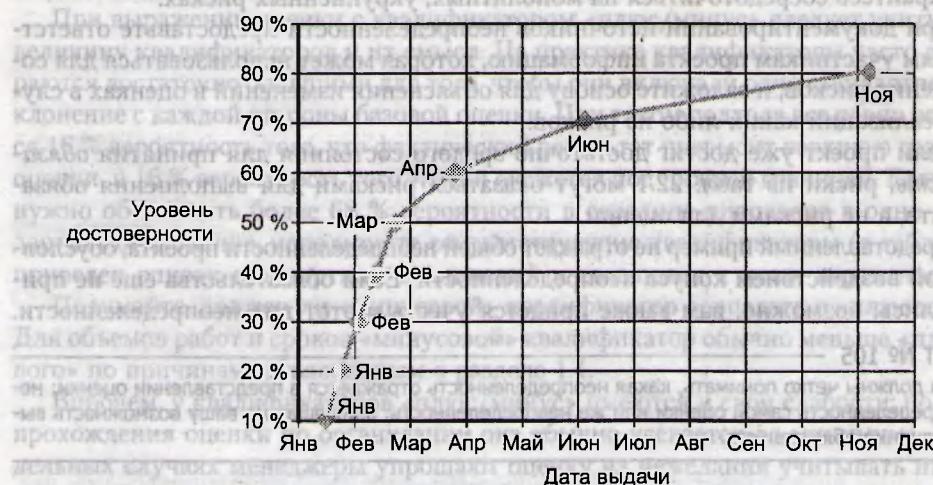


Рис. 22.1. Пример наглядного представления оценок с разной достоверностью в процентах

СОВЕТ № 107

Подумайте, не заменить ли текстовое представление оценки графическим.

Сценарные оценки

Сценарные оценки являются разновидностью оценок с коэффициентами достоверности. Оценка представляется для лучшего случая, худшего случая и текущего случая в сочетании с обязательствами (или запланированного случая).

Промежутки между запланированным и лучшим/худшим случаями дают представление об изменчивости, заложенной в проект, и оптимистичности плана. Если запланированный случай заметно смещен к лучшему исходу, это указывает на оптимизм планирования. В табл. 22.3 представлен пример сценарных оценок.

Таблица 22.3. Пример сценарных оценок

Случай	Оценка/обязательство
Лучший случай (оценка)	15 января
Запланированный случай (обязательство)	1 марта
Текущий случай (оценка)	1 апреля
Худший случай (оценка)	1 ноября

Интерес представляют отношения между датами. Если запланированный случай совпадает с лучшим, а текущий — с худшим, у вашего проекта большие неприятности!

Используя эту методику, будьте готовы объяснить ключевым участникам проекта, что должно произойти для достижения лучшего или худшего случая. Скорее всего, они захотят узнать об обеих возможностях.

На рис. 22.2 показан пример визуального представления информации такого рода.

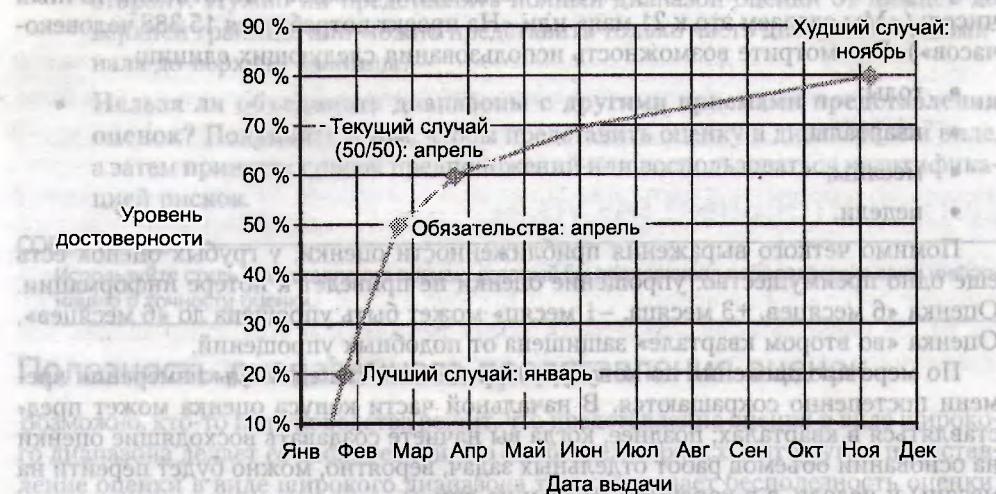


Рис. 22.2. Пример наглядного представления сценарных оценок

В зависимости от того, чему уделяется основное внимание при управлении — срокам или функциональности, сценарная оценка может быть выражена в готовности функций, а не в календарных датах. В табл. 22.4 показано возможное представление сценарной оценки для функциональности.

Таблица 22.4. Пример сценарной оценки для функциональности

Случай	Оценка/обязательство
Лучший случай (оценка)	100 % функций уровня 1
Запланированный случай (обязательство)	100 % функций уровня 2
	100 % функций уровня 3
Текущий случай (оценка)	100 % функций уровня 1
	100 % функций уровня 2
	50 % функций уровня 3
Худший случай (оценка)	100 % функций уровня 1
	80 % функций уровня 2
	0 % функций уровня 3
	100 % функций уровня 1
	20 % функций уровня 2
	0 % функций уровня 3

Грубая оценка дат и периодов времени

Старайтесь представлять свои оценки в единицах, соответствующих точности базовых данных оценки. Если оценки являются приближенными, используйте очевидно грубые оценки («Мы сделаем это во втором квартале», или «Проект потребует 10 человеко-лет») вместо создающих ложное впечатление точных чисел: («Мы сделаем это к 21 мая» или «На проект потребуется 15 388 человеко-часов»). Рассмотрите возможность использования следующих единиц:

- годы;
- кварталы;
- месяцы;
- недели.

Помимо четкого выражения приближенности оценки, у грубых оценок есть еще одно преимущество: упрощение оценки не приведет к потере информации. Оценка «6 месяцев, +3 месяца, -1 месяц» может быть упрощена до «6 месяцев». Оценка «во втором квартале» защищена от подобных упрощений.

По мере продвижения по конусу неопределенности единицы измерения времени постепенно сокращаются. В начальной части конуса оценка может представляться в кварталах; позднее, когда вы начнете создавать восходящие оценки на основании объемов работ отдельных задач, вероятно, можно будет перейти на месяцы и недели, а в конечном итоге и на дни.

22.3. Диапазоны (любого типа)

В книге уже неоднократно говорилось о том, что диапазоны позволяют наиболее адекватно отразить неточность, изначально заложенную в разных точках конуса неопределенности. Диапазоны объединяются с другими приемами, описанными

в этой главе (то есть диапазоны в грубых периодах времени, диапазоны с квантификацией рисков и т. д.).

Представляя оценку в виде диапазона, попробуйте ответить на следующие вопросы.

- **Какой уровень вероятности должен быть включен в диапазон?** Ограничиться ли ± 1 стандартным отклонением (68 % возможных исходов), или диапазон должен быть шире?
- **Как диапазоны воспринимаются бюджетными и отчетными процессами вашей компании?** Учтите, что во многих компаниях бюджетные и отчетные процессы отказываются принимать диапазоны, и последние часто упрощаются по причинам, никак не связанным с оценкой программного обеспечения, например: «Бюджетная электронная таблица нашей компании не позволяет вводить диапазоны». Принимайте во внимание ограничения, которым приходится подчиняться вашему руководству.
- **Устроит ли вас средняя точка диапазона?** Иногда руководство упрощает диапазоны, публикуя их нижнюю границу. Однако чаще при невозможности использования диапазона вычисляется средняя точка, которая используется в качестве оценки.
- **Нужно ли представлять весь диапазон или только часть диапазона от номинальной оценки до верхней границы?** Проекты со временем чаще увеличиваются, чем уменьшаются, и оценки обычно ошибаются в нижнюю сторону. Нужно ли представлять полный диапазон оценки от нижней до верхней границы или можно представить только часть диапазона от номинала до верхней границы?
- **Нельзя ли объединить диапазоны с другими приемами представления оценок?** Подумайте о том, чтобы представить оценку в диапазонном виде, а затем привести список предположений или воспользоваться квантификацией рисков.

СОВЕТ № 108

Используйте стиль представления оценок, который бы подчеркивал передаваемую вами информацию о точности оценки.

Полезность диапазонного представления оценок

Возможно, кто-то из руководства решит, что представление оценки в виде широкого диапазона делает ее бесполезной. На самом деле происходит другое: представление оценки в виде широкого диапазона точно передает бесполезность оценки! Не представление делает оценку бесполезной, а неопределенность, заложенная в самой оценке. Невозможно исключить неопределенность из оценки, представляя ее без неопределенности. Тем самым вы лишь игнорируете неопределенность, но это лишь обернется всем во вред.

Два крупнейших профессиональных сообщества разработчиков программного обеспечения — IEEE Computer Society и Association of Computing Machinery — совместно решили, что включение неопределенности в оценку относится к числу

профессиональных обязанностей разработчиков. Статья 3.9 этического кодекса разработчика IEEE-CS/ACM гласит:

«Разработчики должны следить за тем, чтобы их продукты и модификации соответствовали высочайшим профессиональным стандартам. В частности, разработчики обязаны по обстоятельствам:

3.09 Обеспечивать реалистичную количественную оценку стоимости, сроков, персонала, качества и результатов всех проектов, в которых они работают или собираются работать, и обеспечить анализ неопределенности в таких оценках».

(выделено автором).

Иначе говоря, включение неопределенности в оценку — это не любезность, а этическая ответственность разработчика.

Диапазоны и обязательства

Когда руководство сопротивляется принятию диапазонных оценок, иногда оно на самом деле сопротивляется включению оценки в обязательства. В таких случаях следует представить широкий диапазон оценки и указать, что в оценке существует слишком большая доля неопределенности для принятия осмысленных обязательств.

После сокращения неопределенности до уровня, который позволяет принимать обязательства, диапазоны обычно не являются оптимальным средством выражения обязательств. Диапазон оценки наглядно представляет саму природу обязательств — сопряженных с большей или меньшей степенью риска, однако сами обязательства обычно должны выражаться в виде отдельных чисел.

СОВЕТ № 109

Не пытайтесь выражать обязательства в диапазонной форме. Обязательства должны быть более конкретными.

Дополнительные ресурсы

Gotterbarn, Don, Keith Miller, and Simon Rogerson. «Computer Society and ACM Approve Software Engineering Code of Ethics», «IEEE Computer», October 199, pp. 84–88 (www.computer.org/computer/code-of-ethics.pdf). В статье рассказано, как принимался этический кодекс разработчика, и приводится его полный текст.

Политика, переговоры и решение проблем

23

Факт существования внесших требования еще не означает, что эти требования могут быть выполнены. Тогда же, когда вы получите от заказчика требование, что все требования должны быть выполнены, это означает, что вы должны выполнить все требования.

Совет: Многие из этих требований от заказчика являются враждебными. Убедитесь, что вы можете выполнить эти требования и не захотите.

Бюджет и даты: Бюджет и даты — это основные критерии для оценки. Установленные бюджетом и сроком работы должны соответствовать реальным потребностям и возможностям.

Область применения методов этой главы:

Принципиальные переговоры	
Что оценивается	Размер, объем работы, срок, функциональность
Размер проекта	М С Б
Стадия разработки	Ранняя—поздняя
Итеративный или последовательный стиль	Оба
Возможная точность	—

Филип Метцгер (Philip Metzger) много лет назад заметил, что технические специалисты хорошо составляют оценки, но плохо их защищают (Metzger 1981). У меня нет оснований полагать, что за прошедшие годы технические специалисты научились лучше защищать свои оценки. В этой главе описаны причины, по которым возникают трудности с принятием оценок, и методика, помогающая успешно вести переговоры по поводу оценок.

23.1. Особенности руководства

Первая проблема в переговорах по оценкам обусловлена спецификой личности людей, участвующих в переговорах. Технические специалисты обычно склонны к интроверсии. Около 3/4 технических работников являются интровертами, в отличие от 1/3 всего населения (McConnell 2004b). Многие технические специалисты нормально ладят с другими людьми, но в области конфликтных социальных взаимодействий они не сильны.

В переговорах по поводу программных проектов обычно участвует технический персонал и администраторы или технический персонал и маркетологи. Джеральд Вайнберг (Gerald Weinberg) указывает, что маркетологи и администраторы обычно по крайней мере на 10 лет старше и занимают более высокие должности,

чем технические специалисты. Кроме того, переговоры являются одним из основных аспектов их работы (Weinberg 1994). Другими словами, переговоры по поводу оценок ведутся между интровертами-«технарями» и опытными профессиональными переговорщиками.

С учетом этого обстоятельства становится понятно, почему технические специалисты относятся к переговорам примерно так же, как к удалению зубов без наркоза. Факторы, затрудняющие переговоры с руководящими работниками, вряд ли изменятся в обозримом времени. В табл. 23.1 перечислены некоторые из этих факторов.

Таблица 23.1. Десять основных характеристик администраторов

1. Администраторы всегда требуют то, что хотят получить.
2. Администраторы всегда пытаются получить то, что хотят, если не вышло с первого раза.
3. Администраторы склонны прощупывать почву до тех пор, пока не обнаружат ваше слабое место.
4. Администраторы не всегда знают, что невозможно, но всегда знают, что было бы полезно для бизнеса, если бы было возможно.
5. Администраторы настойчивы. Собственно, без этого они бы не стали администраторами.
6. Администраторы будут уважать вас, если вы будете настойчивы. На самом деле они считают, что вы будете настойчивы только при крайней необходимости.
7. Администраторы хотят, чтобы вы в своей работе руководствовались интересами организации.
8. Администраторы стремятся исследовать как можно больше вариантов, чтобы добиться максимальной коммерческой эффективности.
9. Администраторам известно о бизнесе, рынке и о компании много такого, что неизвестно вам. Их приоритеты проекта могут отличаться от ваших.
10. Администраторы всегда стремятся получить четкие прогнозы и обязательства на ранней стадии (действительно, это было бы крайне полезно для дела — если бы было возможно).

Как правило, администраторы обладают этими характеристиками, потому что это *полезно для организации*. Не рассчитывайте, что эти характеристики когда-нибудь изменятся!

СОВЕТ № 110

Поймите, что администраторы настойчивы по своей природе и по должности, и планируйте обсуждение оценок соответствующим образом.

Возможно, переговоры по поводу оценок — одна из составляющих вашей работы, которая вам не нравится, но никто не обещал, что ваша работа будет на 100 % приятной. Я обнаружил, что простое осознание этого факта, что я не люблю переговоры, помогает мне вести их более конструктивно.

23.2. Влияние политических факторов на оценки

Реакция руководства на оценки программных проектов зависит от ряда факторов, не относящихся к технической стороне дела.

Внешние ограничения

Довольно часто руководство беспокоится о важных внешних факторах, требующих выдать программный продукт к заданной дате или в пределах заданной стоимости. Может действовать внешний, жестко фиксированный предельный срок (например, сезон рождественских продаж, выставка и т. д.). Аналогично, стоимость проекта может ограничиваться условиями тендера; руководство считает, что компания не получит заказ, если выставленные требования будут достаточно высокими для покрытия издержек.

Факт существования внешних требований еще не означает, что эти требования могут быть выполнены. Тем не менее вы должны дать понять руководству, с которым имеете дело, что все требования поняты и вы относитесь к ним серьезно.

СОВЕТ № 111

Учитывайте воздействие внешних факторов. Ваши собеседники должны видеть, что вы понимаете коммерческие требования и их важность.

Бюджет и даты

Во многих компаниях даты выдачи проектов также зависят от квартального деления. Компании сообщают о своих затратах и доходах ежеквартально. Иногда бывает проще перенести проект на более позднюю дату, чем переносить более раннюю дату на предыдущий квартал. Если вы предложите назначить срок выдачи проекта на 15 июля, возможно, на вас будут давить с предложениями перенести его на 30 июня (то есть на второй квартал вместо третьего). Но если предложить в качестве даты выдачи 15 сентября, ближе к концу третьего квартала, может оказаться, что на эту дату руководство согласится скорее, чем на 15 июля, потому что у него будет меньше стимулов сдвигать выдачу на предыдущий квартал. Этот фактор действует еще сильнее на границах отчетных годов.

Переговоры по поводу оценок и обязательств

В одних обстоятельствах переговоры уместны, в других — нет. Мы не ведем переговоры по поводу непреложных фактов (скажем, температуры поверхности Солнца или объема Великих Озер), а просто принимаем их как данность. Аналогично, оценка программного проекта является результатом аналитической деятельности и обсуждать ее нерационально. Однако рационально обсудить обязательства, связанные с оценкой. Обсуждение может выглядеть примерно так:

Технический руководитель: По нашей оценке, проект займет от 5 до 7 месяцев. Впрочем, сейчас мы еще находимся в ранней части конуса неопределенности, поэтому оценка будет уточняться в ходе работы.

Администратор: От 5 до 7 месяцев — слишком большой разброс. Нельзя ли просто использовать оценку в 5 месяцев?

Технический руководитель: Мы должны различать оценки и обязательства. Оценку я изменить не могу, потому что она была получена в результате серьезных вычислений. Тем не менее моя группа может согласиться на срок в 5 месяцев, если мы все согласимся на подобный уровень риска.

Администратор: Ничего не понял. А в чем разница?

Технический руководитель: Наш диапазон от 5 до 7 месяцев включает одно стандартное отклонение в каждую сторону от медианной оценки в 6 месяца. Это означает, что вероятность завершения за 7 месяцев равна 84 %. Наши оценки показывают, что вероятность соблюдения 5-месячных обязательств составляет всего 16 %.

Администратор: Нам нужна более чем 50 % уверенность в сроке, по которому заключаются обязательства, но 84 % – более консервативная оценка, чем нужно. Какой срок можно считать достоверным на 75 %?

Технический руководитель: Согласно нашим вычислениям, около 6,5 месяца.

Администратор: На нем и остановимся.

Технический руководитель: Договорились.

Многие технические специалисты считают, что подобный диалог вредит карьерному росту. По собственному опыту могу сказать, что все совсем наоборот. Если вы готовы пережить неприятные разговоры (и если при этом руководствуетесь интересами своей организации), в действительности это лишь способствует вашей карьере. Настоящий вред карьере приносит другое – принятие необоснованных, нереалистичных обязательств, которые не удается выполнить.

СОВЕТ № 112

Обсуждайте обязательства, но не оценки.

Что делать, если ваша оценка не принята

Разработчики и менеджеры иногда беспокоятся, что слишком высокая оценка приведет к отклонению проекта. Это нормально. Высшее руководство обладает и ответственностью, и правом решить, что проект экономически не оправдан. Когда технический персонал занижает оценку проекта, он скрывает от руководства важную информацию, необходимую для принятия эффективных решений, и фактически мешает ему принимать решения. В результате ресурсы компании отвлекаются с экономически оправданных проектов на экономически неоправданные. Хорошие проекты поддерживаются недостаточно, а плохие проекты получают излишнюю поддержку. Этот сценарий чрезвычайно вреден для дела и обычно плохо кончается для людей, добившихся принятия проектов, которые изначально принимать не следовало.

Ответственность технического персонала за обучение

Если вы хотите гарантировать успех своих программных проектов, расскажите ответственным участникам, не связанным с технической стороной, об опасностях произвольного урезания оценок сроков и стоимости без соответствующего сокращения объема работы. Расскажите о конусе неопределенности, о различиях между оценками, целями и обязательствами. По собственному опыту могу сказать, что эти идеи отлично воспринимаются в контексте попыток действовать в интересах организации.

СОВЕТ № 113

Расскажите ответственным участникам, не связанным с технической стороной проекта, о принципах эффективной оценки программных проектов.

Помимо обучения нетехнических участников проекта займитесь изучением основных принципов, приоритетов и слабых мест вашей отрасли — это поможет вести дискуссии по поводу оценок на конструктивном уровне.

23.3. Решение проблем и принципиальные переговоры

В своей книге «*Rapid Development*» от 1996 года я охарактеризовал обсуждение оценок термином «переговоры». С течением времени я все меньше и меньше уверен в том, что переговоры являются наиболее конструктивным подходом к дискуссиям, возникающим вокруг оценок стоимости, срока и функциональности.

В переговорах участвуют стороны с конфликтующими интересами, а их целью является распределение «пирога» между двумя и более сторонами. При антагонистических переговорах каждая сторона пытается захватить как можно большую часть, и любые приобретения одной стороны автоматически оборачиваются потерями для другой стороны. При совместных переговорах каждая сторона ищет способы увеличения «пирога», но в конечном счете «пирог» все равно делится.

В переговорах по поводу программного обеспечения делить нечего. В переговорах со специалистами по продажам, маркетингу или администрацией мы сидим за одним столом. Вопрос вовсе не ставится в виде «Мы выигрываем, а они проигрывают» — скорее, «мы все проигрываем» или «мы все выигрываем». Наши интересы совпадают. Мы либо закладываем основу для успеха программного проекта, который оборачивается успехом для всех, либо создаем условия для его неудачи, также всеобщей. Следовательно, обсуждения по поводу оценок программных проектов нельзя назвать *переговорами* в традиционном понимании.

Более правильной моделью для дискуссий между техническим персоналом, отделами продаж, маркетинга, администрацией и другими ответственными сторонами является совместное решение проблемы. Мы все работаем на общую цель, делимся своим опытом в разных областях и создаем решение, которое в конечном счете отвечает интересам бизнеса.

СОВЕТ № 114

Относитесь к обсуждению оценок как к решению проблемы, а не как к переговорам. Поймите, что все ответственные участники проекта находятся на одной стороне. Либо все вместе проигрывают, либо все вместе выигрывают.

Осознав, что речь идет не о конфликте интересов, а о совместном решении проблем, мы, технические специалисты, конструктивно подходим к обсуждению целей, оценок и обязательств. Остается только направить других участников обсуждения на такой же конструктивный путь.

Переговоры как совместное решение проблем

Даже если мы знаем, что речь идет о совместном решении проблем, те люди, с которыми мы обсуждаем оценку, могут решить, что они участвуют в традиционных переговорах. Существует много разных стратегий ведения переговоров. Одни стратегии основаны на силе переговорных позиций, устрашении, дружелюбии, стремлении добиться похвалы или заслужить расположение. Другие стратегии базируются на обмане и искусственных психологических маневрах.

Поскольку процесс обсуждения оценок переходит между оценками, целями, обязательствами и планами, обсуждение невозможно четко разделить на «чистые» переговоры и «чистое» решение проблем. Обычно вам придется иметь дело с обоими элементами.

В числе хороших стратегий, заполняющих разрыв между переговорами и решением проблем, можно назвать метод принципиальных переговоров, описанный в книге «Getting to Yes» (Fisher and Ury 1991). Метод называется «переговорами», но его участники рассматриваются как лица, занимающиеся решением задач. Метод не зависит от переговорных приемов; кроме того, он объясняет, как реагировать на их применение другими сторонами. Метод основан на идее создания беспроигрышных альтернатив. Он достаточно хорошо работает, когда используется только вами, и еще лучше — если он используется также и другой стороной.

Стратегия состоит из четырех составляющих:

- отделение людей от проблемы;
- сосредоточение на интересах, а не позициях;
- разработка вариантов, обеспечивающих взаимный выигрыш;
- применение объективных критериев.

Все составляющие метода описываются в следующих секциях.

Отделение людей от проблемы

В обсуждениях оценок задействованы, во-первых, люди, а во-вторых, интересы и позиции. Там, где возникает конфликт личных качеств (например, личных качеств технических работников и специалистов по маркетингу), дискуссия нередко заходит в тупик.

Прежде всего постарайтесь понять позицию другой стороны. Менеджеры часто становятся заложниками устарелой политики своей организации. В некоторых организациях принципы финансирования программных проектов оказываются несовместимыми с методами разработки программного обеспечения. Такие организации не разрешают менеджерам запрашивать финансирование только на то, чтобы разработать требования и планы и выработать хорошую оценку. Чтобы получить средства на осмысленную оценку, менеджер должен сначала добиться финансирования всего проекта. К моменту получения осмысленной оценки ему становится неудобно (и даже небезопасно для карьеры) возвращаться и запрашивать финансирование в реальном объеме. Люди, занимающие самые высокие посты в таких организациях, должны лучше понять принципы оценки программного

обеспечения, чтобы ввести в действие практику финансирования, поддерживающую эффективную разработку программного обеспечения.

В дискуссиях такого рода считайте себя советником по оценкам программных проектов; это поможет избежать вхождения в роль противника. Страйтесь свое-временно возвращать дискуссию к достижению целей, полезных для компании.

Также полезно убрать эмоции из предмета обсуждения. Иногда для этого проще всего дать собеседникам «выпустить пар». Не реагируйте эмоционально на эмоции собеседников; дайте им возможность высказаться. Скажите что-нибудь вроде: «Я вижу, что мы столкнулись с серьезными проблемами, и хочу убедиться в том, что я в полной мере понимаю позицию нашей компании. Что еще вы можете сказать о нашей ситуации?» Когда собеседники завершат объяснения, соглашитесь со всем услышанным и снова выразите свое желание найти решение, подходящее для организации. Другие составляющие процесса принципиальных переговоров помогут вам сформулировать это решение.

СОВЕТ № 115

Нападайте на проблему, а не на людей.

Сосредоточение на интересах, а не позициях

Предположим, вы продаете свою машину, чтобы купить новую лодку. Вы рассчитали, что для покупки нужной вам лодки потребуется \$10 000. Потенциальный покупатель обращается к вам и предлагает \$9000. Вы говорите: «Я ни при каких условиях не расстанусь с машиной менее чем за \$10 000». Покупатель отвечает: «Я могу заплатить \$9000, но это мой максимум».

Переговоры такого рода сосредоточены на позициях, а не на интересах. Позиций называются переговорные утверждения настолько узкие, что выигрыш одной стороны означает проигрыш другой.

Теперь, допустим, покупатель говорит: «Я действительно не могу заплатить выше \$9000, но я слышал, что вы хотите купить новую лодку. По случайности я являюсь региональным дистрибутором крупной компании, торгующей лодками. Я могу продать вам лодку на \$2000 дешевле, чем любой другой продавец. Что вы теперь скажете на мое предложение?» На этот раз предложение выглядит весьма заманчиво — вы получите на \$1000 больше, чем если бы покупатель просто согласился на исходную цену.

Интересы шире переговорных позиций; если вы направите свое внимание на них, перед вами откроется целый мир новых возможностей. Рассмотрим следующий сценарий:

Администратор: Версия Giga-Blat 4.0 нам нужна через 6 месяцев.

Технический руководитель: Мы тщательно оценили проект. К сожалению, оценки показывают, что мы не сможем выдать его менее чем за 8 месяцев.

Администратор: Этого недостаточно. Программа действительно нужна через 6 месяцев.

Технический руководитель: Нам действительно нужна вся заявленная функциональность? Если урезать достаточное количество функций, мы сможем завершить работу за 6 месяцев.

Администратор: Урезать функциональность нельзя. В этой версии мы и так сократили ее до минимума. Нам нужны все функции, и не позже чем через 6 месяцев.

Технический руководитель: Какой главный фактор определяет 6-месячный срок? Возможно, нам удастся найти взаимоприемлемое решение.

Администратор: Через 6 месяцев состоится ежегодная торговая выставка. Пропустив ее, мы лишимся возможности продемонстрировать программу многим важным клиентам. Фактически это задержит цикл продаж на целый год.

Технический руководитель: Я действительно не могу обещать выдать окончательную версию к торговой выставке. Но я могу поручиться, что бета-версия будет готова к выставке, а предоставленный мной специалист будет знать основные проблемы и сможет управлять программой так, что она не «упадет» во время выставки. Что вы на это скажете?

Администратор: Если вы обещаете, что программа будет работать без сбоев, меня это устроит.

Технический руководитель: Договорились.

Важнейшее различие между типичными переговорами и решением проблем посредством обсуждения интересов состоит в том, что переговоры обычно «замораживаются» на позициях. Поворотным моментом в этом диалоге стал вопрос технического руководителя о главном факторе, определяющем 6-месячный срок. Диалог переключился со спора о позициях на попытки понять интересы компании и решить базовую проблему. Сосредоточившись на интересах, вы с большей вероятностью найдете взаимоприемлемое решение.

Разработка вариантов, обеспечивающих взаимный выигрыш

Ваш сильнейший союзник при обсуждении оценок — не оценка, а ваше умение предлагать *варианты планирования*, о которых ничего не известно нетехническим участникам. Вы держите ключ от сейфа с техническими знаниями, и по этой причине ответственность за выдачу творческих решений ложится в большей степени на ваши плечи. Именно вы должны предложить полный спектр возможностей и компромиссов.

В табл. 23.2 перечислены некоторые варианты планирования, которые часто помогают выйти из тупиковой ситуации.

Таблица 23.2. Варианты планирования, которые часто помогают выйти из тупиковой ситуации

Варианты, относящиеся к функциональности

Перемещение части желательных функций в версию 2. Мало кому действительно необходимо получить сразу все, что они просили

Применение итеративного подхода. Выдавайте программу в версиях 0.2, 0.4, 0.6, 0.8 и 1.0, начиная с реализации важнейшей функциональности

Полное исключение некоторых возможностей. В первую очередь исключаются возможности, связанные с максимальными затратами

Использование «метода футбольки» для первоочередной реализации функций, обладающих наибольшей коммерческой ценностью

Упрощенная реализация некоторых функций. Такие функции реализуются до определенной степени, но делаются менее изощренными

Ослабление подробных требований по каждой функции. Определите свою задачу как максимальное приближение к требованиям с использованием существующих компонентов

Построение конуса неопределенности, ориентированного на функциональность. Определите одни функции как «абсолютно обязательные», другие как «абсолютно лишние» и третьи как «возможные». Предложите план сужения функционального конуса неопределенности по мере продвижения проекта

Варианты, относящиеся к ресурсам

Увеличение численности разработчиков или тестеров, если проект еще находится на ранней стадии
Наем внештатного персонала, если проект еще находится на ранней стадии

Введение высокопроизводительного технического персонала (например, экспертов по предметной области или дополнительных ведущих разработчиков)

Увеличение административной поддержки

Повышение уровня поддержки разработчиков

Расширение участия конечных пользователей или клиентов. Включите в проект на полный рабочий день пользователя с правом принятия обязывающих решений по поводу функциональности продукта

Расширение участия руководства, позволяющее ускорить принятие решений

Передача части работы другой группе (но помните о проблемах интеграции, которые при этом возникнут)

Выделение 100 % ресурсов на проект. Участники проекта не должны делить свое рабочее время между новым и старым проектом, или несколькими новыми проектами

Варианты, относящиеся к срокам

Составление «дорожной карты оценок» с изложением плана пересмотра и уточнения оценок

Использование диапазонных или грубых оценок, уточняемых по мере продвижения проекта

Поиск способов планирования для достижения определенных целей по срокам, стоимости или функциональности, по мере уточнения требований и планов

Задержка некоторых обязательств до завершения следующей фазы проекта (то есть работы, необходимой для сужения конуса неопределенности)

Выполнение одной-двух коротких итераций для калибровки производительности и последующее принятие обязательств на основании полученных данных

Главное — предотвратить перепалки типа «Я не могу это сделать».— «Можете».— «Нет, не могу».— «Можете!» — «Не могу!!!» Представьте набор вариантов и сосредоточьте обсуждение на них. Не включайте заранее невозможные варианты в свой набор. Страйтесь не говорить «Нет, это невозможно»; вместо этого направьте дискуссию на то, что возможно. Чем больше вы представите вариантов, работающих на благо организации, тем проще вам будет показать, что вы находитесь на одной стороне с человеком, с которым решается проблема.

СОВЕТ № 116

Выдайте как можно больше вариантов планирования, поддерживающих цели вашей организации.

И одно предупреждение: в атмосфере «мозгового штурма», возникающей в ходе свободной дискуссии при решении проблем, легко согласиться на решение, которое

вам в тот момент кажется удачным. Однако на следующее утро идея выглядит уже не столь привлекательно. В этой ситуации действуют предупреждения, приведенные в разделе 4.8. Не принимайте жестких обязательств, пока у вас не будет достаточно времени для спокойного анализа их последствий.

СОВЕТ № 117

В процессе совместного решения проблем не принимайте поспешных обязательств, основанных на импровизированных оценках.

Применение объективных критериев

В нашем деле есть одна странность: если тщательно выверенная оценка значительно превышает желаемую, клиент или начальник часто попросту игнорирует ее (Jones 1994). Такое возможно даже в том случае, если оценка была получена в оценочной программе, или представлена внешним экспертом, или организация известна частыми превышениями своих оценок в прошлом. Подвергать сомнению оценку — абсолютно допустимо и даже полезно. Проигнорировать оценку и заменить ее своими домыслами — нет.

Когда принципиальные переговоры переходят в решение проблем, вы ищете «разумное соглашение, оцениваемое по каким-либо объективным стандартам». Вы можете спорить с собеседниками относительно того, какие объективные стандарты являются наиболее подходящими, но относитесь без предвзятости к предлагаемым им стандартам. Что еще важнее, не поддавайтесь давлению — только принципам. Чтобы поддержать дискуссию, основанную на принципах, необходимо определить, кто является компетентной стороной в каждом аспекте дискуссии.

Технический персонал и техническое руководство формируют оценку

Вы лучше других понимаете техническую часть работы и создание оценок для нее. Следовательно, вы должны быть основным авторитетом в оценке.

Нетехнические ответственные стороны формируют цели

Администраторы, специалисты по продажам и маркетингу вследствие своей позиции обычно лучше понимают потребности и приоритеты бизнеса. По этой причине они являются основными авторитетами по бизнес-целям.

Технический и нетехнический персонал совместно формируют обязательства

Цели и оценки в конечном итоге сходятся в обязательствах. Если вам удастся достичь соглашения относительно того, что вы авторитетны в оценке, а другие стороны — в целях, большая часть дискуссии естественным образом сосредоточится на обязательствах. Определяющим принципом должно стать достижение соглашения относительно того, какие обязательства будут лучшими для организации.

В этих дискуссиях необходимо учитывать следующие рекомендации.

- **Не обсуждайте саму оценку.** Объясните собеседникам, чем оценка отличается от обязательств. Постоянно направляйте дискуссию на достижение обязательств, отвечающих интересам организации.
- **Настаивайте на том, что оценка должна быть подготовлена компетентной стороной.** Самым компетентным оценщиком нередко оказываетесь вы;

в других случаях им может быть независимая группа оценки. Такие группы эффективны, прежде всего, потому, что они не имеют субъективной заинтересованности ни в выдаче программы за кратчайшее возможное время, ни в том, чтобы избежать тяжелой работы. Если дискуссия зациклится на теме самой оценки, предложите передать оценку третьей стороне и согласитесь принять результат. Попросите другие стороны сделать то же самое.

- Вариация на эту тему — привлечение консультанта или внешнего эксперта для анализа сроков. Посторонний эксперт иногда вызывает больше доверия, чем знакомый. Некоторые организации также добивались успеха, используя оценочные программы. После того как технический персонал откалибрует программу для конкретного проекта, они могли легко и объективно проанализировать последствия различных решений.
- **Ссылайтесь на стандартизированную процедуру оценки вашей организации.** Предварительное принятие стандартизированной процедуры оценки часто позволяет избежать дискуссий по поводу того, кто должен создавать оценку. Вы можете просто сказать: «Наша процедура не позволяет обсуждать саму оценку. Давайте лучше поговорим об основных предположениях оценки (таких, как размер проекта) и о том уровне риска, при котором для организации будет оправдано принятие обязательств по данному проекту».
- **Не поддавайтесь эмоциям.** Люди обладают разной восприимчивостью к давлению, но если ваши клиенты, менеджеры, специалисты по маркетингу и другие стороны хотят, чтобы вы приняли на себя невыполнимые обязательства, на мой взгляд, лучше всего вежливо и твердо отстаивать свои убеждения. Лучше задраить люки и пережить бурю, связанную с неприятными оценками, на ранней стадии проекта, чем ураган нарушений сроков и перерасхода бюджета.

В действительности никто не выигрывает от принятия заведомо невыполнимых целей, хотя некоторым людям кажется иначе. Отстаивая решения, соответствующие реальным деловым потребностям вашего начальства и клиентов, вы лишь повышаете свою репутацию. Обеспечьте предсказуемость и помогите вашей организации соблюдать принятые обязательства.

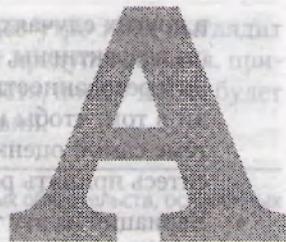
СОВЕТ № 118

Чтобы выйти из тупика, возникшего в дискуссии, почтите возвращайтесь к вопросу: «Что будет лучше для нашей организации?»

Дополнительные ресурсы

Fisher, Roger, William Ury, and Bruce Patton. «Getting to Yes, Second Edition». New York, NY: Penguin Books, 1991. В книге приводится подробное описание стратегии принципиальных переговоров, представленной в этой главе. Книга содержит массу интересных примеров и хорошо читается даже в том случае, если вы не интересуетесь проблемами переговоров.

Проверка разумности оценки



Следующий вопросник поможет определить, какую пользу текущая оценка принесет при управлении проектом. За каждый положительный ответ начисляется одно очко.

1. Использовалась ли стандартизированная процедура при создании оценки?
2. Был ли процесс оценки свободен от давления, способного повлиять на результаты?
3. Если по поводу оценки проводились переговоры, обсуждались ли в них только входные параметры оценки (без обсуждения результатов или самого процесса оценки)?
4. Соответствует ли четкость представления оценки ее точности? (Например, выражается ли оценка в виде диапазона или грубого приближенного значения на ранних стадиях проекта?)
5. Была ли оценка получена с применением нескольких альтернативных методов, приводящих к сходным результатам?
6. Сравнимы ли предположения о производительности, заложенные в основу оценки, с производительностью, фактически достигнутой в прошлых проектах аналогичного размера?
7. Составляет ли оцениваемый срок по меньшей мере $2,0 \times \text{ЧеловекоМесяцы}^{1/3}$? (То есть находится ли оценка за пределами зоны невозможности?)
8. Участвовали ли в создании оценки люди, которым предстоит выполнять работу?
9. Была ли оценка проанализирована экспертом в области оценки?
10. Включен ли в оценку ненулевой допуск для проектных рисков, влияющих на объем работы и сроки?

Рекомендации

11. Входит ли оценка в серию оценок, точность которой постепенно повышается с переходом в узкую часть конуса неопределенности?
12. Учтены ли в оценке *все* элементы проекта, включая создание программы установки, утилит преобразования данных, адаптацию старых систем и т. д.?

ИТОГО: _____

Процедура проверки взята из книги «Software Estimation» Стива Макконнела (Microsoft Press, 2006), © 2006 Steve McConnell. All rights reserved. Копирование разрешено при условии сохранения данного уведомления об авторском праве.

Результат

10–12 — оценка обладает высокой точностью.

7–9 — оценка достаточно хороша для управления проектом, но, скорее всего, несколько оптимистична.

<6 — оценка подвержена существенному смещению и/или слишком оптимистична, а ее точности недостаточно для осмысленного управления проектом.

СОВЕТ № 1	С = 000	БИНОМИАЛЬНЫЙ ОЦЕНЧИВАНИЕ ВЫПУСКАЕТ РАХИБШ ВТОРИЙ ПРОДУКТ
	Не путайте оценки, или и обратительства, вводимые хранящимися в базах данных.	РАХИБШ ВТОРИЙ ПРОДУКТ
СОВЕТ № 2	С = 000	СТАТИСТИЧЕСКАЯ ОЦЕНКА ВЫПУСКАЕТ РАХИБШ ВТОРИЙ ПРОДУКТ
	Когда оцениваете предполагаемую цену, учитывайте факторы, влияющие на цену или путь достижения цели.	РАХИБШ ВТОРИЙ ПРОДУКТ
СОВЕТ № 3	С = 01 × 8,8	СТАТИСТИЧЕСКАЯ ОЦЕНКА ВЫПУСКАЕТ РАХИБШ ВТОРИЙ ПРОДУКТ
	Становитесь стойкой «кошкой», спросите себя, действительно ли это число является реальной и вербальной оценкой цели.	СТАТИСТИЧЕСКАЯ ОЦЕНКА ВЫПУСКАЕТ РАХИБШ ВТОРИЙ ПРОДУКТ
СОВЕТ № 4	С = 000	БИНОМИАЛЬНЫЙ ОЦЕНЧИВАНИЕ ВЫПУСКАЕТ РАХИБШ ВТОРИЙ ПРОДУКТ
	Если вы сталкиваетесь с туманной оценкой, скорее всего, ее достоверность отлична от 100 %.	БИНОМИАЛЬНЫЙ ОЦЕНЧИВАНИЕ ВЫПУСКАЕТ РАХИБШ ВТОРИЙ ПРОДУКТ

Глава 2**СОВЕТ № 5**

Не представляйте процентные оценки достоверности (особенно «достовернее на 90 %»), если только они не подкрепляются количественными методами.

СОВЕТ № 6

Избегайте искусственного сужения диапазона. Следите за тем, чтобы диапазоны, используемые в оценках, не исказили вашего представления о достоверности оценки.

СОВЕТ № 7

Если вы испытываете воздействие, направленное на сужение диапазона, убедитесь в том, что оно идет извне, а не рождается внутри вас.

Ответы на вопросы теста из главы 2


Вопрос
Описание

Температура поверхности Солнца	6000 °С
Широта Шанхая	31° северной широты
Площадь континента Азия	44 390 000 квадратных километров
Год рождения Александра Македонского	356 до н. э.
Общая стоимость американской валюты, находившейся в обращении в 2004 г.	719,9 миллиарда долларов
Общий объем Великих Озер	$6,8 \times 10^{20}$ кубических метров $6,8 \times 10^{23}$ литров
Мировые кассовые сборы фильма «Титаник»	1,835 миллиарда долларов
Общая длина линии побережья Тихого океана	135 663 километров
Количество названий книг, опубликованных в США с 1776 г.	22 миллиона
Максимальный зафиксированный вес голубого кита	170 тонн

тодов, приводящих к сходным результатам?

- б. Сравнивая предположения о производительности, заложенные в оценку, с производительностью, фактически достигнутой в прошлых проектах аналогичного размера?
7. Составляет ли оцениваемый срок по меньшей мере 2,0 × ЧелленджМэстри? (То есть находится ли оценка за пределами зоны невозможности?)
8. Участвовали ли в создании оценки люди, которым предстоит выполнять работу?
9. Была ли оценка проанализирована экспертом в области оценки?
10. Включен ли в оценку ненулевой допуск для проектных рисков, влияющих на объем работы и сроки?

Рекомендации по оценке программных проектов

B

Глава 1

СОВЕТ № 1

Не путайте оценки, цели и обязательства.

СОВЕТ № 2

Когда вас просят предоставить оценку, определите, что именно нужно спрашивающему — оценка или план достижения цели.

СОВЕТ № 3

Сталкиваясь с точечной «оценкой», спросите себя, действительно ли это число является оценкой или на самом деле перед вами цель.

СОВЕТ № 4

Если вы сталкиваетесь с точечной оценкой, скорее всего, ее вероятность отлична от 100 %. Спросите себя, какова вероятность получения этого числа.

Глава 2

СОВЕТ № 5

Не предоставляйте процентные оценки достоверности (особенно «достоверные на 90 %»), если только они не поддерживаются количественными методами.

СОВЕТ № 6

Избегайте искусственного сужения диапазонов. Следите за тем, чтобы диапазоны, используемые в оценках, не искажали вашего представления о достоверности оценки.

СОВЕТ № 7

Если вы испытываете воздействие, направленное на сужение диапазона, убедитесь в том, что оно идет извне, а не рождается внутри вас.

Глава 3

СОВЕТ № 8

Избегайте намеренной недооценки. Потери от недооценки превышают потери от переоценки. Если вас беспокоит возможная переоценка, решайте проблемы посредством планирования и управления, а не за счет смещения оценки.

СОВЕТ № 9

Несоответствие между деловыми целями и оценкой проекта следует рассматривать как ценную информацию о возможной неудаче проекта. Принимайте меры на ранней стадии, когда еще можно что-то сделать.

СОВЕТ № 10

Многие организации ценят предсказуемость выше, чем срок разработки, затраты или гибкость. Обязательно выясните, какие показатели считаются приоритетными в вашем случае.

Глава 4

СОВЕТ № 11

Проанализируйте воздействие конуса неопределенности на точность вашей оценки. Ваша оценка не может быть более точной, чем это возможно на текущей позиции проекта внутри конуса.

СОВЕТ № 12

Не надейтесь, что конус неопределенности будет сужаться сам собой. Вы должны заставить его сужаться, устранив источники неопределенности из проекта.

СОВЕТ № 13

Учитывайте наличие конуса неопределенности, закладывая в своих оценках заранее определенную амплитуду неопределенности.

СОВЕТ № 14

Учитывайте наличие конуса неопределенности, разделяя оценку на две составляющие: один специалист дает количественную оценку, а другой оценивает ее неопределенность.

СОВЕТ № 15

Не рассчитывайте, что совершенствование методики оценки само по себе обеспечит более точную оценку в хаотических проектах. Невозможно точно оценивать процесс, который вами не контролируется. На первом шаге важнее избавиться от хаоса, чем совершенствовать оценку.

СОВЕТ № 16

В условиях нестабильных требований следует ориентироваться на стратегии управления проектом вместо стратегий оценки (или совместно с ними).

СОВЕТ № 17

Включайте в оценку время для всех видов требований — заявленных, неявных и нефункциональных. Ничто не создается «из ничего», и ваша оценка не должна подразумевать, будто возможно обратное.

СОВЕТ № 18

Учитывайте в оценке все необходимые действия, связанные с разработкой программного обеспечения, не только программирование и тестирование.

СОВЕТ № 19

В проектах, продолжительность которых превышает несколько недель, следует предусматривать допуск для дополнительных факторов: праздников, отпусков по болезни, времени обучения, собраний.

СОВЕТ № 20

Не уменьшайте оценки, полученные от разработчиков, — скорее всего, они и без того излишне оптимистичны.

СОВЕТ № 21

Избегайте включения «регуляторов» в свои оценки. Хотя может показаться, будто регуляторы улучшают точность, на самом деле они вводят в оценку субъективность и снижают реальную точность.

СОВЕТ № 22

Не давайте импровизированных оценок. Даже если потратить на оценку всего 15 минут, она будет более точной.

СОВЕТ № 23

Количество значащих цифр в оценке (ее четкость) должно соответствовать точности оценки.

Глава 5

СОВЕТ № 24

Приложите соответствующие усилия для оценки размера программы, над которой вы работаете. Размер вносит наиболее значительный вклад в определение объема работы и сроков.

СОВЕТ № 25

Не следует предполагать, что объем работ линейно зависит от размера проекта. Рост происходит по экспоненте.

СОВЕТ № 26

Используйте специализированные программы для вычисления влияния издержек масштаба.

СОВЕТ № 27

Если ранее вы уже завершали предыдущие проекты, размер которых незначительно отличается от размера оцениваемого проекта (в диапазоне, в котором самый большой проект превосходит самый мелкий не более чем в 3 раза), для оценки нового проекта можно смело использовать масштабный коэффициент (скажем, количество строк кода на человека-месяц).

СОВЕТ № 28

В своей оценке учитывайте тип разрабатываемой программы — в отношении влияния на объем работы и сроки этот фактор является вторым по значимости.

Глава 6

СОВЕТ № 29

При выборе метода оценки следует учитывать оцениваемый показатель, размер проекта, стадию разработки, стиль разработки и требуемую точность.

278 Приложение В • Рекомендации по оценке программных проектов

Глава 7

СОВЕТ № 30

Считайте везде, где это возможно. Если счет невозможен — вычисляйте. Используйте оценки, полученные на основании одного лишь экспертного суждения, только в крайнем случае.

СОВЕТ № 31

Найдите счетный показатель, который может использоваться для осмысленного измерения содержания работы в вашей среде.

СОВЕТ № 32

Соберите исторические данные, которые позволят вам вычислить оценку по счетным показателям.

СОВЕТ № 33

Не пренебрегайте возможностями простых, грубых оценочных моделей — таких, как средний объем работы на дефект, средний объем работы на веб-страницу, средний объем работы на историю пользователя и средний объем работы на сценарий использования.

СОВЕТ № 34

Не используйте экспертные суждения для подгонки оценок, полученных на основании вычислений. Подобная «экспертиза» обычно лишь ухудшает точность оценки.

СОВЕТ № 35

Стройте свои оценки производительности на исторических данных. Производительность вашей организации в прошлом дает наилучшее представление о ее производительности в будущем.

СОВЕТ № 36

Исторические данные помогают избежать решений, имеющих политическую подоплеку и возникающих из предположений типа «Моя группа ниже среднего».

Глава 8

СОВЕТ № 37

При сборе исторических данных для оценок убедитесь в том, что вы понимаете смысл показателей, и не изменяйте его между проектами.

СОВЕТ № 38

Собирайте исторические данные как можно раньше после начала проекта.

СОВЕТ № 39

Организуйте периодический сбор исторических данных во время работы над проектом. Это позволит вам позднее построить профиль выполнения проекта, базирующийся на собранных данных.

СОВЕТ № 40

Используйте данные, полученные в ходе текущего проекта (проектные данные), для создания высокоточных оценок для оставшейся части проекта.

СОВЕТ № 41

Там, где это возможно, используйте для калибровки оценок проектные или исторические данные вместо среднеотраслевых. Помимо повышения точности оценок, исторические данные сокращают разброс оценок, обусловленный неопределенностью предположений по поводу производительности.

СОВЕТ № 42

Если в настоящий момент у вас еще нет исторических данных, начните собирать их как можно скорее.

Глава 9

СОВЕТ № 43

Оценки уровня задач следует поручать людям, непосредственно занимающимся выполнением соответствующей работы.

СОВЕТ № 44

Создание оценок для лучшего и худшего случаев стимулирует мышление и помогает учесть весь возможный диапазон возможных результатов.

СОВЕТ № 45

Используйте контрольные списки для улучшения индивидуальных оценок. Составляйте и ведите собственные контрольные списки.

СОВЕТ № 46

Сравнивайте оценки с фактическими результатами, чтобы повышать качество своих оценок со временем.

Глава 10

СОВЕТ № 47

Разбейте общую оценку на фрагменты, чтобы воспользоваться действием закона больших чисел: ошибки в сторону завышения и занижения до определенной степени компенсируют друг друга.

СОВЕТ № 48

Используйте обобщенную структуру трудозатрат программных проектов (WBS), чтобы не забыть о типовых операциях.

СОВЕТ № 49

Используйте упрощенную формулу среднеквадратического отклонения для вычисления содержательных оценок лучшего и худшего случаев в проектах, состоящих из 10 и менее задач.

СОВЕТ № 50

Используйте сложную формулу стандартного отклонения для вычисления содержательных сводных оценок лучшего и худшего случаев в проектах, состоящих из 10 и более задач.

СОВЕТ № 51

Не используйте деление диапазона между лучшим и худшим случаями на 6 при вычислении отклонений стандартных оценок. Выбирайте делитель в зависимости от точности диапазонов ваших оценок.

280 Приложение В • Рекомендации по оценке программных проектов

СОВЕТ № 52

Направьте усилия на повышение точности оценок ожидаемого случая. Если отдельные оценки точны, то и их объединение не создаст проблем. С другой стороны, если отдельные оценки неточны, объединение станет возможным лишь после того, как вы найдете способ повысить их точность.

Глава 11

СОВЕТ № 53

Оценивайте новые проекты, сравнивая их с похожими прошлыми проектами. Постарайтесь разбить оценку минимум на пять фрагментов.

СОВЕТ № 54

Не решайте проблему неопределенности смещением оценки. Постарайтесь отразить неопределенность в самой оценке.

СОВЕТ № 55

Используйте нечеткую логику для оценки размера программы в строках кода.

СОВЕТ № 56

Рассматривайте метод стандартных компонентов как средство для получения оценки размера с минимальными усилиями на ранних стадиях проекта.

СОВЕТ № 57

Метод абстрактных рейтингов используется для получения ранней оценки объема работ и сроков проекта, и в основу его закладываются данные того же проекта.

СОВЕТ № 58

Будьте внимательны при вычислении оценок, в которых используются числовые рейтинговые шкалы. Убедитесь в том, что числовые категории на шкале действительно ведут себя как числа, а не как отвлеченные категории вроде «малый», «средний» или «большой».

СОВЕТ № 59

Используйте метод футболки, чтобы помочь не-техническим сторонам проекта принять решения по включению или исключению тех или иных функций в широкой части конуса неопределенности.

СОВЕТ № 60

Используйте опосредованные методы для оценки количества тестовых сценариев, дефектов, страниц документации и любых других показателей, которые трудно оценивать напрямую.

СОВЕТ № 61

Подсчитывайте тот показатель, который проще всего считается и обеспечивает максимальную точность в вашей среде. Соберите калибровочные данные и используйте их для создания оценки, адаптированной к вашей среде.

Глава 13

СОВЕТ № 62

Используйте групповое обсуждение для повышения точности оценки.

СОВЕТ № 63

Используйте широкополосный Дельфийский метод для формирования оценок на ранней стадии проекта, в незнакомых системах, а также тогда, когда в проекте задействовано несколько разнородных дисциплин.

Глава 14**СОВЕТ № 64**

Используйте оценочные программы для логической проверки оценок, созданных ручными методами. Оценки крупных проектов должны в большей степени опираться на коммерческие оценочные программы.

СОВЕТ № 65

Не относитесь к результатам оценочной программы как к божественному откровению. Проверяйте их на соответствие здравому смыслу точно так же, как любую другую оценку.

Глава 15**СОВЕТ № 66**

Используйте несколько альтернативных методов оценки и проанализируйте совпадения или расхождения результатов.

СОВЕТ № 67

Если разные методы оценки приводят к разным результатам, попытайтесь выявить факторы, из-за которых возникают различия. Продолжайте оценивать проект заново до тех пор, пока результаты, полученные разными методами, не будут расходиться в пределах 5 %.

СОВЕТ № 68

Если деловая цель противоречит нескольким сходящимся оценкам, лучше доверьтесь оценкам.

Глава 16**СОВЕТ № 69**

Не оспаривайте результаты оценки, а принимайте их как данность. Изменение результата может производиться только одним способом: изменением входных данных и повторным вычислением.

СОВЕТ № 70

Сначала сконцентрируйтесь на оценке размера. Затем вычислите объем работ, сроки, стоимость и функциональность по оценке размера.

СОВЕТ № 71

Проводите повторную оценку.

СОВЕТ № 72

Переходите от неточных оценок к более точным по мере продвижения работы над проектом.

СОВЕТ № 73

Когда проект будет готов к назначению индивидуальных заданий, переходите на восходящую оценку.

282 Приложение В • Рекомендации по оценке программных проектов

COBET № 74 _____

Если оценка пересчитывается в результате нарушения промежуточных сроков, новая оценка должна базироваться на фактическом ходе проекта, а не на запланированных показателях.

COBET № 75 _____

Представляйте свои оценки в таком виде, чтобы они уточнялись по мере продвижения проекта.

COBET № 76 _____

Сообщайте о своих планах повторного проведения оценки заранее.

Глава 17

COBET № 77 _____

Определите стандартизированную процедуру оценки на уровне организации и применяйте ее на уровне отдельных проектов.

COBET № 78 _____

Координируйте стандартизированную процедуру оценки с процессом SDLC, принятым в вашей организации.

COBET № 79 _____

Анализируйте оценки и процедуры получения оценок в своих проектах; это поможет вам повысить точность оценок и свести к минимуму затраты на их создание.

COBET № 80 _____

Используйте строки программного кода для оценки размеров, но помните об общих ограничениях простых метрик, а также специфических опасностях метрики LOC.

COBET № 81 _____

Воспользуйтесь методом функциональных пунктов, чтобы получить относительно точную оценку размера на ранней стадии проекта.

COBET № 82 _____

Используйте голландский метод для получения приближенной оценки с минимальными затратами на ранней стадии проекта.

COBET № 83 _____

Используйте подсчет элементов GUI для получения приближенной оценки с минимальным объемом работы на ранней стадии проекта.

COBET № 84 _____

При правильной методологии оценка размера становится основой для всех остальных оценок. Размер создаваемой системы является важнейшим фактором, определяющим ее стоимость. Для повышения точности используйте альтернативные оценки со сравнением результатов.

COBET № 85 _____

Используйте оценочные программы, основанные на формальных методах оценки, для более точного вычисления оценки объема работ по имеющейся оценке размера.

СОВЕТ № 86

Используйте среднеотраслевые графики для получения грубой оценки объема работ в широкой части конуса неопределенности. Помните, что в крупных проектах затраты, связанные с применением более мощных методов оценки, быстро окупаются.

СОВЕТ № 87

Используйте метод ISBSG для получения грубой оценки объема работ. Сравните его с другими методами и проанализируйте схождение или расхождение оценок.

СОВЕТ № 88

Не все методы оценки равны. При поиске схождения или расхождения между оценками присваивайте большие весовые коэффициенты методам, дающим более точные результаты.

Глава 20

СОВЕТ № 89

Используйте базовую формулу для ранней оценки срока в средних и крупных проектах.

СОВЕТ № 90

Используйте формулу неформального сравнения с прошлыми проектами для ранней оценки срока в проектах любого размера.

СОВЕТ № 91

Используйте метод оценки первого порядка для получения неточной (но требующей крайне малых усилий) оценки срока на ранней стадии проекта.

СОВЕТ № 92

Не сокращайте оценку срока без увеличения оценки объема работ.

СОВЕТ № 93

Не сокращайте номинальный срок более чем на 25 %. Иначе говоря, не пытайтесь ввести оценки в «зону невозможности».

СОВЕТ № 94

Чтобы снизить стоимость проекта, увеличьте срок и используйте группу меньшей численности.

СОВЕТ № 95

В бизнес-системах среднего размера (от 35 000 до 100 000 строк кода) не рекомендуется использовать группы численностью более 7 человек.

СОВЕТ № 96

Используйте оценку срока для проверки реалистичности своих планов. Для формирования окончательного графика следует применять детальное планирование.

СОВЕТ № 97

Прежде чем искать схождение или расхождение между оценками, исключите из набора данных результаты, полученные слишком общими методами.

284 Приложение В • Рекомендации по оценке программных проектов

СОВЕТ № 98

При распределении объема работы проекта следует учитывать размер проекта, тип проекта, а также операции, заложенные в калибровочных данных, использованных для создания исходной монолитной оценки.

СОВЕТ № 99

При распределении сроков между различными операциями следует учитывать размер проекта, его тип и методологию разработки.

СОВЕТ № 100

Используйте среднеотраслевые или исторические данные для оценки предполагаемого количества дефектов в вашем проекте.

СОВЕТ № 101

Данные эффективности исправления дефектов позволяют оценить количество дефектов, которые будут удалены из программы вашими методами контроля качества перед выпуском окончательной версии.

СОВЕТ № 102

Используйте суммарный ущерб рисков вашего проекта в качестве отправной точки при планировании буферов. Проанализируйте отдельные аспекты конкретных рисков и постарайтесь понять, не должен ли запланированный буфер быть больше или меньше суммарного ущерба.

СОВЕТ № 103

Планирование и оценка — взаимосвязанные темы, а тема планирования слишком обширна, чтобы ее можно было изложить в одной главе книги, посвященной оценке программных проектов. Читайте специализированную литературу по планированию.

СОВЕТ № 104

Документируйте и сообщайте предположения, заложенные в оценке.

СОВЕТ № 105

Вы должны четко понимать, какая неопределенность отражается в представлении оценки: неопределенность самой оценки или же неопределенность, влияющая на вашу возможность выполнения обязательств.

СОВЕТ № 106

Не представляйте ответственным сторонам проекта маловероятные результаты.

СОВЕТ № 107

Подумайте, не заменить ли текстовое представление оценки графическим.

СОВЕТ № 108

Используйте стиль представления оценок, который бы подчеркивал передаваемую вами информацию о точности оценки.

СОВЕТ № 109

Не пытайтесь выражать обязательства в диапазонной форме. Обязательства должны быть более конкретными.

COBET № 110

Поймите, что администраторы настойчивы по своей природе и по должности, и планируйте обсуждение оценок соответствующим образом.

COBET № 111

Учитывайте воздействие внешних факторов. Ваши собеседники должны видеть, что вы понимаете коммерческие требования и их важность.

COBET № 112

Обсуждайте обязательства, но не оценки.

COBET № 113

Расскажите ответственным участникам, не связанным с технической стороной проекта, о принципах эффективной оценки программных проектов.

COBET № 114

Относитесь к обсуждению оценок как к решению проблемы, а не как к переговорам. Поймите, что все ответственные участники проекта находятся на одной стороне. Либо все вместе проигрывают, либо все вместе выигрывают.

COBET № 115

Нападайте на проблему, а не на людей.

COBET № 116

Выдайте как можно больше вариантов планирования, поддерживающих цели вашей организации.

COBET № 117

В процессе совместного решения проблем не принимайте поспешных обязательств, основанных на импровизированных оценках.

COBET № 118

Чтобы выйти из тупика, возникшего в дискуссии, почтите возвращайтесь к вопросу: «Что будет лучше для нашей организации?»

СОВЕТ № 98

— определите, какое количество времени потребуется для выполнения операций, запланированных в контрактной основе, и насколько это соответствует реальной продолжительности проекта.

СОВЕТ № 120

СОВЕТ № 99: ПРИМЕРЫ ПОДЧИНЯЮЩИХ СТАВОК ИХ НИЗШЕГО УРОВНЯ
При оценке ставок между уровнями иерархии оценки должны быть согласованы с соответствующими проектами.

Библиография

СОВЕТ № 100

— Используйте среднесречные или исторические данные для выявления дефектов в вашем проекте.

СОВЕТ № 101 — Данные эффективности исправления дефектов являются важными для выявления дефектов, которые будут удалены из программы различными методами контроля качества перед выполнением.

СОВЕТ № 102 — Оцените количество ошибок в программе, которое было удалено из программы различными методами контроля качества перед выполнением.

Abdel-Hamid, T., and S. Madnick, 1986. «Impact of Schedule Estimation on Software Project Behavior», IEEE Software, 3, 4 (July 1986), pp. 70–75.

Albrecht, Allan J., 1979. «Measuring Application Development Productivity». Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, October 1979: 83–92.

Albrecht, A., and J. Gaffney, «Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation», IEEE Transactions on Software Engineering, SE-9(6), 1983, pp. 639–648.

Armour, Phillip, 2002. «Ten Unmyths of Project Estimation», Communications of the ACM, November 2002, pp. 15–18.

Armstrong, J. Scott, ed., 2001. «Principles of forecasting: A handbook for researchers and practitioners». Boston, MA: Kluwer Academic Publishers.

Annone, Michael, 2005. «Azmi: Sentinel Won't Repeat Mistakes», Federal Computer Week, September 13, 2005.

Associated Press, 2003. «Boston's 'Big Dig' Opens to Public: Tunnel Project is Five Years Behind Schedule, Billions Over Budget». MSNBC.com, December 20, 2003.

Baker, F. Terry, 1972. «Chief Programmer Team Management of Production Programming», IBM Systems Journal, vol. 11, no. 1, 1972, pp. 56–73.

Basili, V.R., and B.T. Perricone, 1984. «Software Errors and Complexity: An Empirical Investigation». Communications of the ACM, v. 27, 1984, pp. 42–52.

Bastani, Farokh, and Sitharama Iyengar, 1987. «The Effect of Data Structures on the Logical Complexity of Programs». Communications of the ACM, vol. 30, no. 3, pp. 250–259.

Beck, Kent, and Martin Fowler, 2001. «Planning Extreme Programming», Boston, MA: Addison-Wesley.

Beck, Kent, 2004. «Extreme Programming Explained: Embrace Change», 2d ed., Reading, MA: Addison-Wesley.

Bentley, Jon, 2000. «Programming Pearls», 2d ed., Reading, MA: Addison-Wesley.

- Boehm, Barry, and Richard Turner, 2004. «Balancing Agility and Discipline: A Guide for the Perplexed», Boston, MA: Addison-Wesley.
- Boehm, Barry, et al., 1995. «Cost Models for Future Software Life Cycle Processes: COCOMO 2.0», «Annals of Software Engineering», Special Volume on Software Process and Product Measurement, J.D. Arthur and S.M. Henry Eds., Amsterdam, Netherlands: J.C Baltzer AG, Science Publishers.
- Boehm, Barry W., and Philip N. Papaccio, 1988. «Understanding and Controlling Software Costs», «IEEE Transactions of Software Engineering», v. 14, no. 10, October 1988, pp. 1462–1477.
- Boehm, Barry W., 1981. «Software Engineering Economics», Englewood Cliffs, NJ: Prentice Hall.
- Boehm, Barry W., 1987b. «Industrial software metrics top 10 list», IEEE Software, vol. 4, no.9 (September 1987), pp. 84–85
- Boehm, Barry W., T.E. Gray, and T. Seewaldt, 1984. «Prototyping versus specifying: A multiproject experiment», «IEEE Transactions on Software Engineering», vol. SE-10 (May 1984), pp. 290–303.
- Boehm, Barry, et al., 2000. «Software Cost Estimation with Cocomo II», Reading, MA: Addison-Wesley.
- Brooks, Frederick P., Jr., 1975. «The Mythical Man-Month», Reading MA: Addison-Wesley.
- Brooks, Frederick P., Jr., 1995. «The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Ed)», Reading MA: Addison-Wesley.
- Car and Driver, 2004. «2005 Charting the Changes – BMW», «Car and Driver», October 2004.
- Card, David N., 1987. «A Software Technology Evaluation Program», «Information and Software Technology», v. 29, no. 6, July/August 1987, pp. 291–300.
- Cockburn, Alistair, 2001. «Agile Software Development», Boston, MA: Addison-Wesley.
- Cohn, Mike, 2005. «Agile Estimating and Planning», Upper Saddle River, NJ: Prentice Hall PTR.
- Conte, S.D., H.E. Dunsmore, and V.Y. Shen, 1986. «Software Engineering Metrics and Models», Menlo Park, CA: Benjamin/Cummings.
- Coombs, Paul, 2003. «IT Project Estimation: A Practical Guide to the Costing of Software», Cambridge, United Kingdom: Cambridge University Press.
- Cooper, Robert G., 2001. «Winning at New Products: Accelerating the Process from Idea to Launch», New York, NY: Perseus Books Group.
- Costello, Scott H., 1984. «Software engineering under deadline pressure», ACM Sigsoft Software Engineering Notes, 9:5 October 1984, pp. 15–19.
- Crosstalk, June 2002.
- Crosstalk, April 2002.
- Curtis, Bill, 1981. «Substantiating Programmer Variability», «Proceedings of the IEEE», vol. 69, no.7, p. 846.
- Curtis, Bill, 1981. «Software Psychology: The Need for an Interdisciplinary Program», «Proceedings of the IEEE», vol. 74, no.8 (August 1986), pp. 1092-1106.
- Cusumano, Michael, and Richard W. Selby, 1995. «Microsoft Secrets», New York, NY: The Free Press.

- Davis, John Stephen, and Richard J. LeBlanc, 1998. «A Study of the Applicability of Complexity Measures», *«IEEE Transactions on Software Engineering»*, v. 14, no. 9, September 1988, pp. 1366–1372.
- DeMarco, Tom, and Timothy Lister, 1985. «Programmer Performance and the Effects of the Workplace», *«Proceedings of the 8th International Conference on Software Engineering»*, August 1985, pp. 268–272.
- DeMarco, Tom, and Timothy Lister, 1999. «Peopleware: Productive Projects and Teams», 2d ed., New York, NY: Dorset House.
- DeMarco, Tom, and Timothy Lister, 2003. «Waltzing with Bears: Managing Risks on Software Projects», New York, NY: Dorset House.
- DeMarco, Tom, 1982. «Controlling Software Projects», New York, NY: Yourdon Press.
- Evangelist, Michael, 1984. «Program Complexity and Programming Style», *«Proc. 1st. Int. Conf. Data Engineering»*, New York, NY: IEEE Computer Society Press, pp. 531–541.
- Fenton, Norman E., and Shari Lawrence Pfleeger, 1997. «Software Metrics: A Rigorous and Practical Approach», Boston, MA: PWS Publishing Company.
- Fisher, Roger, William Ury, and Bruce Patton, 1991. «Getting to Yes», 2d ed., New York, NY: Penguin Books.
- Gaffney, John E., Jr., and Richard Werling, 1991. «Estimating Software Size from Counts of Externals, A Generalization of Function Points», Herndon, VA: Software Productivity Consortium document number SPC-91094-N.
- Garmus, David, and David Herron, 2001. «Function Point Analysis: Measurement Practices for Successful Software Projects», Boston, MA: Addison-Wesley.
- Glib, Tom, 1988. «Principles of Software Engineering Management», Wokingham, England: Addison-Wesley.
- Glib, Tom, 2005. «Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering using Planguage», Amsterdam, Netherlands: Elsevier.
- Glass, Robert L., 1994. «IS Field: Stress Up, Satisfaction Down», *«Software Practitioner»*, Nov. 1994, pp. 1,3.
- Goldratt, Eliyahu M., 1997. «Critical Chain», Great Barrington, MA: The North River Press.
- Gorla, N., A.C. Benander, and B.A. Benander, 1990. «Debugging Effort Estimation Using Software Metrics», *«IEEE Transactions on Software Engineering»*, v. 16, no. 2, February 1990, pp. 223–231.
- Gotterbarn, Don, Keith Miller, and Simon Rogerson. «Computer Society and ACM Approve Software Engineering Code of Ethics», *«IEEE Computer»*, October 1999, pp. 84–88. www.computer.org/computer/code-of-ethics.pdf.
- Grady, Robert B., 1992. «Practical Software Metrics for Project Management And Process Improvement», Englewood Cliffs, NJ: Prentice Hall PTR.
- Grady, Robert B., and Deborah L. Caswell, 1987. «Software Metrics: Establishing a Company-Wide Program», Englewood Cliffs, NJ: Prentice Hall.
- Gross, Neil, et al. «Software Hell», *«Business Week»*, Nov. 6, 1999, p. 104.

- Harvey, N., 2001. «Improving Judgement in Forecasting» in «Principles of forecasting: A handbook for researchers and practitioners», Ed., J.S. Armstrong, Boston, MA: Kluwer Academic Publishers, pp. 59–80.
- Heemstra, F.J., and R.J.Custers, 1991. «Function Point Analysis: Evaluation of a Software Cost Estimation Model». «European Journal of Information Systems» 1(4): 223–237.
- Heemstra, F.J., W.J.A.Siskens, and H. van der Stelt, 1989. «Kostenbeheersing Bij Automatiseringsprojecten: Een Empirisch Onderzoek», «Infomatie», vol. 31, no.1 (1989) cited in (Putnam and Myers 2003).
- Henry, Sallie, and Dennis Kafura, 1984. «The Evaluation of Software Systems' Structure Using Quantitative Software Metrics», «Software – Practice and Experience», vol. 14, no. 6 (June 1984), pp. 561–73.
- Herbsleb, James, et al., «Benefits of CMM Based Software Process Improvement: Initial Results», Pittsburgh: Software Engineering Institute, Document CMU/SEI-94-TR-13, August 1994.
- Hihn, J., and H. Habib-Agahi, 1991. «Cost Estimation of Software Intensive Projects: a Survey of Current Practices», International Conference on Software Engineering, IEEE Computer Society Press, Los Alamitos, CA: 276–287.
- Host, M., and C. Wohlin, 1998. «An Experimental Study of Individual Subjective Effort Estimations and Combinations of the Estimates», International Conference on Software Engineering, Kyoto, Japan, IEEE Computer Society, Los Alamitos, CA: 332–339.
- Humphrey, Watts S.. 1995. «A Discipline for Software Engineering», Reading, MA: Addison-Wesley.
- Iansiti, Marco, 1994. «Microsoft Corporation: Office Business Unit», Harvard Business School Case Study, 9–691–033, Revised May 31, 1994, Boston, MA: Harvard Business School.
- «IEEE Software», Nov/Dec 2001 Issue.
- IFPUG, веб-сайт: www.ifpug.org.
- ISBSG 2001. «Practical Project Estimation: A Toolkit for Estimating Software Development Effort and Duration», Australia: International Software Benchmarking Standards Group, March 2001.
- ISBSG 2005. «Practical Project Estimation, 2nd Edition: A Toolkit for Estimating Software Development Effort and Duration», Australia: International Software Benchmarking Standards Group, February 2005.
- ISO/IEC 20926:2003. «Software Engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting practice manual». International Organization for Standardization, 2003.
- Jacobson, Ivar, Grady Booch, James Rumbaugh, 1999. «The Unified Software Development Process», Reading, MA: Addison-Wesley.
- Jones, Capers, 1996. «Software Defect-Removal Efficiency», «IEEE Computer», April 1996.
- Jones, Capers, 2005. «Software Engineering: The State of the Art in 2005», Version 5, Software Productivity Research Whitepaper, February 11, 2005.
- Jones, Capers, 1986. «Programming Productivity», New York, NY: McGraw-Hill.

290 Библиография

- Jones, Capers, 1994. «Assessment and Control of Software Risks», Englewood Cliffs, NJ: Yourdon Press.
- Jones, Capers, 1995c. «Determining Software Schedules», «IEEE Computer», February 1994, pp. 73–75.
- Jones, Capers, 1997. «Applied Software Measurement: Assuring Productivity and Quality», 2d ed., New York, NY: McGraw-Hill.
- Jones, Capers, 1998. «Estimating Software Costs», New York NY: McGraw-Hill.
- Jones, Capers, 2000. «Software Assessments, Benchmarks, and Best Practices». Reading, MA: Addison-Wesley.
- Jørgensen M., 2002. «A Review of Studies on Expert Estimation on Software Development Effort».
- Jørgensen M., and D.I.K. Sjøberg, 2002. «The Impact of Customer Expectation on Software Development Effort Estimates», «International Journal of Project Management».
- Josephs, R., and E.D. Hahn, 1995. «Bias and Accuracy in Estimates of Task Duration», «Organizational Behavior and Human Decision Process», 61(2): 202–213.
- Kemerer, C.F., 1987. «An Empirical Validation of Software Cost Estimation Models», «Communications of the ACM», 30(5), 1987, pp. 416–429.
- Kemerer, Chris, and Benjamin Porter, 1992. «Improving the Reliability of Function Point Measurement: An Empirical Study», «IEEE Transactions on Software Engineering», vol. 18, no. 11, November 1992, Page 1011.
- Kitchenham, B., S.L. Pfleeger, B. McCall, and S. Eagan, 2002. «A Case Study of Maintenance Estimation Accuracy», «Journal of Systems and Software».
- Knorr, Eric, 2005. «Anatomy of an IT Disaster: How the FBI Blew It», InfoWorld, March 21, 2005.
- Krasner, Jerry, 2003. «Embedded Software Development Issues and Challenges», Embedded Market Forecasters whitepaper, www.embeddedforecast.com.
- Lais, Sami, 2003. «Watch your step: Can major efforts avoid more slipups?», «Government Computer News», vol. 22, no. 34, 12/15/03.
- Laranjera, Luiz, 1990. «Software Size Estimation on Object-Oriented Systems», «IEEE Transactions on Software Engineering», May 1990.
- Larsen, Richard J., and Morris L. Marx, 2001. «An Introduction to Mathematical Statistics and Its Applications», 3rd ed., Upper Saddle River, NJ: Prentice Hall.
- Lawlis, Dr. Patricia K., Capt. Robert M. Flowe, and Capt. James B. Thordahl, 1995. «A Correlational Study of the CMM and Software Development Performance», Crosstalk, September 1995.
- Lederer, Albert L., and Jayesh Prasad, 1992. «Nine Management Guidelines for Better Cost Estimating», «Communications of the ACM», February 1992, pp. 51–59.
- Libby, R., and R.K. Blashfield, 1978. «Performance of a Composite as a Function of the Number of Judges», «Organizational Behaviour and Human Performance» 21(2): 121–129.
- Lim, J.S., and M. O'Connor, 1996. «Judgemental Forecasting With Time Series and Causal Information», «International Journal of Forecasting» 12(1): 139–153.
- McCabe, Tom, 1976. «A Complexity Measure», «IEEE Transactions on Software Engineering», Volume SE-2, Number 12 (December 1976), pp. 308–320.

- McConnell, Steve, 1993. «Code Complete», Redmond, WA: Microsoft Press, 1993.
- McConnell, Steve, 1996. «Rapid Development», Redmond: WA: Microsoft Press.
- McConnell, Steve, 1998. «Software Project Survival Guide», Redmond, WA: Microsoft Press.
- McConnell, Steve, 2000. «Sitting on the Suitcase», «IEEE Software», May/June 2000.
- McConnell, Steve, 2002. «Real Quality for Real Engineers», «IEEE Software», March/April, 2002.
- McConnell, Steve, 2004a. «Code Complete», 2d ed., Redmond, WA: Microsoft Press.
- McConnell, Steve, 2004b. «Professional Software Development», Boston, MA: Addison-Wesley.
- McGarry, John, et al., 2002. «Practical Software Measurement: Objective Information for Decision Makers», Boston, MA: Addison-Wesley.
- McGraw, Gary, 2003. «From the Ground Up: The DIMACS Software Security Workshop», «IEEE Security & Privacy», March/April 2003. Volume 1, Number 2, pp. 59–66.
- Metzger, Philip W., 1981. «Managing a Programming Project», 2d ed., Englewood Cliffs, NJ: Prentice Hall.
- Mills, Harlan D., 1983. «Software Productivity», Boston, MA: Little, Brown, pp. 71–81.
- Mohanty, S.N., 1981. «Software Cost Estimation: Present and Future», «Software – Practice and Experience», 11, pp. 103–121.
- Mosemann, Lloyd K., II, 2002. «Did We Lose Our Religion?» Crosstalk, August 2002, pp. 22–25.
- NASA SEL, 1990. «Manager's Handbook for Software Development, Revision 1». Document number SEL-84-101. Greenbelt, MD: Goddard Space Flight Center, NASA.
- NASA SEL, 1991. «Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules», SEL-91-101, February 1991.
- NASA SEL, 1995. «Software Management Guidebook», Document Number NASA-GB-001-94, Greenbelt, M: Goddard Space Flight Center, NASA, June 1995.
- NASA, «ISD Wideband Delphi Estimation», Number 580-PROGRAMMER-016-01, September 1, 2004, <http://software.gsfc.nasa.gov/AssetsApproved/PA1.2.1.2.pdf>.
- Niessink, F., and H. van Vliet, 1997. «Predicting Maintenance Effort With Function Points», International conference on software maintenance, Bari, Italy, IEEE Computer Society, Los Alamitos, CA, pp. 32–39.
- Park, R.E., 1996. «A Manager's Checklist for Validating Software Cost and Schedule Estimates», American Programmer 9(6), pp. 30–35.
- Paynter, J., 1996. «Project Estimation Using Screenflow Engineering», International Conference on Software Engineering: Education and Practice, Dunedin, New Zealand, IEEE Computer Society Press, Los Alamitos, CA, pp.150–159.
- Pehrson, Ron J., 1996. «Software Development for the Boeing 777», CrossTalk, January 1996.
- Pfleeger, Shari Lawrence, Felicia Wu, and Rosalind Lews, 2005. «Software Cost Estimation and Sizing Methods», Santa Monica, CA: The Rand Corporation.

- Pietrasanta, Alfred M., 1990. «Alfred M. Pietrasanta on Improving the Software Process», «Software Engineering: Tools, Techniques, Practices», vol. 1, no. 1 (May/June 1990), pp.29–34.
- Pitterman, Bill, 2000. «Telcordia Technologies: The Journey to High Maturity», «IEEE Software», July 2000, pp. 89–96.
- Prechelt, Lutz, 2000. «An Empirical Comparison of Seven Programming Language», «IEEE Computer», October 2000, pp. 23–29.
- Putnam, Lawrence H., and Ware Myers, 1992. «Measures for Excellence: Reliable Software On Time, Within Budget». Englewood Cliffs, NJ: Yourdon Press.
- Putnam, Lawrence H., and Ware Myers, 1997. «Industrial Strength Software: Effective Management Using Measurement», Washington, DC: IEEE Computer Society Press.
- Putnam, Lawrence H., and Ware Myers, 1999. «Get the Estimate Right», «American Programmer», July 1999, pp. 4–12.
- Putnam, Lawrence H., and Ware Myers, 2003. «Five Core Metrics», New York, NY: Dorset House.
- Reifer, Donald J., 2002. «Estimating Web Development Costs: There Are Differences», CrossTalk, June 2002, pp. 13–17.
- Roetzheim, William H., 1988. «Structured Computer Project Management», Englewood Cliffs, NJ: Prentice Hall.
- Rule, Grant, 1998. From Stutzke.
- Rule, Grant, 2000. «Bees and the Art of Estimating». «IEEE Software», November/December 2000.
- Sackman, H., Erikson, W.J., Grant, E.E., 1968. «Exploratory Experimental Studies Comparing Online and Offline Programming Performance». «Communications of the ACM», v. 11, no. 1, January 1968, pp. 3–11.
- Sanchez, Roberto, 1998. «UW Learns a Lesson», «Seattle Times», April 12, 1998, page B1.
- Schlender, Brenton, 1989. «How to Break the Software Logjam», «Fortune», September 25, 1989.
- Schneider, Geri, and Jason P. Winters, 2001. «Applying Use Cases», 2d ed., Boston, MA: Addison Wesley Longman.
- Schwaber, Ken, and Mike Beedle, 2002. «Agile Software Development with Scrum», Englewood Cliffs, NJ: Prentice Hall.
- Sheppard, S.B., et al., 1978. «Predicting Programmers' Ability to Modify Software», TR 78-388100-3, General Electric Company, May 1978.
- Sheppard, S.B., et al., 1979. «Modern Coding Practices and Programmer Performance», «IEEE Computer», no. 12, Dec 1979, pp. 41–49.
- Shull, et al., 2002. «What We Have Learned About Fighting Defects», «Proceedings, Metrics 2002», IEEE; pp. 249–258.
- Smith, John, 1999. «The Estimation of Effort Based on Use Cases», Rational Software Whitepaper TP-171, October 1999.
- Standish Group, The, 1994. «Charting the Seas of Information Technology», Dennis, MA: The Standish Group.
- Stutzke, Richard D., 2005. «Estimating Software-Intensive Systems», Upper Saddle River, NJ: Addison Wesley.

- Symons, Charles, 1991. «Software Sizing and Estimating: Mk II FPA (Function Point Analysis)», Chichester: John Wiley & Sons.
- The Age, 2005. «Waiter, there is a bug in my Prius», «The Age», www.theage.com.au, May 25, 2005.
- «The American Heritage Dictionary», Second College Edition, 1985.
- «The Irish Times», 2005. «HSE to suspend roll-out of 150m computer system», October 4, 2005.
- Tockey, Steve, 2005, «Return on Software», Boston, MA: Addison-Wesley.
- Todd, P., and I. Benbasat, 2000. «Inducing Compensatory Information Processing Through Decision Aids That Facilitate Effort Reduction: an Experimental Assessment», «Journal of Behavioural Decision Making», 13(1): pp. 91–106.
- University of Southern California, «Cocomo II Model Definition Manual», version 1.4, undated (circa 1997).
- Valett, J., and F.E. McGarry, 1989. «A Summary of Software Measurement Experiences in the Software Engineering Laboratory», «Journal of Systems and Software», 9(2), pp. 137–148.
- Van Genuchten, Michael, 1991. «Why Is Software Late? An Empirical Study of Reasons for Delay in Software Development», «IEEE Transactions on Software Engineering» SE-17, no. 6 (June), pp. 582–590.
- Vu, John, 2004. «Lessons Learned in Process Improvement», SEPG 2004.
- Walston, C.E., and C.P. Felix, 1977. «A Method of Programming Measurement and Estimation», «IBM Systems Journal», v. 16, no. 1, 1977, pp. 54–73.
- Ward, William T., 1989b. «Software Defect Prevention Using McCabe's Complexity Metric», «Hewlett Packard Journal», April 1989, pp. 64–68.
- Weinberg, Gerald M., and Edward L. Schulman, 1974. «Goals and Performance in Computer Programming», vol. 16, no. 1, pp. 70–77.
- Weinberg, Gerald M., 1994. «Quality Software Management, Vol. 3, Congruent Action», New York, NY: Dorset House.
- Weyuker, Elaine J., 1988. «Evaluating Complexity Measures», «IEEE Transactions on Software Engineering», v. 14, no. 9, September 1988, pp. 1357–1365.
- Wheeler, David A., 2001. «More than a Gigabuck: Estimating GNU/Linux's Size», <http://www.dwheeler.com/sloc>.
- Wiegers, Karl, 2000. «Stop Promising Miracles», «Software Development», February 2000.
- Wiegers, Karl, 2003. «Software Requirements: Thorny Issues and Practical Advice», Redmond, WA: Microsoft Press.
- Withers, Bud, 1999. «Take Me Out to Ballpark», «Seattle Times», July 11, 1999, pp. S3–S4.
- Zultner, Richard E., 1999. «Project Estimation with Critical Chain: Third-Generation Risk Management», «American Programmer», July 1999, pp. 4–12.

- ## Алфавитный указатель
- A**
- Angel (Analogy Software Tool), 169
- C**
- Chaos Report (Standish Group), 42
 - Cocomo II, модель, 64, 79, 169
 - Construx Estimate, программа, 76, 169
 - Costar, программа, 169
- E**
- Estimate Express, программа, 169
- K**
- Knowledge PLAN, программа, 169
- P**
- PERT, метод, 118
 - Price-S, программа, 169
- R**
- RUP, процесс, 92
- S**
- Scrum, стиль разработки, 92
 - SEER, программа, 169
 - SLIM-Estimate, программа, 169
- W**
- WBS, 126
- Г**
- голландский метод, 207
 - групповое обсуждение оценок, 156
- абстрактные рейтинги, 149**
административная поддержка, 249
аналогия, метод оценки, 137
арбитраж, 166
базовая формула для вычисления срока, 224
бизнес-логика, 141
больших чисел, закон, 124, 145
буферы, 248
бюджет, точность, 46
бюджетное планирование сроков, 263
величина относительной ошибки, 119
внешние входные элементы, 203
внешние выходные элементы, 203
внешние запросы, 203
внешние интерфейсные файлы, 204
внешние политические ограничения, 263
внутренние версии, 31
внутренние логические файлы, 203
восходящие оценки, 122
вычислительные методы, 98

Д

данные
 исторические, 97, 102
 калибровка, 96, 102
 проектные, 102
 среднеотраслевые, 102
 счетные методы, 97
 декомпозиция, 122
 Дельфийский метод, 158
 дефекты, 106
 диапазоны, 116
 документирование процедуры оценки, 186
 достоверность, 27

З

закон больших чисел, 124, 145
 закон Паркинсона, 40
 зона невозможности, 230

И

идеальный объем работы, 242
 издержки масштаба, 73, 86
 импровизированные оценки, 49
 интерпретируемые языки, 81
 исторические данные, 102
 итеративная разработка, 57, 91

К

календарные сроки
 оценка, 224
 предсказуемость, 47
 калибровка, 102, 109
 исторические данные, 109
 оценочные программы, 164
 качество программ, 46
 квалифицированные языки, 81
 контроль качества, 237
 контрольные списки задач, 119
 конус неопределенности, 52

Л

лучший случай, сценарий, 27, 54, 116

М

масштаб, издержки, 86
 метод оценки первого порядка, 227

методы оценки, 90

альтернативные, 170
 выбор, 90
 метод футболки, 152
 опосредованные, 143
 по аналогии, 137
 счетные, 96

модели оценки, 79

Монте-Карло, моделирование, 164

Н

наиболее вероятный случай, оценка, 118
 недооценка, 40
 незнакомая среда, 250
 необоснованный оптимизм, 63, 128
 неопределенность, выражение, 254
 непредвиденные события, 31
 нечеткая логика, 144, 208

О

обсуждение оценок, 156, 178, 181
 объективные критерии, 270
 объем работы
 вычисление, 213
 и сроки, 231
 оценка, 211
 программы, 164
 сбор данных, 107
 объем работы, идеальный, 242
 обязательства, 23, 142, 182, 270
 опосредованные методы, 143
 оптимизм, необоснованный, 63, 128
 отладка, 245
 относительная ошибка, 119
 оценка стоимости, 243
 оценки

определение, 22
 пересмотр, 183
 планы, 23
 хорошие, 28
 оценочные программы, 163
 калибровка, 168
 список, 168
 ошибка оценки, 51
 импровизация, 66
 конус неопределенности, 52
 необоснованный оптимизм, 63
 политика, 261
 пропущенные операции, 61

П

Паркинсона, закон, 40
переговоры, 263
переоценка, 40
планирование проектов
оценка, 23
параметры, 236
планы, 142
полезность оценки, 32
плюс/минус, квалификаторы, 254
последовательная разработка, 91
поэтапная выдача, 92
предсказуемость, 47
принципиальные переговоры, 266
пропущенные операции, 61
процесс поэтапного контроля, 186
процесс разработки
контрольные точки, 187
стадии, 92
прямые затраты, 244

Р

размер проекта, 43, 71
распределение объема работ, 238
регулирующие факторы, Cocomo II, 81
резервный буфер, 248
риски, квантификация, 254
рост требований, 59

С

сверхурочные, компенсация, 243
сводные оценки лучшего/худшего
случая, 133
события, непредвиденные, 31
сокращение сроков, 228
среднеотраслевые графики объема, 215
стандартизированная процедура, пример, 192
стиль разработки, 91
стоимость
оценка, 243
строки программного кода, 71
структурированные экспертные оценки, 115
студенческий синдром, 40
субъективность, 64
схождение оценок, 170

счетные методы, 96

функциональные пункты, 203

Т

точечные оценки, 25, 116
точность оценок
выражение неопределенности, 254
исторические данные, 103
преимущества, 45
четкость, 68

У

управление проектом, 30
устранение дефектов, 244
уточнение оценок, 179

Ф

Фибоначчи, последовательность, 151
формирование оценки, 175
функциональность

дефекты, 244
классификация, 145

функциональные пункты, 203

Х

хаотические процессы разработки, 58
хорошая оценка, определение, 28
худший случай, сценарий, 27, 116

Ц

цели, 22

Ч

четкость, 68
что-если, анализ, 166

Ш

ширина диапазона оценки, 37
широкополосный Дельфийский
метод, 158

Э

эволюционное макетирование, 91
экспертные суждения, 100, 114
экстремальное программирование, 92
эффективность устранения дефектов, 245

Стив Макконнелл

Сколько стоит программный проект

Заведующий редакцией
Ведущий редактор
Научный редактор
Технический редактор
Литературный редактор
Художник
Корректор
Верстка

*A. Кривцов
A. Пасечник
E. Матвеев
C. Романов
A. Пасечник
L. Адуевская
B. Листова
P. Гришанов*

**Совместный проект издательства «Русская Редакция»
и издательства «Питер»**

 **РУССКАЯ РЕДАКЦИЯ**

 **ПИТЕР®**

Подписано в печать 17.01.07. Формат 70×100/16. Усл. п. л. 24,51. Тираж 2000. Заказ 893
ООО «Питер Пресс», 198206, Санкт-Петербург, Петергофское шоссе, д. 73, лит. А29.

Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2; 95 3005 — литература учебная.
Отпечатано с готовых диапозитов в ОАО «Техническая книга».
190005, Санкт-Петербург, Измайловский пр., д. 29.

MICROSOFT PRESS — ДЛЯ ПРОФЕССИОНАЛОВ

склюзивные издания.

значеные для программистов

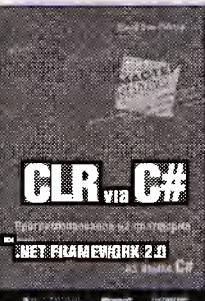
емных администраторов, созданы гуру

информационных технологий и призваны

ответить на вопросы, редко освещаемые

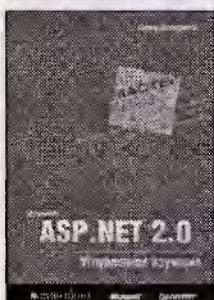
в современной компьютерной литературе.

(Совместный проект издательства «Русская Редакция»
и издательства «Питер»)



Джеффри Рихтер
CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#. Мастер-класс. 656 стр., 2007 г.

Эта книга — подробное описание внутреннего устройства и функционирования общеязыковой исполняющей среды (CLR) Microsoft .NET Framework версии 2.0. В ней раскрыта система типов .NET Framework и разъяснены способы управления ими. Представлены концепции программирования с широким использованием библиотеки FCL, относящиеся ко всем языкам, ориентированным на работу с .NET Framework. Особое внимание уделено обобщениям, управлению асинхронными операциями и синхронизации потоков. Книга ориентирована на разработчиков любых видов приложений на платформе с .NET Framework.



Дино Эспозито
Microsoft ASP.NET 2.0. Углубленное изучение.
Мастер-класс. 592 стр., 2006 г.

Эта книга — подробное руководство для профессионалов-разработчиков приложений ASP.NET. В ней раскрыты тонкости внутреннего функционирования исполняющей среды ASP.NET 2.0 и возможности ее конфигурирования, детально описан процесс выполнения приложений и средства, позволяющие сделать их надежными, эффективными и хорошо защищенными. Вы узнаете, как создавать пользовательские элементы управления, освоите новые навигационные средства ASP.NET 2.0 и научитесь формировать оптимальное представление данных с помощью новых элементов управления.



Чарльз Петцольд
Программирование с использованием Microsoft Windows Forms. Мастер-класс. 432 стр., 2006 г.

В этой книге подробно рассказывается о создании программ для Microsoft Windows с использованием языка C# и библиотеки классов Windows Forms, входящей в Microsoft .NET Framework 2.0. Вы научитесь создавать новые нестандартные и комбинировать существующие элементы управления, а также разрабатывать панели инструментов, меню и строки состояния, используя, появившиеся в .NET Framework 2.0, новые элементы управления, узнаете о новом механизме динамического размещения элементов управления на форме и о привязке элементов управления к данным.



Стив Макконнелл
Совершенный код. Мастер-класс. 896 стр., 2005 г.

Каков бы ни был ваш профессиональный уровень, с какими бы средствами разработки вы ни работали, какова бы ни была сложность вашего проекта, в этой книге вы найдете нужную информацию, она заставит вас размышлять и поможет создать совершенный код.

«Это исчерпывающее исследование тактических аспектов создания хорошо спроектированных программ. Книга Макконнелла охватывает такие разные темы, как архитектура, стандарты кодирования, тестирование, интеграция и суть разработки ПО» (Гари Буч, автор книги «Object Solutions»).

Microsoft®
www.microsoft.com

РУССКАЯ РЕДАКЦИЯ
www.rusedit.com

ПИТЕР®
www.piter.com

ИГРЫ

**Ежемесячный
научно-популярный
компьютерный журнал**

Издаётся с апреля 1994 года

Тираж 53 200 экземпляров
объём не менее 128 страниц.
Распространяется по России
и странам СНГ
Приложение на CD-ROM и DVD

www.gamers.ru

Журнал является одним из лидеров среди российских
компьютерных периодических изданий

КЛУБ ПРОФЕССИОНАЛ

Основанный Издательским домом «Питер» в 1997 году, книжный клуб «Профессионал» собирает в своих рядах знатоков своего дела, которых объединяет тяга к знаниям и любовь к книгам. Для членов клуба проводятся различные мероприятия и, разумеется, предусмотрены привилегии.

Привилегии для членов клуба:

- карта члена «Клуба Профессионал»;
- бесплатное получение клубного издания – журнала «Клуб Профессионал»;
- дисконтная скидка на всю приобретаемую литературу в размере 10% или 15%;
- бесплатная курьерская доставка заказов по Москве и Санкт-Петербургу;
- участие во всех акциях Издательского дома «Питер» в розничной сети на льготных условиях.

Как вступить в клуб?

Для вступления в «Клуб Профессионал» вам необходимо:

- совершить покупку на сайте www.piter.com или в фирменном магазине Издательского дома «Питер» на сумму от **800** рублей без учета почтовых расходов или стоимости курьерской доставки;
- ознакомиться с условиями получения карты и сохранения скидок;
- выразить свое согласие вступить в дисконтный клуб, отправив письмо на адрес: postbook@piter.com;
- заполнить анкету члена клуба (зарегистрированным на нашем сайте этого делать не надо).

Правила для членов «Клуба Профессионал»:

- для продления членства в клубе и получения **скидки 10%**, в течение каждого из **шести месяцев** нужно совершать покупки на общую сумму от **800** до **1500** рублей, без учета почтовых расходов или стоимости курьерской доставки;
- Если же за указанный период вы выкупите товара на сумму от **1501** рублей, скидка будет увеличена до **15%** от розничной цены издательства.

Заказать наши книги вы можете любым удобным для вас способом:

- по телефону: (812) 703-73-74;
- по электронной почте: postbook@piter.com;
- на нашем сайте: www.piter.com;
- по почте: 197198, Санкт-Петербург, а/я 619 ЗАО «Питер Пост».

При оформлении заказа укажите:

- ваш регистрационный номер (если вы являетесь членом клуба), фамилию, имя, отчество, телефон, факс, e-mail;
- почтовый индекс, регион, район, населенный пункт, улицу, дом, корпус, квартиру;
- название книги, автора, количество заказываемых экземпляров.

ИЗДАТЕЛЬСТВО
ПИТЕР[®]
WWW.PITER.COM



Нет времени ходить по магазинам?

наберите:

www.piter.com

Здесь вы найдете:

Все книги издательства сразу

Новые книги — в момент выхода из типографии

Информацию о книге — отзывы, рецензии, отрывки

Старые книги — в библиотеке и на CD

**И наконец, вы нигде не купите
наши книги дешевле!**

КНИГА-ПОЧТОЙ



ЗАКАЗАТЬ КНИГИ ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»
МОЖНО ЛЮБЫМ УДОБНЫМ ДЛЯ ВАС СПОСОБОМ:

- по телефону: (812) 703-73-74;
- по электронному адресу: postbook@piter.com;
- на нашем сервере: www.piter.com;
- по почте: 197198, Санкт-Петербург, а/я 619,
ЗАО «Питер Пост».

Как воспользоваться услугами?

**ВЫ МОЖЕТЕ ВЫБРАТЬ ОДИН ИЗ ДВУХ СПОСОБОВ ДОСТАВКИ
И ОПЛАТЫ ИЗДАНИЙ:**

- Наложенным платежом с оплатой заказа при получении посылки на ближайшем почтовом отделении. Цены на издания приведены ориентировочно и включают в себя стоимость пересылки по почте (но без учета авиатарифа). Книги будут высланы нашей службой «Книга-почтой» в течение двух недель после получения заказа или выхода книги из печати.
- Оплата наличными при курьерской доставке (**для жителей Москвы и Санкт-Петербурга**). Курьер доставит заказ по указанному адресу в удобное для вас время в течение трех дней.

ПРИ ОФОРМЛЕНИИ ЗАКАЗА УКАЖИТЕ:

- фамилию, имя, отчество, телефон, факс, e-mail;
- почтовый индекс, регион, район, населенный пункт, улицу, дом, корпус, квартиру;
- название книги, автора, код, количество заказываемых экземпляров.

Вы можете заказать бесплатный журнал «Клуб Профессионал»

издательский дом
ПИТЕР®
WWW.PITER.COM



СПЕЦИАЛИСТАМ
КНИЖНОГО БИЗНЕСА!

ПРЕДСТАВИТЕЛЬСТВА ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»

предлагают эксклюзивный ассортимент компьютерной, медицинской,
психологической, экономической и популярной литературы

РОССИЯ

Москва м. «Павелецкая», 1-й Кожевнический переулок, д.10; тел./факс (495) 234-38-15,
255-70-67, 255-70-68; e-mail: sales@piter.msk.ru

Санкт-Петербург м. «Выборгская», Б. Сампсониевский пр., д. 29а;
тел./факс (812) 703-73-73, 703-73-72; e-mail: sales@piter.com

Воронеж Ленинский пр., д. 169; тел./факс (4732) 39-43-62, 39-61-70;
e-mail: pitervn@comch.ru

Екатеринбург ул. 8 Марта, д. 267б, офис 202;
тел./факс (343) 256-34-37, 256-34-28; e-mail: piter-ural@isnet.ru

Нижний Новгород ул. Совхозная, д. 13; тел. (8312) 41-27-31;
e-mail: office@nnov.piter.com

Новосибирск ул. Немировича-Данченко, д. 104, офис 502;
тел./факс (383) 211-93-18, 211-27-18, 314-23-89; e-mail: office@nsk.piter.com

Ростов-на-Дону ул. Ульяновская, д. 26; тел. (8632) 69-91-22, 69-91-30;
e-mail: piter-ug@rostov.piter.com

Самара ул. Молодогвардейская, д. 33, литер А2, офис 225; тел. (846) 277-89-79;
e-mail: pitvolga@samtel.ru

УКРАИНА

Харьков ул. Сузdalские ряды, д. 12, офис 10-11; тел./факс (1038067) 545-55-64,
(1038057) 751-10-02; e-mail: piter@kharkov.piter.com

Киев пр. Московский, д. 6, кор. 1, офис 33; тел./факс (1038044) 490-35-68, 490-35-69;
e-mail: office@kiev.piter.com

БЕЛАРУСЬ

Минск ул. Притыцкого, д. 34, офис 2; тел./факс (1037517) 201-48-79, 201-48-81;
e-mail: office@minsk.piter.com



Ищем зарубежных партнеров или посредников, имеющих выход на зарубежный рынок.

Телефон для связи: (812) 703-73-73.

E-mail: grigorjan@piter.com



Издательский дом «Питер» приглашает к сотрудничеству авторов.

Обращайтесь по телефонам: Санкт-Петербург — (812) 703-73-72,

Москва — (495) 974-34-50.



Заказ книг для вузов и библиотек: (812) 703-73-73.

Специальное предложение — e-mail: kozin@piter.com

965-



УВАЖАЕМЫЕ ГОСПОДА!
КНИГИ ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»,
ВЫ МОЖЕТЕ ПРИОБРЕСТИ
ОПТОМ И В РОЗНИЦУ
У НАШИХ РЕГИОНАЛЬНЫХ ПАРТНЕРОВ.

ЗАКАЗЫ ВЫМОГАЮТСЯ ПОДАВЛЕНИЕМ
МОЖНО ПОЛУЧИТЬ ПОДАВЛЕНИЕМ
ДЛЯ ВАС СПОСОБОМ:

Башкортостан

Уфа, «Азия», ул. Гоголя, д. 36, офис 5,

тел./факс (3472) 50-39-00, 51-85-44.

E-mail: asiaufa@ufanet.ru

Дальний Восток

Владивосток, «Приморский торговый дом книги»,

тел./факс (4232) 23-82-12.

E-mail: bookbase@mail.primorye.ru

Хабаровск, «Мирс»,

тел. (4212) 30-54-47, факс 22-73-30.

E-mail: sale_book@bookmirs.khv.ru

Хабаровск, «Книжный мир»,

тел. (4212) 32-85-51, факс 32-82-50.

E-mail: postmaster@worldbooks.kht.ru

Европейские регионы России

Архангельск, «Дом книги»,

тел. (8182) 65-41-34, факс 65-41-34.

E-mail: book@atnet.ru

Калининград, «Вестер»,

тел./факс (0112) 21-56-28, 21-62-07.

E-mail: nshibkova@vester.ru

http://www.vester.ru

Северный Кавказ

Ессентуки, «Россы», ул. Октябрьская, 424,

тел./факс (87934) 6-93-09.

E-mail: rossy@kmw.ru

Сибирь

Иркутск, «Продалитъ»,

тел. (3952) 59-13-70, факс 51-30-70.

E-mail: prodalit@irk.ru

http://www.prodalit.irk.ru

Иркутск, «Антей-книга»,

тел./факс (3952) 33-42-47.

E-mail: antey@irk.ru

Красноярск, «Книжный мир»,

тел./факс (3912) 27-39-71.

E-mail: book-world@public.krasnet.ru

Нижневартовск, «Дом книги»,

тел. (3466) 23-27-14, факс 23-59-50.

E-mail: book@nvartovsk.wsnet.ru

Новосибирск, «Топ-книга»,

тел. (3832) 36-10-26, факс 36-10-27.

E-mail: office@top-kniga.ru

http://www.top-kniga.ru

Тюмень, «Друг»,

тел./факс (3452) 21-34-82.

E-mail: drug@tyumen.ru

Тюмень, «Фолиант»,

тел. (3452) 27-36-06, факс 27-36-11.

E-mail: foliant@tyumen.ru

Татарстан

Казань, «Таис»,

тел. (8432) 72-34-55, факс 72-27-82.

E-mail: tais@bancorp.ru

Урал

Екатеринбург, магазин № 14,

ул. Челюскинцев, д. 23,

тел./факс (3432) 53-24-90.

E-mail: gvardia@mail.ur.ru

Екатеринбург, «Валео-книга»,

ул. Ключевская, д. 5,

тел./факс (3432) 42-56-00.

E-mail: valeo@etel.ru

Челябинск, ТД «Эврика», ул. Барбюса, д. 61,

тел./факс (3512) 52-49-23.

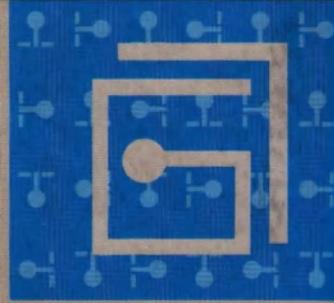
E-mail: evrika@chel.surnet.ru



БИБЛИОТЕКА ПРОГРАММИСТА

С. Макконнелл

СКОЛЬКО СТОИТ ПРОГРАММНЫЙ ПРОЕКТ



В этой книге изложены результаты четырех десятков лет исследований и еще более долгой практической работы, направленной на то, чтобы разработчики, руководители проектов и специалисты по тестированию могли эффективно работать в области оценки программных проектов. Знания в области оценки программных проектов вообще полезны, потому что факторы, влияющие на оценку, также влияют и на сам процесс разработки программного обеспечения.

Ориентируясь на неформальное искусство оценки, автор освещает ряд важных вопросов:

- способы отличить хорошую оценку от плохой;
- различные приемы, помогающие лично вам создавать хорошие оценки;
- оценочные методы, предназначенные для мелких проектов, и методы, работающие на крупных проектах.

Материал книги поможет вам лучше оценивать атрибуты программных проектов, в том числе:

- разработку новых проектов, включая планирование, объем и стоимость работ;
- планирование, объем и стоимость работ по наследным системам;
- объем функциональности, обеспечиваемой для всего проекта при фиксированном временном графике и размере группы
- и вообще практически все, что вы захотите оценивать.

Тема: **Общие вопросы/планирование**

Уровень пользователя: **опытный**

ISBN 978-5-7502-0295-9



9 785750 202959

Издательство
Русская Редакция

Москва
Шелепихинская наб., 32
E-mail: info@rusedit.com
Internet: www.rusedit.com
Тел./факс: (495) 256-7145

ISBN 978-5-91180-090-1



9 795911 800901

Издательский дом
Питер

Санкт-Петербург
Б. Сампсониевский пр., 29а
E-mail: sales@piter.com
Internet: www.piter.com
Тел./факс: (812) 703-7383