Министерство образования и науки, молодежи и спорта Украины

Харьковский национальный университет

Ревенчук Илона Анатольевна

Программной инженерии

Раздел 17 - Оценка длительности и стоимости разработки ПО (Модели COCOMO и SLIM)

6.050103 - Программная инженерия, 8.05010301 - Программное обеспечения систем, 8.05010302 - Инженерия программного обеспечения

Харьков

2005

Содержание

Раздел 17 - Оценка длительности и стоимости разработки ПО (Модели COCOMO и SLIM)

В разделе рассматриваются:

- конструктивная модель затрат-Constructive COst Model - COCOMO, которая применяется для оценки трудозатрат, графика и остальных затрат проекта;

- модель управления процессом жизненного цикла ПО QSM (QSM's Software Lifecycle Management, SLIM), которая состоит из методологий, связанных воедино с помощью инструментов принятия решений: SLIM-Estimate©, SLIM-Control© и SLIM-Metrics©. Инструмент SLIM-Esrimate© поддерживает оценивание и планирование, SLIM-Control© - отслеживание и прогнозирование, а SLIM-Metrics©-захват данных и анализ.

Теория

Введение

Размер программного проекта выражается в тысячах строк программного кода (KSLOC или KLOC). После определения размера оценивают способ производства кода и величину накладных расходов.

Большинство заказчиков и инвесторов не слишком интересуются размером программного кода. Как только менеджер проекта сможет реально планировать и распределять трудозатраты, необходимые для производства того или иного программного продукта, он займется набором персонала, обучением, ротацией, а также другими вопросами, связанными с управлением персоналом.

Оценивание трудозатрат

Под термином "трудозатраты" в процессе оценки ПО подразумевается объем труда, который необходимо выполнить для достижения какой-то цели, выражается в человеко-часах (персональные часы), человеко-днях (персональные дни) либо (наиболее часто) человеко - месяцах (персональные месяцы).

Если показатели производительности сравниваются для различных организаций, определение чел.-мес. будет одинаковым. (В каждом чел.-мес. содержится 4нед.*5раб.дн.*8раб.ч.)

Также имейте в виду, что каждый чел.-мес. определяет, что 1 чел. работает на протяжении 1 мес. (либо 2 чел. работают по 2 недели и т.д.). Для оценки трудозатрат, графика и остальных затрат применяется конструктивная модель затрат -Constructive COst Model - COCOMO.

Применение этой модели предполагает, что в каждом человеко-месяце содержится 19 рабочих человеко-дней и 152 человеко- часа. Эти значения подсчитываются путем учета среднего количества праздничных и выходных дней в США - в случае с другими странами эти значения могут отличаться.

Невозможно производить значительные оценки в случае, если измерения трудозатрат для каждого проекта производятся различным образом.

Для оценки трудозатрат, включают следующее:

задачи, выполняемые в будущем (WBS):

- задачи но разработке ПО (проект, код, тестирование);

- дополнительные задачи по разработке (требования, система);

- задачи поддержки (CM, QA, менеджмент);

- задачи, требующие дополнительных трудозатрат (документы и т.д.);

- дополнительные денежные затраты (поездки, оборудование и т.д.);

оценка размера ПО;

- хронологические данные по трудозатратам и производительности;

график высокого уровня;

- процесс и методы;

- язык программирования;

- операционная система для целевой системы;

используемые инструменты;

уровень профессионального опыта.

Простое вычисление трудозатрат основывается на хронологических данных:

размер * хронологическая производительность = трудозатраты

Независимо от того, какова применяемая единица измерения, использование хронологических данных обеспечивает лучший метод достижения требуемого уровня точности.

Пример1.

Пусть текущие хронологические данные показали следующую производительность:

- сложное ПО: 4 000 LOC на чел.-день;

- простое ПО: 8 000 LOC на чел.-день.

Таким образом, если размер нового ПО оценивается в 8 000000 LOC, то в случае разработки сложного ПО может потребоваться до 8000000 / 4000 = 2 000 человек-дней. Если программа является простой, может потребоваться до 1 000 чел.-дней.

Типичные значения производительности:

50 - 300 000 LOC/месяц (2000 -15 000 LOC/день) для языков высокого уровня;

60 - 500000 LOC/месяц для языков ассемблера.

Минимальные значения характерны для правительственных проектов, которые имеют весьма серьезные ограничения (например, встроенные системы реального времени, надежность которых жизненно важно для обеспечения безопасности людей). Более высокие значения характерны для коммерческих приложений с несколькими ограничениями. Высокие значения этого показателя могут также достигаться в случае использования языков 4GL (fourth-generation programming language), высокой степени повторного использования и в некоторых других подобных ситуациях.

Пример 2.

Трудозатраты на разработку сложного ПО варьируются от 2000 до 8000 LOC на чел.-день, причем среднее значение равно 5000 LOC на один чел.-день.

В этом случае ожидаемые трудозатраты на создание программного кода объемом в 5000000 LOC составят:

5000000 /5000 = 1000 чел.-дней.

Однако это значение может составлять:

5000000 /2000 = 2500 чел.-дней

либо

5000000 /8000 = 625 чел.-дней (минимальное и максимальное значения).

Этапы оценивания

Этап 1. Достижение целей, связанных с оценкой затрат.

Следует четко представлять, каким образом может применяться оценивание. Может ли оно применяться при составлении контракта с фиксированной ценой работ/услуг? Требуется ли внешнее финансирование?

При оценивании необходимо скорректировать цели так, чтобы они соответствовали принимаемым решениям:

Оценивание требует повторного оценивания.

Этап 2. Разработка плана действий при оценивании; план распределения ресурсов.

В плане определяется ожидаемая точность в процессе оценивания затрат. Оценка, выполняется на основе WBS.

Этап 3. Определение требований по разработке ПО.

- Определите требования, необходимые для достижения целей оценивания.

- Сформулируйте спецификации ПО таким образом, чтобы они были, по возможности, четкими и однозначными (желательно исчисляемые). Соответствие ПО имеющимся спецификациям проверяется с помощью тестов.

Этап 4. Учет максимального количества деталей.

По мере роста объема оцениваемого ПО согласно закону больших чисел, уменьшается отклонение результатов оценивания от фактических значений.

Чем больше мы будем размышлять относительно всех программных функций, тем меньшей будет вероятность того, что напрасно будут потрачены средства на использование менее очевидных функций.

Этап 5. Использование нескольких независимых методик.

Ни один из методов не может быть лучше других методов во всех отношениях.

Преимущества и недостатки различных методов являются взаимно дополняемыми.

Комбинация применяемых методик позволяет преодолеть недостатки отдельных методов.

Этап 6. Сравнение, понимание и последовательный просмотр оценок.

Каждый человек обладает уникальным опытом и набором побуждений (феномен оптимиста/ пессимиста).

Применение нескольких независимых методик оценивания позволяет выполнить исследование причин необходимости проведения различных оценок (особенно сильно это сказывается в процессе обучения). В ходе выполнения итераций постепенно достигается реалистичная оценка.

Проявляется феномен закона Парето (Pareto): 80% затрат приходится на создание 20% компонентов. Обращайте более пристальное внимание именно на эти компоненты.

Этап 7. Обзор точности оценивания.

Правильно подбирайте методики и модели оценивания. Как и в случае с оцениванием размера ПО, для оценки трудозатрат доступны несколько моделей. Большинство из них основаны на фундаментальных математических и экспериментальных понятиях, а также в них используется регрессионный анализ.

Регрессионная модель COCOMO

Модель конструктивных затрат (Constructive COst Model, COCOMO) относится к числу наиболее широко применяемых технологий оценивания. Основанная на использовании регрессии модель была разработана доктором Барри В. Боэмом (Dr. Barry W. Boehm) в начале 1970 годов. В то время Барри работал в фирме TRW. Он начал с анализа 63 программных проектов различных типов. При этом оценивался фактический размер (показатель LOC), понесенные трудозатраты, а также фактическую длительность разработки ПО. Регрессионный анализ используется на этапе разработки экспоненциальных уравнений, которые лучше всего описывают связь между разбросанными точками данных.

Режимы модели COCOMO

В модели COCOMO используются три режима, с помощью которых классифицируется сложность системы, а также среды разработки (см.таблицу рис.17.1).

Органический режим обычно классифицируется как платежная ведомость, опись либо научное вычисление. Другие характеристики режима: небольшая команда по разработке проекта, требуются небольшие нововведения, имеются нестрогие ограничения и конечные сроки, а среда разработки является стабильной.

Сблокированный режим типизируется прикладными системами, например, компиляторами, системами баз данных либо редакторами. Другие характеристики: небольшая команда по разработке проекта среднего размера, требуются некоторые инновации, умеренные ограничения и конечные сроки, а среда разработки немного нестабильна.

Внедренный режим характеризуется режимами реального времени, например, системами контроля воздушного движения, сетями ATM или военными системами. Другие характеристики: большая команда разработчиков проекта, большой объем требуемых инноваций, жесткие ограничения и сроки сдачи. Среда разработки в этом случае состоит из многих сложных интерфейсов, включая те из них, которые поставляются заказчикам вместе с аппаратным обеспечением.

Режим	Размер программного продукта	Проект/команда	Потребность в инновациях	Срок сдачи и ограничения	Среда разработки
Органический	2 000 - 50 000 LOC	Небольшой проект и команда - разработчики знакомы с инструментами и языком программирования	Незначительная	Либеральные	Стабильная в домашних условиях
Сблокированный	50 000 – 300 000 LOC	Средние проекты, средняя команда, обладающая средним уровнем возможностей	Средняя	Средние	Средняя
Внедренный	более 300 000 LOC	Большие проекты, требующие большой команды	Максимальная	Серьезные ограничения	Сложные/ интерфейсы заказчика

Рисунок 17.1 - Характеристики режимов COCOMO

Уровни модели COCOMO

Три уровня детализации обеспечивают пользователю последовательное повышение степени точности на каждом последующем уровне.

Базовый уровень. На этом уровне для определения необходимых трудозатрат и графика используется лишь значение размера и сведения о текущем режиме. Он пригоден при выполнении быстрых и приближенных оценок при выполнении небольших и средних по объему проектов.

Промежуточный уровень. На этом уровне применяются сведения о размере, режиме и 15 дополнительных переменных с целью определения необходимых трудозатрат. Дополнительные переменные называются "драйверами затрат" и имеют отношение к атрибутам продукта, персонала, компьютера и проекта, которые могут являться результатом более ли менее значительных трудозатрат.

Драйвер затрат или Фактор формирования затрат - причина, определяющая изменение величины затрат. Фактор затрат должен быть измеримым, т.е. поддаваться количественному определению. Факторами прямых затрат являются объекты, на единицу которых выделяются бюджетные средства

Произведение драйверов затрат называется ***корректировочным множителем среды (Environmental adjustment factor, EAF)***.

Детализированный уровень. Этот уровень надстраивается на промежуточном уровне COCOMO путем внедрения дополнительных множителей трудозатрат, чувствительных к фазе, и трехуровневой иерархии программных продуктов. Промежуточный уровень может быть настроен по фазе и по уровню разработки продукта с целью достижения детализированного уровня. В качестве примера множителей трудозатрат, чувствительных к фазе, можно рассматривать ограничения по памяти, которые могут применяться при попытках оценивания фаз кодирования или тестирования проекта. Однако в то же самое время показатели размера памяти могут не оказывать влияния на величину затрат либо трудозатрат на фазе анализа. Это становится еще более очевидным после описания множителей трудозатрат (либо драйверов затрат).

Множители, чувствительные к фазе, обычно резервируются для использования зрелыми организациями и требуют применения автоматизированных инструментов.

Трехуровневая иерархия программных продуктов состоит из системы, подсистемы и модуля, подобно тому, как реализовано расположение в структуре WBS. Большие проекты могут разбиваться, как минимум, на три уровня.

Базовая модель COCOMO

Оценка трудозатрат

Показатель KLOC касается исключительно входной переменной.

Режим	a	b	Формула для оценки трудозатрат =a*(размер, KLOC)^{b}, чел.-мес.	Формула для определения времени разработки
Органический	2,4	1,05	$E=2,4*(S)^{1,05}$	$TDEV=2,5*(E)^{0,38}$ месяцы
Сблокированный	3,0	1,12	$E=3,0*(S)^{1,12}$	$TDEV=2,5*(E)^{0,35}$ месяцы
Внедренный	3,6	1,20	$E=3,6*(S)^{1,20}$	$TDEV=2,5*(E)^{0,32}$ месяцы

Рисунок 17.2 - Базовые формулы оценки необходимых для разработки времени и трудозатрат в модели COCOMO

Трудозатраты измеряются в человеко - месяцах (19 дней в месяце либо 152 рабочих часа в месяце, константы ***a*** и ***b*** могут определяться с помощью процедуры построения кривой по точкам (регрессионный анализ), причем данные проекта сравниваются с помощью уравнения. Большинство организаций не располагают массивом данных, достаточным для выполнения подобного анализа, начиная с применения дерева уровней.

В случае использования различных режимов проекты одинакового масштаба требуют различных трудозатрат.

Трудозатраты - ***E***, Время разработки - ***TDEV***, Средняя численность персонала - ***SS***.

$$SS = E / TDEV$$

Пример 3.

Пусть размер проекта 200 KLOC из таблицы рис.17.1 размер средний, режим сблокированный. Заполним таблицу на рис.17.3.

Режим	a	b	Формула для оценки трудозатрат, E= a*(размер, KLOC)^b, чел.-мес.	Формула для определения времени разработки, TDEV, месяцы	Средняя численность, SS = E / TDEV	Производительность, = (размер, LOC) / E
Органический	2,4	1,05	$E=2,4*(200)^{1,05}=626$	$TDEV=2,5*(626)^{0,38}=2,5*11,55=28,8$	=21,7 = 22	200 000 /626 = 319,5
Сблокированный	3,0	1,12	$E=3,0*(200)^{1,12}=1133$	$TDEV=2,5*(1133)^{0,35}=2,5*11,72=29,3$	=38,6 = 39	200 000 /1133 = 176,5
Внедренный	3,6	1,20	$E=3,6*(200)^{1,20}=2077$	$TDEV=2,5*(2077)^{0,32}=2,5*11,52=28,8$	72,1 = 73	200 000 /2077 = 96,3

Рисунок 17.3 - Разработка времени и трудозатрат в модели COCOMO для проекта 200 KLOC

Фаза распределения трудозатрат и пунктов графика в базовой модели COCOMO

В дополнение к оценке графика и трудозатрат во время разработки программного проекта, зачастую требуется оценить, каким образом трудозатраты распределяются среди действий первичного жизненного цикла. При использовании модели COCOMO обеспечивается упрощенный подход к применению фаз жизненного цикла. При реализации этого подхода учитываются только планы и требования, разработка проекта продукта, кодирование, а также интеграция и тестирование в качестве четырех фаз разработки. Фаза поддержки рассматривается в качестве финальной фазы жизненного цикла. Каждое из упомянутых действий может выполняться на протяжении любой из перечисленных ниже фаз: анализ требований, разработка проекта продукта, кодирование, планирование тестирования, проверка, офисные функции проекта, управление конфигурацией и обеспечение качества, документирование и т.д.

А теперь будет рассмотрен пример фазы распределения трудозатрат и пунктов графика (таблица рис.17.4).

Предположим, что размер проекта, выполняемого во внедренном режиме, составляет 80 KLOC.

E (трудозатраты, чел.-месяцы) = 3,6*(KLOC) 1,2 = 3,6*(80)1,2° = 3,6*(192,18) = 692 чел.-месяца
TDEV (время разработки) = 2,5*(E)0,32 = 2,5 (692) 0,32 = 2,5*(8,106) = 20 месяцев.

Фазы проекта→	Планы и требования	Разработка проекта продукта	Кодирование	Интеграция и тестирование	Итого
Трудозатраты в % (% трудозатрат на каждой основной фазе на основе хронологических данных)	10	12	50,5	27,5	100% трудозатрат, распределенных на 4 основные фазы
Трудозатраты SM = типичный процент трудозатрат на данной фазе * на оцененный SM	0,1*692 SM =6,92 мес.	0,12*692 SM =83,04 мес.	0,505*692 SM=349,46 мес.	0,275*692 SM=190,3 мес.	Итого 692 мес. (из расчета на 1 чел.-мес.)
График % (% графика на каждой основной фазе, определенный на базе хронологических данных)	4	33	38	25	100% графика, распределенного на 4 основных фазах
График (месяцы) = типичный % графика на данной фазе * на оцененный TDEV	0,04*20=0,8 мес.	0,33*20=6,6 мес.	0,38*20=7,6 мес.	0,25*20=5 мес.	20 мес. Общей длительности (график)
Средняя численность персонала на основной фазе (трудозатраты SM / график-месяцы)	6,92/0,8 = 8,65	83,04/6,6 = 12,5818	349,46/7,6 = 45,98	190,3/5 = 38,06	692/20 = 34,6

Рисунок 17.4 - Пример фазы распределения трудозатрат и пунктов графика

Промежуточная модель COCOMO

В промежуточной модели COCOMO используются значения размера и режимы, подобные тем, которые применялись в базовой модели. Дополнительно применяются 15 переменных, называемых драйверами затрат, с помощью которых могут быть объяснены и модифицированы уравнения трудозатрат (таблица рис.17.5). Идея, применяемая в этом случае, заключается в том, что характеристики данного проекта управляют затратами (трудозатратами).

Режим	a	b	Формула для оценки трудозатрат =a*(размер, KLOC)^{b}*C, чел.-мес.	Формула для определения времени разработки
Органический	3,2	1,05	$E=3,2*(S)^{1,05}*C$	$TDEV=2,5*(E)^{0,38}$ месяцы
Сблокированный	3,0	1,12	$E=3,0*(S)^{1,12}*C$	$TDEV=2,5*(E)^{0,35}$ месяцы
Внедренный	2,8	1,20	$E=2,8*(S)^{1,20}*C$	$TDEV=2,5*(E)^{0,32}$ месяцы

Рисунок 17.5 - Формулы для оценки трудозатрат в промежуточной модели COCOMO

Трудозатраты измеряются в человеко - месяцах (19 дней в месяце либо 152 рабочих часа в месяце, константы ***a*** и ***b*** могут определяться с помощью процедуры построения кривой по точкам (регрессионный анализ), причем данные проекта сравниваются с помощью уравнения. Большинство организаций не располагают массивом данных, достаточным для выполнения подобного анализа, начиная с применения дерева уровней.

В случае использования различных режимов проекты одинакового масштаба требуют различных трудозатрат.

Трудозатраты - ***E***, Время разработки - ***TDEV***, Средняя численность персонала - ***SS***.

$$SS = E/TDEV$$

Пример 4.

Пусть размер проекта 200 KLOC из таблицы рис.17.1 размер средний, режим сблокированный. Заполним таблицу на рис.17.6.

Режим	a	b	Формула для оценки трудозатрат, E= a*(размер, KLOC)^{b}, чел.-мес.	Формула для определения времени разработки, TDEV, месяцы	Средняя численность, SS = E / TDEV	Производительность, = (размер, LOC) / E
Органический	3,2	1,05	$E=3,2*(200)^{1,05}=$ 3.2*260 = **832**	$TDEV=2,5*(832)^{0,38}=$ 2,5*11,60 = **29**	=28,7 = **29**	200 000 /832 = **240,4**
Сблокированный	**3,0**	**1,12**	**$E=3,0*(200)^{1,12}=$ 3.0*378 = 1133**	**$TDEV=2,5*(1133)^{0,35}$ = 2,5*11,72 = 29,3**	=38,6 = 39	**200 000 /1133 = 176,5**
Внедренный	2,8	1,20	$E=2,8*(200)^{1,20}=$ 2.8*577 = **1616**	$TDEV=2,5*(1616)^{0,32}$ =2,5*10,63 = **26,58**	60,8 = **61**	200 000 /1616 = **123,8**

Рисунок 17.6 - Оценка трудозатрат в промежуточной модели COCOMO для проекта в 200 KLOC

Драйверы затрат

Концепция, связанная с фактором корректировки трудозатрат (Effort adjustment factor, EAF), заключается в том, что он создает эффект увеличения либо уменьшения трудозатрат, а следовательно, и затрат, в зависимости от набора факторов среды.

Факторы среды иногда называются факторами корректировки затрат [C,s] либо драйверами затрат. Определение этого фактора-множителя происходит в два этапа.

На этапе 1 драйверам затрат назначаются числовые значения.

На этапе 2 происходит перемножение драйверов затрат, в результате чего генерируется фактор корректировки трудозатрат, т.е. C.

Фактор ***EAF*** представляет собой произведение факторов корректировки затрат.

EAF=C1*C2*....Cn

Факторы корректировки затрат могут сказываться на оценках графика и затрат проекта, изменяя их в 10 и более раз!

Драйверы затрат группируются в виде четырех категорий, как показано в таблице рис.17.7.

Программный продукт	Компьютер	Персонал	Проект
Требуемая надежность ПО (RELY)	Ограничения времени выполнения (TIME)	Способности аналитика (ACAP)	Использование практики современного программирования (MODR)
Размер базы данных (DATA)	Ограничения основного хранилища (STOR)	Опыт в создании приложений (AEXP)	Использование инструментов разработки ПО (TOOL)
Сложность программного продукта (CPLX)	Изменяемость виртуальной машины (VIRT)	Способности программиста (PCAP)	План требуемой разработки (SCED)
	Оборотное время компьютера (TURN)	Опыт в области виртуальных машин (VEXP)	
		Опыт в области языков программирования (LEXP)	

Рисунок 17.7 - Категории драйверов затрат в промежуточной модели COCOMO

EAF= RELY * DATA * CPLX* TIME * STOR * VIRT * TURN * ACAP * AEXP * PCAP * VEXP* LEXP* MODR* TOOL * SCED

Каждый драйвер затрат определяет умножающий фактор, который позволяет оценить эффект

действия атрибута на величину трудозатрат.

Числовые значения драйверов затрат при их совместном перемножении образуют фактор корректировки, т.е. C,

Атрибуты программного продукта .

Некоторые из атрибутов, которые могут изменять величину затрат проекта, могут применяться наравне с самим продуктом или выполняться в ходе соответствующей работы. Ниже перечислены эти атрибуты:

- требуемая надежность - как правило, применяется в системах реального времени;

- размер базы данных - в основном применяется в приложениях обработки данных;

- сложность продукта - ограничения на время выполнения.

Атрибуты, связанные с аппаратными средствами.

Другие атрибуты имеют отношение к компьютерной платформе и могут применяться в качестве средства поддержки, а также при наличии работы, которая должна быть выполнена:

- ограничения времени выполнения - применяются в том случае, когда быстродействие процессора является ограниченным;

- ограничения основного хранилища - применяются в случае, когда размер памяти является ограниченным;

- меняемость виртуальной машины - включает аппаратное обеспечение и операционную систему на целевом компьютере;

- оборотное время компьютера - применяется при разработке.

Атрибуты проекта.

Атрибуты, связанные с практикой и инструментами:

- практика современного программирования - структурные или ОО-технологии;

- современные инструменты программирования - CASE-инструменты, хорошие отладчики, инструменты, используемые при выполнении тестирования;

- сжатие (или расширение) графика - отклонение от идеала всегда удручает, но меньшая степень отклонения всегда лучше, чем большая.

Атрибуты персонала.

Некоторые атрибуты применяются для описания исполнителей работ:

- способности аналитика;

- опыт в создании приложений;

- способности программиста;

- опыт в области виртуальных машин, включая операционную систему и аппаратное обеспечение;

- опыт в области языков программирования, включая инструменты и практику.

Другие драйверы затрат.

Несмотря на то, что наиболее часто с приложениями в рамках промежуточной модели COCOMO связываются указанные выше четыре категории атрибутов, менеджер проекта может добавлять дополнительные атрибуты:

- изменяемость требований - некоторые из них являются ожидаемыми, однако большинство из них могут представлять значительную проблему;

- изменяемость машины, предназначенной для разработки - нестабильные ОС, компиляторы, CASE-инструменты и т.д.;

- требования безопасности - применяются для классифицированных программ;

- доступ к данным - иногда является весьма затрудненным;

- влияние стандартов и навязанных методов;

- влияние физического окружения.

Драйверы затрат выбираются в соответствии с их общей значимостью для всех программных проектов, причем они являются независимыми от размера проекта.

Поскольку драйверы затрат являются мультипликативными, в случае, если драйвер затрат не влияет на трудозатраты, его значение равно 1. При этом конечное значение С не изменяется. Подобные драйверы затрат называются нормальными либо "номинальными".

Например, если опыт в области языков программирования (LEXP) команды в рассматриваемой организации больше, чем аналогичный показатель в любой другой организации в этом городе, значение LEXP будет оставаться равным 1.

Это связано с тем, что превосходящие способности в области языков программирования нормируются в данной среде. Оценщик может выполнять поиск условий, при наступлении которых возрастает показатель трудозатрат (произведение всех драйверов затрат превышает 1, ***EAF>1***) либо значение этого показателя уменьшается (произведение всех драйверов затрат меньше 1, ***EAF<1***). При поиске применяется критерий "обычности" для данной среды. Как правило, объем трудозатрат увеличивает в случае, если применяется новая технология,

команда разработчиков только что сформирована либо состоит из неопытных в данной области программистов, имеет место повышенная сложность технологической проблемы либо имеют место другие условия, отличные от стандартных. Если же требуется меньше трудозатрат, то это означает, что подобные проблемы были успешно разрешены ранее.

Сложность продукта (CPLX) может принимать значение 0,70 (очень низкое) и 1,60 (очень высокое). Этот показатель становится заметным в случае, если произведение драйверов затрат (C) оказывает влияние на оценку трудозатрат. Если показатель CPLX представляет только драйвер затрат, который не является номинальным, а физические трудозатраты составляют 24 чел.-мес. Уровень сложности может оказывать влияние на трудозатраты, он ранжируется в диапазоне от 16,8 (24 чел.-мес * 0,7) до 38,4 (24 чел.-мес * 1,6) чел.-мес.

Возможные значения исходных 15 драйверов затрат перечислены в таблице рис.17.8.

Программный продукт	Показатели					
	Очень низкий	низкий	номинальный	высокий	Очень высокий	сверхвысокий
Атрибуты продукта						
Требуемая надежность ПО (RELY)	0,75	0,88	1	1,15	1,40	
Размер базы данных (DATA)		0,94	1	1,08	1,16	
Сложность программного продукта (CPLX)	0,7	0,85	1	1,15	1,3	1,65
Атрибуты компьютера						
Ограничения времени выполнения (TIME)			1	1,11	1,3	1,66
Ограничения основного хранилища (STOR)			1	1,06	1,21	1,56
Изменяемость виртуальной машины (VIRT)		0,87	1	1,15	1,3	
Оборотное время компьютера (TURN)		0,87	1	1,07	1,15	
Способности аналитика (ACAP)	1,46	1,19	1	0,86	0,71	
Опыт в создании приложений (AEXP)	1,29	1,13	1	0,91	0,82	
Способности программиста (PCAP)	1,42	1,17	1	0,86	0,7	
Опыт в области виртуальных машин (VEXP)	1,21	1,1	1	0,9		
Опыт в области языков программирования (LEXP)	1,14	1,07	1	0,95		
Атрибуты проекта						
Использование практики современного программирования (MODR)	1,24	1,10	1	0,91	0,82	
Использование инструментов разработки ПО (TOOL)	1,24	1,10	1	0,91	0,82	
План требуемой разработки (SCED)	1,23	1,08	1	1,04	1,10	

Рисунок 17.8 -Значения драйверов затрат при разработке ПО в рамках модели COCOMO

Существуют отдельные таблицы оценки драйверов трудозатрат сложности продукта.

Примеры реализации промежуточной модели COCOMO

Ниже приводятся два примера промежуточной модели COCOMO. В первой модели

применяются нормальные значения для драйверов затрат; в другой модели увеличиваются оценки для показателей ACAP и PCAP.

Пример 5.

Рассматривается программный проект внедренного режима, оцениваемый показателем в 10 KLOC, реализующий функции обработки коммуникаций в коммерческом микропроцессоре.

- Формула для внедренного режима обеспечивает номинальное значение трудозатрат: En = $2,8+(10)^{1,20}$ = 2,8*15.85 = 44 человеко-месяца.

- В результате оценивания среды проекта получаются результаты, обеспечивающие вычисление вариантов значений множителя драйвера затрат. Эти значения перечислены в таблице рис.17.9.

- Фактор корректировки применяется по отношению к номинальным трудозатратам: E = $2,8*(10)^{1,20}$ * C = 44 * 1,17 = 51 человеко-месяц

Драйвер затрат	Применение	Оценка	Множитель трудозатрат
Атрибуты продукта			
Требуемая надежность ПО (RELY)	Локальное применение системы. Не возникают серьезные проблемы с восстановлением данных	Номинальная	1
Размер базы данных (DATA)	30000 байт	Низкая	0,94
Сложность программного продукта (CPLX)	Обработка коммуникаций	Очень высокая	1,3
Атрибуты компьютера			
Ограничения времени выполнения (TIME)	Будет применяться 70% свободного времени	Высокая	1,11
Ограничения основного хранилища (STOR)	45 Кбайт из 64 Кбайт доступного хранилища (70 %)	Высокая	1,06
Изменяемость виртуальной машины (VIRT)	Основано на коммерческом микропроцессорном аппаратном обеспечении	Номинальная	1
Оборотное время компьютера (TURN)	Среднее время обхода равно 2 часам	Номинальная	1
Способности аналитика (ACAP)	Опытный старший аналитик	Высокая	0,86
Опыт в создании приложений (AEXP)	3-летний опыт	Номинальная	1
Способности программиста (PCAP)	Опытные старшие программисты	Высокая	0,86
Опыт в области виртуальных машин (VEXP)	6 месяцев	Низкая	1,10
Опыт в области языков программирования (LEXP)	12 месяцев	Номинальная	1
Атрибуты проекта			
Использование практики современного программирования (MODR)	Большинство технологий применяется более одного года	Высокая	0,91
Использование инструментов разработки ПО (TOOL)	На уровне базового миникомьютерного инструмента	Низкая	1,10
План требуемой разработки (SCED)	10 месяцев	Номинальная	1
EAF	C = 1,00 * 0,94 * 1,30 * 1,11 * 1,06 * 1,00 * 1,00 * 0,86 * 1,00 * 0,86 * 1,10 * 1,00 * 0,91 * 1,10 * 1,00	C= 1,17	

Рисунок 17.9 - Значения драйверов затрат для промежуточной модели COCOMO, пример 5

Пример 6

При выполнении оценки проекта получается значение, равное 44 человеко- месяцам (SM). Если при выполнении проекта привлекается более квалифицированный персонал, оценки PCAP и ACAP уменьшаются от номинальных (1,00) до высоких (0,86). Однако затраты на персонал возрастают с $5000 до $6000 из расчета на один SM. Предположим, что значения других драйверов затрат будут номинальными (1,00).

Фактор корректировки трудозатрат (EAF) = C =

RELY * DATA * CPLX * TIME * STOR * VIRT * TURN * ACAP * AEXP * PCAP * VEXP * LEXP * MODP * TOOL * SCED = 1,00 * 1,00 * 1,00 * 1,00 * 1,00 * 1,00 * 1,00 * 0,86 * 1,00 * 0,86 * 1,00 * 1,00 * 1,00 * 1,00 * 1,00 * 1,00 = 0,74

Скорректированный показатель человеко-месяцев: 44 SM * 0,74 = 32,6

Разница в затратах:

44 SM * $5000 /SM = $220000

32,6 SM * $6000 /SM = $195600Разница: $220000 - $195600 = $24400

Вывод

В настоящем примере использование услуг более квалифицированного персонала обходится дешевле, несмотря на возросшие при этом расходы на оплату труда.

Рассмотренные 15 драйверов затрат способствуют лучшему использованию сред, в которых отсутствуют либо оказывают незначительное влияние хронологические данные, они не могут применяться одинаковым образом в нескольких организациях либо даже в одной организации на протяжении длительного периода времени.

После того как станут доступными фактические данные, с их помощью будут изменены исходные данные, а модель будет калиброваться до тех пор, пока она не будет полностью соответствовать данному типу приложения и данной организационной среде либо среде и приложению одновременно. Многие производители поддерживают автоматизированные инструменты, предназначенные для выполнения оценок, имеющих заданные размер и сложность. Многие из этих инструментов обеспечивают автоматизированную поддержку добавления фактических данных и калибровку модели путем изменения показателя, коэффициента, значений драйвера затрат либо всех трех параметров одновременно.

Детализированная модель COCOMO

Теперь рассмотрим наиболее сложную версию COCOMO. Она включает следующие дополнительные шаги.

1. Программа разбивается на специфические продукты и компоненты этих продуктов. Согласно Боэму (Boehm), подобное разбиение называется трехуровневой иерархией продуктов: ***система, подсистема и модуль***. Верхний уровень, уровень системы, используется для применения самых общих отношений, связанных с проектом, таких как номинальные трудозатраты и уравнения графика, а также для применения номинальных трудозатрат на уровне проекта и пофазной разбивки графика. Самый нижний уровень, уровень модуля, описывается с помощью показателя KLOC в модуле и драйверов затрат, которые могут варьироваться на этом уровне. Второй уровень, уровень подсистемы, описывается с помощью

оставшихся драйверов затрат и может отличаться в различных подсистемах. Однако этот уровень не будет изменяться в различных модулях, входящих в состав одной подсистемы.

2. Анализ драйверов затрат производится отдельно для каждого компонента. Подсистемы и модули наследуют драйверы затрат системы. Они называются: RELY, VIRT, TURN, MODP, TOOL и SCED. Модули наследуют драйверы затрат подсистемы DATA, TIME, STOR, ACAP, AEXP (проявляется тенденция к применению одних и тех же модулей внутри подсистемы). Драйверы затрат модуля имеют такие названия: KLOC, AAF, CPLX, PCAP, VEXP и LEXP. Драйвер AAF является новым - результат адаптации существующих модулей. Дополнительная информация доступна до этапа изменения существующего ПО - благодаря этим данным обеспечиваются более корректные оценки. Как правило, большинство создаваемых программных продуктов не разрабатываются "с нуля", а являются результатом повторного использования существующих модулей, реализуемых с помощью объектно-ориентированных методов. Использование детализированной модели COCOMO зачастую эквивалентно дополнительным трудозатратам. Следует отметить, что затраты на этапе повторного использования не всегда равны нулю. Ведь не обойтись без трудозатрат на этапе работы с существующим кодом и разработки соответствующего интерфейса. Затраты на переписывание системы могут быть меньшими, чем продолжение ее поддержки (по причине энтропии структуры). Однако переписывание старой системы может быть более дорогостоящим, чем создание "новой" системы. Распространено мнение, согласно которому точка экономической безубыточности достигается в результате изменения 20% кода; после прохождения точки безубыточности повторное использование кода не будет эффективным.

3.Действия по разработке проекта разбиваются на фазы. В работах Боэма (Boehm) используются четыре основных фазы: требования (RQ), разработка проекта продукта (PD), детализированный дизайн продукта (DD), кодирование и тестирование разрабатываемого модуля (CUT). Интеграция и тестирование (IT), а также поддержка (MN) описываются на протяжении всего жизненного цикла. Фазы могут применяться для разбиения систем, подсистем, и/или модулей. На каждой фазе могут применяться различные множители трудозатрат. Различные значения множителей драйверов затрат устанавливаются на каждом из трех уровней иерархии программных продуктов (система, подсистема, модуль), а также на каждой фазе (RPD, DD, CUT, IT) внутри иерархий.

Составление графика с помощью модели COCOMO

После оценки размера и величины трудозатрат начинается этап составления графика работ. В этот процесс включается настройка структуры WBS (при необходимости), а также определение ответственности для членов команды. При составлении графика также планируется начальная и конечная дата выполнения каждой задачи, указываются зависимости между задачами и обеспечивается параллелизм выполнения (если это возможно). При наличии зависимостей одна задача нуждается во входных данных другой либо нескольких других задач. В силу этого приходится ожидать завершения выполнения других задач, после чего может начаться выполнение текущей задачи. При наличии параллелизма несколько задач могут выполняться одновременно. Диаграммы Ганта и Перта представляют собой два наиболее широко используемых метода, используемых для графического представления задач, зависимостей,

параллелизма и временных интервалов. При создании этих диаграмм используются автоматизированные инструменты.

Подгонка модели COCOMO

Организация-заказчик может заказать разработку специальным образом калиброванной версии, которая должна более точно отражать текущие применяемые практики и возможности. Существует три различных способа, применяемых для повторной калибровки и корректировки уравнений промежуточной модели COCOMO с учетом применения локальных условий и с основой на локальной хронологической базе данных для и проектов: коэффициент повторной калибровки уравнений трудозатрат, настройка режима (повторная калибровка показателя и коэффициента), а также проверка значений драйверов затрат. В этой книге мы не будем подробно рассматривать формулы и детали, касающиеся повторной калибровки моделей. Подробные инструкции могут быть найдены в книге Боэма (Boehm), Software Engineering Economics. Точная повторная калибровка коэффициента требует наличия базы данных, которая содержит, как минимум, пять проектов. Для точной повторной калибровки показателя база данных должна включать, как минимум, десять проектов.

Преимущества модели COCOMO

- фактические данные подгоняются в соответствии со многими реальными программами, поддерживающих набор констант COCOMO и факторов корректировки, которые могут идеально соответствовать организации;

- применяемый в данном случае процесс является повторяемым;

- метод позволяет добавлять уникальные факторы корректировки, связанные с данной организацией;

- он является достаточно универсальным и может поддерживать различные "режимы" и "уровни";

- идеально подходит для проектов, между которыми нет существенных отличий относительно размера, сложности или протекания процесса;

- возможна высшая степень калибровки с опорой на предыдущий опыт,

- обязательная документация;

- простота в применении.

Недостатки модели COCOMO

- игнорируется изменяемость требований (однако в организациях этот момент может учитываться с помощью дополнительного фактора корректировки, EAF);

- игнорируется документация и другие требования;

- игнорируются атрибуты заказчика- навыки, кооперирование, знания и способность к реагированию;

- слишком упрощается влияние вопросов безопасности;

- игнорируются проблемы обеспечения безопасности ПО;

- не учитывается среда разработки ПО;

- игнорируются уровни взаимодействия персонала;

- игнорируются многие вопросы, связанные с аппаратным обеспечением;

- все уровни зависят от оценки размера - точность оценки размера оказывает влияние на точность оценки трудозатрат, время разработки, подбор персонала и оценку производительности;

- оценки, основанные на опыте, могут быть некорректными по причине устаревания используемых хронологических данных либо по причине забывчивости специалиста по оценке;

- существует зависимость между знаниями по драйверам затрат и количеством времени, затраченного на каждой фазе.

Модель COCOMO изначально представляет трудозатраты на этапе разработки (от фазы планирования до фазы реализации). Проблемы поддержки, повторного рабочего процесса, переноса и повторного использования не могут четко описываться в рамках одной и той же модели. Эти действия могут быть также оценены с помощью применяемых вариаций базовой модели.

При использовании модели COCOMO предполагается наличие очень простого базового уровня трудозатрат, используемого при управлении конфигурацией и обеспечении качества. В этом случае используется примерно 5% общего бюджета (основываясь на типичной коммерческой практике, принятой при использовании модели COCOMO). Показатели затрат могут быть увеличены в 2-4 раза при разработке современного ПО с учетом сложности современных программных продуктов.

Ваши данные не должны соответствовать данным, применяемым при разработке в рамках модели COCOMO, - если это условие не выполняется, потребуется сбор данных, необходимых для корреляции модели.

В модели COCOMO предполагается использование базовой модели процесса каскада: разработка проекта (30%), кодирование (30%), интеграция и тестирование (40%).

В модели COCOMO исключаются следующие компоненты:

- разработка и спецификация требований, которые не слишком часто применяются в некоторых коммерческих приложениях (даже, несмотря на то, что эта часть выполненной оценки обычно рассматривается в качестве области ответственности функций инжиниринга систем или анализа систем, она будет завышаться до тех пор, пока программисты не выполнят точную оценку затрат);

- прирост на 20% обычно требуется на фазе формулирования требований, - даже если применяются объектно-ориентированные методы;

- менеджмент (обычно выполняется менеджмент на уровне строк, а не общий менеджмент);

- накладные расходы;

- расходы на командировки и другие побочные расходы;

- системная интеграция и поддержка тестирования;

- поддержка тестовых полей;

- компьютеры;

- источники поставок;

- определенное место в офисе.

Согласно Боэму (Boehm), трудозатраты изначально были распределены таким образом, чтобы 30% приходилось на дизайн, 30% - на кодирование и тестирование модуля, а 40% - на интеграцию и тестирование.

Если добавить 20% трудозатрат к общему показателю трудозатрат для анализа требований, получаем 17% на фазе анализа требований, 25% - на фазе разработки проекта, 25% - на фазе кодирования и тестирования модуля и 33% - на фазе интеграции и тестирования.

Объем затраченного времени является максимальным на фазе анализа требований и разработки проекта, но конкретная специфика зависит от используемого процесса.

Обычные значения для распределения времени составляют 30% на фазе анализа требований, 30% - на фазе разработки проекта, 15% - на фазе кодирования, а 25% - на фазе интеграции и тестирования.

Некоторые типичные проблемы, влияющие на выполнение графика или уменьшение трудозатрат

- отсутствие адекватного оборудования, ПО, инструментов, квалифицированного персонала и

т.д.;

- замедленные циклы утверждения;

- плохая координация с другими дисциплинами, другими компаниями и т.д.;

- недостаточно подготовленные заказчики и менеджеры;

- иррационально мыслящие заказчики и менеджеры;

- преднамеренные препятствия в лице конкурентов.

Модель COCOMO II

Модель COCOMO II является улучшенной версией исходной модели COCOMO, обладающей улучшенными возможностями. Благодаря этой модели облегчается выполнение оценки для объектно-ориентированного ПО, программ, созданных с применением спиральной либо эволюционной моделей, а также приложений, разработанных на базе готовых коммерческих программ.

В частности, эта модель применяется при оценке программных затрат на разработку баз данных и возможностей поддержки инструментов в процессе непрерывного улучшения модели. Эта модель также формирует аналитическую основу, набор инструментов и техник, применяемых при оценке эффектов, связанных с улучшением технологии разработки ПО на базе графиков и затрат на фазе жизненного цикла разработки ПО.

На ранних концептуальных стадиях проекта в модели применяется оценка на основе метода объектных точек с целью вычисления трудозатрат. На ранних стадиях разработки проекта, когда еще практически ничего не известно о размере проекта и о его разработчиках, не скорректированные функциональные точки применяются в качестве входных данных для модели. После того как была выбрана архитектура, фазы разработки проекта и кодирования начинаются с ввода параметра SLOC для модели.

В модели COCOMO II поддерживаются вероятностные диапазоны оценок, представляющие одно стандартное отклонение на фоне наиболее благоприятных оценок.

Аккомодирующие факторы, которые получили не слишком большое распространение в первой версии модели COCOMO, в настоящее время применяются для создания возможности повторного использования ПО, а также повторного инжиниринга. Две последние возможности представляют собой автоматизированные инструменты, используемые для трансляции существующего ПО. В модели COCOMO II также учитывается изменчивость требований для выполняемых оценок.

В то время как показатель размера в уравнении оценки трудозатрат в оригинальной модели COCOMO варьируется в зависимости от текущего режима разработки, в модели COCOMO II применяются факторы масштабирования с целью обобщения и замещения эффектов в режиме

разработки.

В композиционной прикладной модели COCOMO II для выполнения оценок применяется метод объектных точек. При использовании этой модели предполагается использование интегрированных CASE-инструментов с целью выполнения быстрого прототипирования. Объекты включают экраны, отчеты и модули, имеющие отношение к языкам программирования третьего поколения. Оценивается количество физических объектов, сложность каждого объекта, а также вычисляется взвешенный итог (подсчитывается количество объектных точек). Также оценивается процентное соотношение показателей повторного использования и ожидаемой производительности. На основе этой информации может выполняться оценка трудозатрат.

В модели COCOMO II явным образом обеспечивается доступ к дополнительной информации на поздних стадиях проекта, обрабатываются нелинейные затраты при повторном использовании программных компонентов, а также оцениваются эффекты воздействия нескольких факторов по шкале показателей убытков. Некоторые из перечисленных параметров представляют собой оценку оборота персонала, географическое распределение команды, а также "зрелость" процесса разработки в том виде, в котором она описывается Институтом SEI. В этой модели также проверяются некоторые значения коэффициента, и устраняется присутствие "сосредоточенных неоднородностей" в старой модели (связанных с "режимами разработки", поддержкой и адаптацией).

Фактически COCOMO II включает три различные модели.

Композиционная прикладная модель - эта модель подходит для проектов, созданных с помощью современных инструментальных средств, применяемых для "строительства" GUI. Модель основывается на новых объектных точках.

Модель ранней разработки проекта - эта модель применяется для получения приближенных оценок проектных затрат и периода выполнения проекта перед тем, как будет определена архитектура в целом. В этом случае используется небольшой набор новых драйверов затрат и новых уравнений оценки. Также в качестве базы используется набор не настраиваемых функциональных точек либо KSLOC.

Пост-архитектурная модель - наиболее детализированная модель COCOMO II, которая используется после разработки общей архитектуры проекта. В состав этой модели включены новые драйверы затрат, новые правила подсчета строк, а также новые уравнения. Усилиями фирмы Rational, Inc. модель COCOMO II интегрирует фазы и основные стадии с аналогичными объектами, разрабатываемыми фирмой Rational Unified Process, которая, в свою очередь, поддерживает распределение фаз и действий для оценщиков, использующих модель COCOMO II.

Математическая модель SLIM

Процесс управления жизненным циклом разработки ПО QSM (QSM's Software Lifecycle Management, SLIM) состоит из методологий, связанных воедино с помощью инструментов

принятия решений: SLIM-Estimate©, SLIM-Control© и SLIM-Metrics©.

Инструмент SLIM-Esrimate© поддерживает оценивание и планирование, SLIM-Control© - отслеживание и прогнозирование, а SLIM-Metrics©-захват данных и анализ.

Производимое ПО (размер) = трудозатраты/время на производственном уровне.

Все методы и процессы, реализующие измерение и оценивание, взаимодействуют с коллекциями фактических данных. Преимущество, проявляющееся при использовании автоматизированного инструмента, заключается в том, что большинство из этих инструментов поддерживают "стартовые наборы" данных, основанных на наблюдаемых проектах. Исходя из этой точки зрения, можно отметить, что модель SLIM является особенно полезной в связи с тем, что могут собираться данные более чем из 5500 проектов. Программные уравнения Патнама связывают размер ПО со временем разработки и общим объемом трудозатрат.

Продукт = производительность * трудозатраты * время

$$S = C * K^{1/3} * t_d^{4/3}$$

где

S - размер ПО (в LOC)

C - фактор среды, зависящий от состояния технологии

K - общие трудозатраты для всего проекта

t_d - ограничения времени поставки (график), выраженные в годах

Фактор среды может быть вычислен следующим образом:

$$C = S / K^{1/3} * t_d^{4/3}$$

Коэффициенты ***K*** и t_d определяются на базе хронологических данных, относящихся к предыдущим проектам с размером S. Настраиваемое значение C может применяться для будущих оценок.

Технологическая константа C объединяет эффект использования инструментов, языков программирования, методологии, процедур гарантирования качества, стандартов и т.д. Она определяется на основе хронологических данных (прошлые проекты).

Значения технологической константы могут варьироваться от 610 до 57314. Константа C определяется, исходя из размера проекта, размера области под кривой трудозатрат, а также длительности проекта.

Оценка: C = 2000 - плохо, C = 8000 - хорошо, C = 11000 - превосходно

Например, предположим, что значение технологической константы C равно 4000 (среднее значение), а размер ПО оценивается величиной в 200000 LOG. А теперь подставим эти значения

в формулу:

общие трудозатраты жизненного цикла B = $(1/T^4)$ *$(S/C)^3$

общие трудозатраты жизненного цикла B = $(1/T^4)$ *$(200000/4000)^3$ = $(1/T^4)$ *$(50)^3$ трудозатраты на разработку E = 0,3945 *B

Если период целевой разработки равен 2 годам, то

Общие трудозатраты жизненного цикла B = (1/16) * $(50)^3$ = 7812,5 чел.-лет.

Трудозатраты на разработку E = 0,3945* B = 3082 человеко-лет

Согласно рекомендациям Патнама, значение C для различных типов проектов будет следующим:

- внедренный в режиме реального времени - 1500;

- пакетная разработка - 4894;

- поддерживаемый и организованный - 10040.

В случае изменения времени разработки в промежутке между двумя и тремя годами, трудозатраты и производительность варьируются следующим образом (см. рис. 17.10).

T	E	B	T	E	B	T	E	B
Время разработки, год	трудозатраты на разработку	Общие трудозатраты жизненного цикла						
2	3082	7812,5	2,5	1262	3200	3	609	1513

Рисунок 17.10 - Изменение трудозатрат в случае изменения времени выполнения проекта с 2 до 3-х лет

Преимущества модели SLIM

- поддерживает исчерпывающий набор инструментов менеджмента разработки ПО, выполняющих поддержку программ на протяжении всего жизненного цикла;

- способствует приобретению "хороших привычек" членами команды инжиниринга и менеджмента (планирование программных проектов, отслеживание программных проектов и надзор над областями ключевых процессов на уровне 2 SEI CMM);

- предлагает эффективное планирование с добавлением значений, особенно при работе с большими проектами;

- использует линейное программирование, статистическое моделирование, оценку программ, а также техники обзора, применяемые при оценке затрат на разработку ПО (благодаря применению метода линейного программирования пользователь может "навязать" максимум затрат, максимально возможное соблюдение графика, верхние и нижние границы оценки численности персонала, а также располагать приемлемым диапазоном времени и результативными решениями по трудозатратам.);

- позволяет выбирать "разработку проекта по затратам", если пользователь выбирает ввод размера и желаемого количества человеко-месяцев; модель позволяет также оценщику затрат, понесенных при разработке ПО, выполнять следующие функции:

калибровка - точно настраивайте модель с целью представления локальной среды программной разработки путем интерпретации хронологической базы для прошлых проектов;

построение - создайте информационную модель программной системы, объединив характеристики ПО, атрибуты персонала, атрибуты компьютера и т.д.;

измерение ПО - Используется автоматизированная версия техники оценки затрат с помощью LOC;

- позволяет организации настраивать фазы жизненного цикла и ключевые стадии для данной среды;

- упрощает стратегический процесс принятия решений;

- поддерживает оптимальную политику управления персоналом в контексте среды разработки, реализует анализ типа "что-если"; генерирует отчеты и графики для: месячных профилей персонала; месячных профилей бюджета; профилей риска для затрат, графика и трудозатрат; прогнозов надежности ПО; прогнозов начальной и конечной ключевых стадий;

- поддерживает информацию о количестве дефектов; - в модели SLIM выводится минимальное время, а также соответствующие затраты (в том числе трудозатраты), включая анализ чувствительности, в ходе выполнения которого показывается, как изменяются значения, т.е. подобно тому, как варьируются оценки размеров в трех стандартных реализациях.

Недостатки модели SLIM

- ее лучше всего использовать при работе с большими проектами (когда размер кода превышает 5000 строк, трудозатраты больше 1,5 человеко-лет, а время разработки превышает 6 месяцев);

- для использования модели необходимо заранее определить размер ПО;

- оценки являются сверхчувствительными к технологическому фактору;

- модель очень чувствительна к времени поставки (td);

- модель очень чувствительна к оценке размера;

- предполагается использование жизненного цикла каскада, который не отображается на инкрементальный итеративный (спиральный) процесс разработки либо на рациональный унифицированный процесс (Rational Unified Process);

- пользователи должны помнить о необходимости добавления фаз и интегральных задач, таких как программный менеджмент;

- этот инструмент является сложным;

- вряд ли будет возможным с его помощью за 2-5 минут модифицировать модель;

- при использовании модели часто получается так, что общая сумма трудозатрат при выполнении малых проектов будет меньшей, чем трудозатраты при выполнении большого проекта. Поэтому требуется проявлять осторожность при разбиении больших проектов на меньшие структурные единицы, учитывая при этом существование интерфейсов.

Резюме

При оценке размера программного продукта, трудозатрат персонала, графика проекта и остальных затрат наблюдаются сложные взаимосвязи с процессом планирования проекта, который первый раз происходит в начале жизненного цикла разработки проекта, а затем несколько раз повторяется (обычно к концу каждой основной фазы). Точность оценки обычно далека от идеала в начале осуществления проекта, однако она улучшается по мере сбора сведений о проекте на каждой фазе, в результате чего точность оценки постоянно повышается, совпадая в итоге фактическими значениями размера, трудозатрат, графика и остальных затрат.

Необходимо оценить трудозатраты (человеко-часы) и длительность (календарные дни) выполняемого проекта, благодаря чему менеджеры смогут определять затраты на производство продукта, срок возврата инвестиций, время выхода на рынок и качество. Процесс оценки зачастую является затруднительным, поскольку проекты часто должны удовлетворять взаимоисключающим требованиям (поддержка функциональных специфических свойств наравне со специфическими требованиями к производительности, учитывая специфические затраты и график, а также некоторый желаемый уровень качества). Множество ограничений приводит к усложнению процесса оценивания. Помимо этого, оценки производятся еще до того, как будут хорошо определены спецификации. Разработка архитектурного либо высокоуровневого проекта лишь намечает понятийную область в терминах поддержки исчисляемости и тестируемости требований, даже если первая оценка предшествует подобной деятельности.

Процесс оценивания обычно относится к области навыков менеджмента проектов, которая связана с документированием планов, оценкой затрат (трудозатрат), составлением графика и выбором метрических показателей. При осуществлении всех действий в рамках менеджмента программных проектов в "игру вступает" большинство навыков из области менеджмента персонала.

Организация, которая ведет документацию, следует инструкциям и постоянно улучшает процесс оценивания, будет соответствовать требованиям относительно ключевых процессов уровня 2 SEI CMM, а также требованиям планирования программных проектов. Описание цели 1 звучит следующим образом, "Программные оценки документированы с целью использования при планировании и отслеживании программных проектов".

Наибольшей проблемой при разработке ПО является создание полнофункционального высококачественного продукта в пределах бюджета, используя спрогнозированные активы. Эти ограничения называются парадигмой оценивания ПО.

Первый шаг на пути оценки программ заключается в определении размера программного продукта. Размер продукта обычно выражается в количестве KLOC, однако могут также применяться функциональные точки, точки свойств, объектные точки либо другие единицы измерения. Обычно перед переходом к следующему шагу оценивания (оценка затрат (трудозатрат) и затрат графика) происходит преобразование других единиц измерения в единицы измерения KLOC. Для любого современного широко применяемого языка программирования разработаны таблицы преобразования, позволяющие выполнять преобразование в единицы измерения LOC. Для оценки величины размера ПО используются следующие методы: Wideband Delphi, метод аналогии, экспертных оценок, анализ функциональных точек, анализ точек свойств, анализ объектных точек и т.д.

Для оценки трудозатрат (количества труда, использованного при создании программного продукта заданного размера) применяются эмпирические, регрессионные и математические модели. В главе рассматриваются только регрессионные модели.

Все модели оценки трудозатрат зависят от размера ПО. Если определены трудозатраты (обычно выраженные в человеко-месяцах), может быть оценен график и остальные затраты программного проекта. График обычно основывается на факторах производительности, количестве доступного персонала, а также фазе распределения трудозатрат.

Регрессионная модель COCOMO является наиболее широко используемой и известной среди всех моделей оценивания. В модели COCOMO поддерживается три режима (органический, сблокированный и внедренный), а также три уровня сложности (базовый, промежуточный и детализованный). "Режим" просто описывает тип проекта (большой, малый, простой, сложный и т.д.). "Уровень" описывает количество входных данных - с увеличением количества входных данных улучшается точность оценки. Разработчиком модели COCOMO является доктор Барри Боэм (Dr. Barry Boehm); свои исследования он начал в TRW, а затем продолжил их в Южно-Калифорнийском университете.

В базовой модели COCOMO используется формула, с помощью которой рисуется кривая, аппроксимирующая набор точек данных. Затем применяется простой регрессионный анализ. В промежуточной модели COCOMO усовершенствуется базовая модель путем добавления драйверов затрат (либо факторов среды), с помощью которых изменяется величина трудозатрат при разработке программного продукта. В детализированной модели COCOMO поддерживаются дополнительные аналитические инструменты, а оценка производится в соответствии с уровнями структуры WBS (трехуровневая иерархия: система, подсистема, модуль). Также на каждой фазе жизненного цикла применяются корректировочные формулы. В большинстве случаев

применяется промежуточная модель, поскольку в этом случае могут применяться электронные таблицы. Для получения оценок могут применяться многие автоматизированные инструменты.

Модель COCOMO, подобно другим моделям и инструментам оценивания, имеет свои преимущества и недостатки. Ей присущи простота в применении, а также общий язык, применяемый для общения в сообществе специалистов в области планирования /менеджмента программных проектов. С другой стороны, эта модель может принести реальную пользу, если она будет калибрована на базе хронологических данных организации. Возможно одним из наибольших недостатков этой модели является то, что она основана на использовании оценки размера программного продукта, выраженной в LOC. Однако ни доктор Боэм (Dr. Boehm), ни создатели других моделей и инструментов не предложили лучшего способа для начала- даже если оценивание начинается с применения функциональных свойств вместо размеров, а трансляция функций в значения размера происходит до продолжения оценивания затрат (трудозатрат) и затрат графика.

Модель COCOMO может быть настроена с учетом нужд конкретной организации. В процессе подобной настройки происходит калибровка на основе общедоступных хронологических данных и формул. Соответствующие сведения можно найти в классической книге Боэма, Software Engineering Economics.

Модель COCOMO II является наиболее современной: Она поддерживает эволюционный, управляемый рисками и совместный программный процесс; языки четвертого поколения и генераторы приложений; подходы с применением коммерческих готовых продуктов и управляемого методом повторного использования ПО; подходы с разработкой ПО быстрого отслеживания; инициатив но обеспечению зрелости программных процессов.

Применяется также другая математическая модель оценивания, называемая менеджментом жизненного цикла разработки ПО (Software Lifecycle Management, SLIM). Этот инструмент имеет отношение к QSM. Благодаря использованию модели SLIM специалист по оценке может использовать анализ линейного программирования, в котором рассматриваются ограничения разработки, имеющие отношение к затратам (трудозатратам), и поддерживается помесячное распределение трудозатрат и проверка совместимости для данных, имеющих отношение к программным системам одного размера. Модель SLIM основана на анализе жизненных циклов разработки ПО, производимого в терминах распределения Рейлайха (Rayleigh), связывающего уровень квалификации персонала и затраченное время. Эта модель была разработана Патнамом

Практика

Решение примеров

Дано размер проекта :

а) 1 KLOC.

б) 55 KLOC.

в) 350 KLOC.

Для каждого случай а), б) и в) на основе таблицы рис. 17.1. и используя базовую и промежуточную модель COCOMO определить:

1. Оценку трудозатрат (E).
2. Время разработки (TDEV).
3. Среднюю численность персонала (SS).
4. Производительность.

Данные занести в таблицы рис. 17.3 и рис. 17.6 согласно примерам 3 и 4 соответсвенно.

Вопросы для самоконтроля

1. Что включают для оценки трудозатрат?
2. Режимы модели COCOMO.
3. Уровни модели COCOMO.
4. Основные характеристики регрессионной модели COCOMO.
5. Основные характеристики базовой модели COCOMO.
6. Основные характеристики промежуточной модели COCOMO.
7. Основные характеристики детализированной модели COCOMO.
8. Преимущества и недостатки модели COCOMO.
9. Что такое драйвер затрат?

10. Категории и группы драйверов затрат.
11. Основные характеристики модели COCOMO II.
12. Математическая модель SLIM.
13. Преимущества и недостатки модели SLIM

Вывод к разделу 17 - Оценка длительности и стоимости разработки ПО (Модели COCOMO и SLIM)

В разделе рассматривались:

- конструктивная модель затрат-Constructive COst Model - COCOMO, которая применяется для оценки трудозатрат, графика и остальных затрат проекта;

- модель управления процессом жизненного цикла ПО QSM (QSM's Software Lifecycle Management, SLIM), которая состоит из методологий, связанных воедино с помощью инструментов принятия решений: SLIM-Estimate©, SLIM-Control© и SLIM-Metrics©. Инструмент SLIM-Esrimate© поддерживает оценивание и планирование, SLIM-Control© - отслеживание и прогнозирование, а SLIM-Metrics©-захват данных и анализ.

Перечень ссылок

Источники, использованные в материалах

Американский национальный стандарт ANSI/PMI 99-001-2004. Руководство к Своду знаний по управлению проектами. Введ. 2004.- Третье издание. (Руководство PMBOOK®). 401с.

Мари Кантор. Управление программными проектами. Практическое руководство по разработке успешного программного обеспечения . СПБ. Вильямс. 2002. -642с.

Управление программными проектами. Достижение оптимального качества при минимуме затрат. Роберт, Т. Фатрелл, Дональд Ф. Шафер, Линда И. Шафер / М-СПб-К. Вильямс. 2003. -1118с.

Уокер Ройс. Управление проектами по созданию программного обеспечения . М. Лори. 2002. -450с.

Элейн Маркел. Microsoft Project 2002. Библия пользователя. М. Диалектика. 2003. -880с.

Microsoft Project 2003 course certification materials. Trainer kit. [Электронный ресурс] Режим доступа: http://www.cheltenhamcourseware.com/