Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана
Каф. ИУ-7

baryshnikovam@mail.ru

Лекции 6_7

Экономическая модель разработки ПО. Оценка технико-экономических показателей проекта. Оценка размера программного продукта в строках программного кода. Модели COCOMO и COCOMO II. Влияние эффектов повторного использования кода на размер ПО

Несоответствие производительности изначально предполагаемым показателям может иметь две следующие причины: плохая работа или некорректная оценка. В мире программного обеспечения можно получить множество свидетельств плохо выполненной оценки, однако практически невозможно доказать, что персонал не трудился усердно над разрешением проблемы, либо не является в достаточной степени компетентным

Том Де Марко

Технико-экономическое обоснование программного проекта

представляет собой процедуру оценивания трудовых, временных и финансовых ресурсов по созданию программного продукта, соответствующего требованиям заказчика

Объем требуемых ресурсов зависит от:

- совокупности бизнес-процессов, описывающих предметную область, и их приоритетов для заказчиков
- требований к функциональной полноте и качеству реализации каждого бизнес-процесса

Две части жизненного цикла программных средств

- П*ервая часть* – включает в себя работы по системному анализу, проектированию, разработке, тестированию и испытаниям базовой версии программного продукта
- *Вторая часть* - отражает эксплуатацию, сопровождение, модификацию, управление конфигурацией и перенос ПС на иные платформы

Программные продукты являются одними из наиболее сложных объектов, создаваемых человеком, и в процессе их производства творчество специалистов в контексте поиска новых методов, альтернативных решений и способов реализации заданных требований, а также формирование и декомпозиция этих требований составляют значительную часть всех трудозатрат

Исходные данные, используемые для прогнозирования и планирования

- функции и номенклатура характеристик самого прогнозируемого объекта или процесса
- характеристики прототипов и пилотных проектов, в некоторой степени подобных планируемому объекту, которые уже завершились и в отношении которых известны необходимые экономические характеристики и можно оценить качество процессов планирования и прогнозирования

Типовое распределение стоимости между основными этапами жизненного цикла (без сопровождения)

- 25% - проектирование – разработка и верификация проекта
- 15% - спецификации – формулировка требований и условий разработки
- 20% - разработка – кодирование и тестирование компонент
- 40% - интеграция и тестирование

Причины для сравнения реальных данных с прогнозируемыми оценками характеристик проекта

- несовершенство исходных данных при оценивании технико-экономических показателей программных проектов вызывает необходимость периодического пересмотра прогнозных оценок с учетом новой информации, чтобы обеспечить более реальную основу для дальнейшего управления проектом

- вследствие несовершенства методов оценивания технико-экономических показателей программных проектов следует сравнивать прогнозные оценки этих показателей с действительными значениями, формирующимися в ходе реализации проекта, и использовать эти результаты для улучшения самих методов оценивания

- программные проекты имеют тенденцию к изменению характеристик и экономических факторов, поэтому для их успеха необходимо идентифицировать эти изменения и выполнять реалистичное обновление оценок затрат

Факторы, повышающие точность оценок

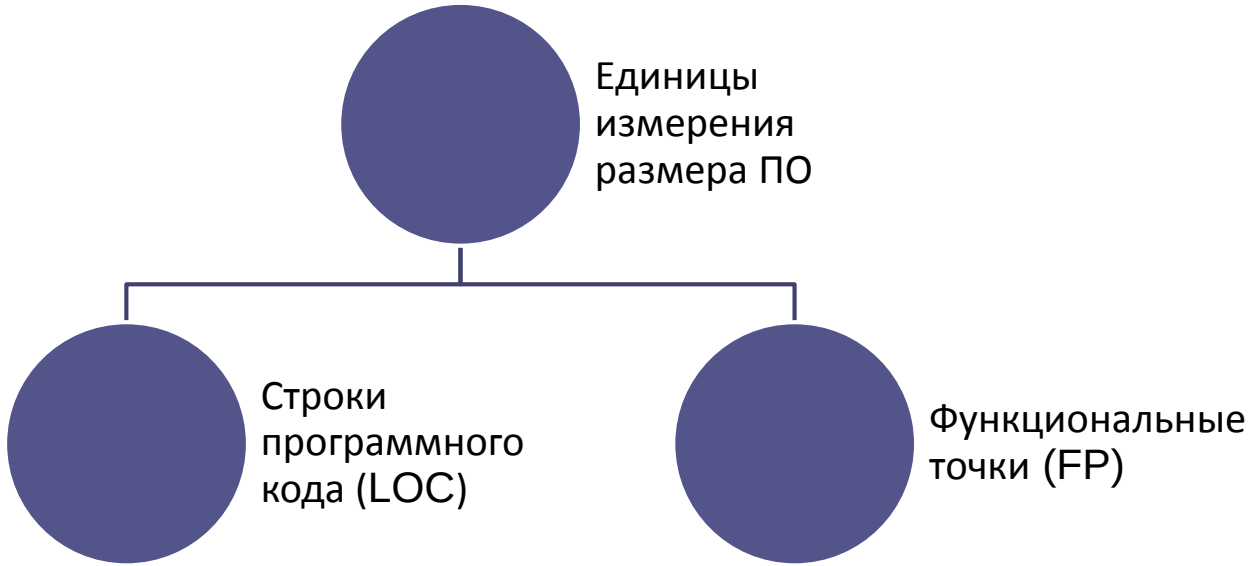
- цели оценивания технико-экономических показателей должны быть согласованы с потребностями в информации, способствующей принятию решений на соответствующем этапе программного проекта

- достоверность оценок должна быть сбалансирована для различных компонентов системы и величина уровня неопределенности для каждого компонента должна быть примерно одинаковой, если в процессе принятия решения все компоненты имеют одинаковый вес

- следует возвращаться к предшествующим целям оценивания технико-экономических показателей и изменять их, когда это необходимо для ответственных бюджетных решений, принимаемых на ранних этапах и влияющих на следующие этапы

Основные параметры оценки при создании ПП

- сложность (размеры)
- трудозатраты на разработку
- длительность разработки в целом и ее отдельных этапов
- численность и квалификация специалистов, привлекаемых к созданию ПП



Проблемы использования LOC в качестве единицы измерения размера программного продукта

- число строк исходного кода зависит от уровня мастерства программиста. Фактически, чем выше мастерство программиста, тем меньшим количеством строк кода ему удастся обойтись для реализации определенной функциональной возможности (или функциональности) ПС
- высокоуровневые языки или языки визуального программирования требуют гораздо меньшего числа строк кода для отражения одной и той же функциональности, чем, например, язык Ассемблера или С. Очевидно, что существует обратная взаимосвязь между уровнем языка и производительностью труда (количеством строк кода в день) программиста
- фактическое число строк программного кода остается неизвестным до тех пор, пока проект не будет почти завершен. Поэтому LOC сложно использовать для предварительной оценки трудозатрат на разработку и построения плана-графика проекта
- в программистском сообществе не достигнуто соглашения о методе подсчета строк кода. Языковые конструкции, используемые, например, в Visual C++, Ассемблере, Коболе или SQL существенно различаются. Метод же остается общим для любых приложений, в том числе использующих комбинацию различных языков
- заказчику сложно понять, каково соотношение указанных им функциональных и нефункциональных (технических) требований к ПС и объемов программистской работы

Рекомендации по повышению достоверности LOC - оценок

- убедитесь, что каждая учитываемая строка исходного кода содержит лишь один оператор. Если в одной строке содержатся два выполняемых оператора, разделенных точкой с запятой, то они должны учитываться как две строки. Если же один оператор разбит на несколько «физических» строк, он будет учитываться как одна строка. В языках программирования допускаются различные правила кодирования, но обычно проще определять в строке один оператор, обрабатываемый компилятором или интерпретатором
- учитывайте все выполняемые операторы. Конечный пользователь может не иметь возможности практически использовать каждый оператор, но все операторы должны поддерживаться данным продуктом, в том числе и утилитами
- определения данных учитывайте лишь один раз
- не учитывайте строки, содержащие комментарии
- не учитывайте отладочный код либо другой временный код (пробное ПО, средства тестирования и пр.)
- учитывайте каждую инициализацию, вызов или включение макроса (директивы компилятора) в качестве части исходного кода, в которой осуществляется какое-либо действие. Не учитывайте повторно используемые операторы

Пример оценки показателя LOC с помощью экспертных оценок

Некоторый объект, отображенный на структуре WBS, по мнению экспертов может занимать от 200 до 400 строк кода, но скорее всего, его размер ближе к 250 строкам.

Используя метод PERT, размер объекта можно оценить в 266 строк:

(200+(250*4)+400)/6 = 266 LOC

Оценка количества LOC по аналогии

Например, у нас имеется уже готовый модуль A, размер которого составляет 2345 LOC. Мы хотим создать новый модуль A’, который будет во многом схож с модулем A, но в него будут добавлены некоторые дополнительные свойства. Кроме того, мы знаем как сделать программный код более компактным. В результате этого размер модуля A’ может быть оценен в 3000 LOC

Использование языка ассемблера для сравнительного анализа различных проектов

Язык программирования	**Basic Assembler**
Ассемблер	1
C	2,5
Кобол	3
Фортран	3
Паскаль	3,5
C++	6
Java	6
Ada 95	6,5
Access	8,5
Delphi Pascal	11
CORBA	16

Пусть решается задача по переводу некоторой операционной системы, написанной на языке C и содержащей 50000 строк кода на язык C++

Операционная система, содержащая 50000 строк кода на языке C, эквивалентна 125000 строкам кода на языке ассемблера, которые, будучи переписаны на C++, составят 20833 LOC

Примечание:
50000*2,5 = 125000
125000/6 = 20833

Преимущества использования LOC в качестве единицы измерения

- Эти единицы широко распространены и могут адаптироваться
- Они позволяют проводить сопоставление методов измерения размера и производительности в различных группах разработчиков
- Они непосредственно связаны с конечным продуктом
- Единицы LOC могут быть оценены еще до завершения проекта
- Оценка размеров ПО производится с учетом точки зрения разработчиков
- Действия по непрерывному улучшению базируются на количественных оценках. При этом спрогнозированный размер может быть легко сопоставлен с реальным размером на этапе постпроектного анализа. Это позволяет экспертам накапливать опыт и улучшать сами методы оценки
- Знание размера программного продуктам в LOC – единицах позволяет применять большинство существующих методов оценки технико-экономических показателей проекта (таких как трудозатраты, длительность проекта, его стоимость и др.)

Недостатки, связанные с применением LOC – оценок

- Данные единицы измерения сложно применять на ранних стадиях жизненного цикла, когда высок уровень неопределенности
- Исходные инструкции могут различаться в зависимости от языков программирования, методов проектирования, стиля и способностей программистов
- Применение методов оценки с помощью количества строк не регламентируется промышленными стандартами, например ISO
- Разработка ПО может быть связана с большими затратами, которые напрямую не зависят от размеров программного кода (это затраты, связанные с разработкой спецификации требований, подготовкой пользовательской документации и пр., которые не включены в прямые затраты на кодирование)
- При подсчете LOC – единиц следует различать автоматически сгенерированный код и код, написанный вручную, что сильно затрудняет применение автоматических методов подсчета
- Генераторы кода зачастую провоцируют его избыточный объем, что может привести к значительным погрешностям в оценке размера ПП
- Единственным способом получения LOC – оценки является сравнение с аналогичными разработками или экспертные мнения, а эти методы изначально не относятся к числу точных

Способ учета качества программного кода

Программисты могут быть незаслуженно премированы за достижение высоких показателей LOC, если служба менеджмента посчитает это высоким признаком продуктивности

При этом показатели LOC не могут осуществляться для сравнения производительности различных команд разработчиков (либо для нормирования труда) в случае, если использовались разные платформы или типы языков программирования

Количество дефектов / количество строк кода

Софтверные организации склонны вознаграждать программистов, которые: а) пишут много кода; б) исправляют много ошибок. Соответственно, наилучший способ отличиться в таких условиях – это создать большое количество некачественного кода, а потом героически устранять в нем собственные же промахи

Джоэль Спольски (Joel Spolsky) - руководитель компании Fog Creek Software

Экономическая модель разработки ПО

Трудозатраты= (Персонал)(Среда)(Качество)(Размер $^{\text{Процесс}}$)

Размер – размер конечного продукта (для компонентов, написанных вручную), который обычно измеряется числом строк исходного кода или количеством функциональных точек, необходимых для реализации данной функциональности. В это понятие также должны входить и другие создаваемые материалы, такие как документация, совокупность тестовых данных и обучающие материалы

Персонал – возможности персонала, участвующего в разработке ПО, в особенности его профессиональный опыт и знание предметной области проекта. Источниками сложностей могут быть требуемая надежность программного обеспечения, ограничения на производительность и хранение, требуемое повторное использование программных компонентов, а так же опыт работы программистов с данной средой программирования

Среда – состоит из инструментов и методов, используемых для эффективной разработки ПО и автоматизации процесса. Т.е. фактически, это приобретенная или потерянная эффективность вследствие уровня автоматизации процесса (больший уровень автоматизации приводит к уменьшению усилий и повышению эффективности)

Качество – требуемое качество продукта, что включает в себя его функциональные возможности, производительность, надежность и адаптируемость

Процесс – особенности процесса, используемого для получения конечного продукта, в частности, его способность избегать непроизводительных видов деятельности: переделок, бюрократических проволочек, затрат на взаимодействие

Алгоритм оценки затрат на разработку ПО

- оценка размера разрабатываемого продукта
- оценка трудозатрат (трудоемкости) в человеко-месяцах или человеко-часах
- оценка продолжительности проекта в календарных месяцах
- оценка стоимости проекта

Основным интегральным экономическим показателем каждого программного проекта, отражающим суммарные затраты интеллектуального труда специалистов на производство программного продукта, является трудозатраты (трудоемкость)
Полный анализ и оптимизацию суммарных затрат на проект целесообразно проводить на всем протяжении жизненного цикла, при этом в ряде случаев очень важно учитывать в том числе затраты на сопровождение и эксплуатацию программного продукта
Эти виды затрат характеризуются значительной неопределенностью из-за сложности прогнозирования длительности жизненного цикла, требований качества, степени модификации программ и затрат на сопровождение

Модель оценки стоимости COCOMO (COnstructive COst MOdel — конструктивная модель стоимости)

Трудозатраты= C1* EAF *(Размер)^{p1}

Время = C2*(Трудозатраты)^{p2}

Трудозатраты (работа) — количество человеко-месяцев;

C1 — масштабирующий коэффициент

EAF — уточняющий фактор, характеризующий предметную область, персонал, среду и инструментарий, используемый для создания рабочих продуктов процесса

Размер — размер конечного продукта (кода, созданного человеком), измеряемый в исходных инструкциях (DSI, delivered source instructions), которые необходимы для реализации требуемой функциональной возможности

P1 — показатель степени, характеризующий экономию при больших масштабах, присущую тому процессу, который используется для создания конечного продукта; в частности, способность процесса избегать непроизводительных видов деятельности (доработок, бюрократических проволочек, накладных расходов на взаимодействие)

Время — общее количество месяцев

C2 — масштабирующий коэффициент для сроков исполнения

P2 — показатель степени, который характеризует инерцию и распараллеливание, присущие управлению разработкой ПО

Допущения модели COCOMO

- Исходные инструкции конечного продукта включают в себя все (кроме комментариев) строки кода, обрабатываемого компьютером
- Начало жизненного цикла проекта совпадает с началом разработки продукта, окончание — совпадает с окончанием приемочного тестирования, завершающего стадию интеграции и тестирования
- Работа и время, затрачиваемые на анализ требований, оцениваются отдельно, как дополнительный процент от разработки в целом
- Виды деятельности включают в себя только работы, направленные непосредственно на выполнение проекта
- Человеко-месяц состоит из 152 часов
- Проект управляется надлежащим образом, в нем используются стабильные требования

Режимы модели COCOMO

Название режима	Размер проекта	Описание	Среда разработки
Обычный	До 50 KLOC	Некрупный проект разрабатывается небольшой командой, для которой нехарактерны нововведения, разработчики знакомы с инструментами и языком программирования	Стабильная
Промежуточный	50 – 500 KLOC	Относительно небольшая команда занимается проектом среднего размера, в процессе разработки необходимы определенные инновации	Среда характеризуется незначительной нестабильностью
Встроенный	Более 500 KLOC	Большая команда разработчиков трудится над крупным проектом, необходим значительный объем инноваций	Среда состоит из множества нестабильных элементов

Формулы для оценки основных работ и сроков

Обычный вариант

Трудозатраты = $3,2*EAF*(Размер)^{1,05}$

Время (в месяцах) = $2,5*(Трудозатраты)^{0,38}$

Промежуточный вариант

Трудозатраты = $3,0*EAF*(Размер)^{1,12}$

Время (в месяцах) = $2,5*(Трудозатраты)^{0,35}$

Встроенный вариант

Трудозатраты = $2,8*EAF*(Размер)^{1,2}$

Время (в месяцах) = $2,5*(Трудозатраты)^{0,32}$

Трудозатраты (работа) — количество человеко-месяцев

EAF — результат учета 15 уточняющих факторов (см. таблицу)

Размер — число исходных инструкций конечного продукта (измеряемое в тысячах строк кода KLOC)

Значение драйверов затрат в модели COCOMO

Идентификатор	Уточняющий фактор работ	Диапазон изменения параметра	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий
Атрибуты программного продукта							
RELY	Требуемая надежность	0,75-1,40	0,75	0,86	1,0	1,15	1,4
DATA	Размер базы данных	0,94-1,16		0,94	1,0	1,08	1,16
CPLX	Сложность продукта	0,70-1,65	0,7	0,85	1,0	1,15	1,3
Атрибуты компьютера							
TIME	Ограничение времени выполнения	1,00-1,66			1,0	1,11	1,50,
STOR	Ограничение объема основной памяти	1,00-1,56			1,0	1,06	1,21
VIRT	Изменчивость виртуальной машины	0,87-1,30		0,87	1,0	1,15	1,30
TURN	Время реакции компьютера	0,87-1,15		0,87	1,0	1,07	1,15
Атрибуты персонала							
ACAP	Способности аналитика	1,46-0,71	1,46	1,19	1,0	0,86	0,71
AEXP	Знание приложений	1,29-0,82	1,29	1,15,	1,0	0,91	0,82
PCAP	Способности программиста	1,42-0,70	1,42	1,17	1,00	0,86	0,7
VEXP	Знание виртуальной машины	1,21-0,90	1,21	1,1	1,0	0,9	
LEXP	Знание языка программирования	1,14-0,95	1,14	1,07	1,0	0,95	
Атрибуты проекта							
MODP	Использование современных методов	1,24-0,82	1,24	1,1	1,0	0,91	0,82
TOOL	Использование программных инструментов	1,24-0,83	1,24	1,1	1,0	0,91	0,82
SCED	Требуемые сроки разработки	1,23-1,10	1,23	1,08	1,0	1,04	1,1

Распределение работ и времени по стадиям жизненного цикла

Вид деятельности	***Трудозатраты (%)***	***Время (%)***
Планирование и определение требований	(+8)	(+36)
Проектирование продукта	18	36
Детальное проектирование	25	18
Кодирование и тестирование отдельных модулей	26	18
Интеграция и тестирование	31	28

Примечание: В основе распределения трудозатрат и времени лежит каскадная модель жизненного цикла

Декомпозиция работ по созданию ПО

Вид деятельности	***Бюджет (%)***
Анализ требований	4
Проектирование продукта	12
Программирование	44
Тестирование	6
Верификация и аттестация	14
Канцелярия проекта	7
Управление конфигурацией и обеспечение качества	7
Создание руководств	6
Итого	**100**

Пример использования модели COCOMO

По контракту с бюджетной организацией разрабатывается большая, критически важная система (например, для управления электростанцией). Объем программного кода был предварительно оценен в 100 000 строк (100 KSLOC). Используемая технология является новой для разработчиков.

Произвести оценку параметров по методике COCOMO

Все драйверы затрат номинальные, кроме:

Фактор, влияющий на стоимость	**Идентификатор**	**Значение**	**Значение параметра**
Использование программных инструментов	TOOL	Высокое	0,88
Знание приложений	AEXP	Низкое	1,1
Требуемая надежность	RELY	Высокое	1,15
Сложность продукта	CPLX	Высокое	1,15
EAF			**1,28**

Расчеты по методике COCOMO

- Трудозатраты $=2,8*EAF*(KDSI)^{1,2}=2,8*1,28*(100)^{1,2}=$ 900 человеко-месяцев на разработку+72 человеко-месяца на планирование, определение требований=972 человеко-месяца
- Время$=2,5*(\text{Трудозатраты})^{0,32}=2,5*(900)^{0,32}=22$ месяца на разработку + 8 месяцев на планирование, определение требований = 30 месяцев

Учитывая критическую важность и сложность системы для расчетов использовался встроенный вариант

Распределение работ по этапам жизненного цикла

Вид деятельности	***Бюджет (%)***	***Человеко-месяцы***
Анализ требований	4	39
Проектирование продукта	12	117
Программирование	44	428
Тестирование	6	58
Верификация и аттестация	14	136
Канцелярия проекта	7	68
Управление конфигурацией и обеспечение качества	7	68
Создание руководств	6	58
Итого	***100***	***972***

Достоинства модели COCOMO

- Метод является достаточно универсальным и может поддерживать различные режимы и уровни программных разработок
- При расчетах используются множители и показатели степени, полученные на основе анализа данных большого количества практически реализованных проектов
- Предложенные драйверы затрат хорошо подгоняются под специфику конкретной организации
- Точность оценок повышается по мере накопления в организации опыта применения модели
- Метод снабжен обширной документацией и прост в применении

Недостатки модели COCOMO

- Все уровни зависят от оценки размера – точность оценки размера оказывает влияние на точность оценки трудозатрат, времени разработки, подбор персонала и оценку производительности
- Метод основан на каскадной модели жизненного цикла и прежде всего не учитывает изменяемость требований
- Слишком поверхностное внимание уделено вопросам обеспечения безопасности и надежности
- Модель не учитывает возможности повторного использования кода, итерационные возвраты по этапам жизненного цикла, объектно-ориентированные технологии разработки ПО

Концепция повторного использования

- Многие программы происходят от предыдущих версий этих же программ
- В результате может быть достигнута экономия средств и/или времени на разработку, а также повышен уровень качества ПО
- Повторному использованию подлежат: программный код, тестовый код, тестовые процедуры, документация, дизайн, спецификации требований и др.
- Код, повторно используемый в полном объеме, имеет идентичную документацию, идентичные тестовые процедуры и сам тестовый код и только одну копию, поддерживаемую системой управления конфигурацией

Повторное использование кода

- Новый код – это код, разработанный для нового приложения, который не включает большие порции ранее написанного кода
- Модифицируемый код – это код, разработанный для предыдущих приложений, который будет пригоден для использования в новом приложении после внесения умеренного объема изменений
- Повторно используемый код – это код, разработанный для предыдущих приложений, который будет пригодным для новых приложений без внесения каких-либо изменений

Первый этап при оценке систем, в которых возможно повторное использование кода, заключается в отделении нового кода от модифицируемого и повторно используемого кода. Это необходимо, так как интеграция модифицируемого и повторно используемого кода требует дополнительного объема трудозатрат

Рекомендации по выделению повторно используемого кода

- В качестве оцениваемой единицы выбирается программный модуль (примерный размер около 100 LOC)
- Если единица не была изменена она принимается за повторно используемый код
- Если единица была изменена (неважно идет ли речь об исполняемом коде, либо о комментариях) она считается модифицируемой
- Если объем изменений затрагивает более 50% кода единицы она принимается за новую
- В модифицируемом коде различают изменения, направленные на устранение дефектов и изменения, направленные на расширение функциональности

Пример, показывающий распределение новых модифицируемых и повторно используемых строк кода

Элемент	LOC для нового кода	LOC для модифицируемого кода	LOC для повторно используемого кода	Итого LOC
Компонент 1	1233	0	0	1233
Компонент 2	0	988	0	988
Компонент 3	0	0	781	781
Компонент 4	560	245	0	805
Компонент 5	345	549	420	1314
ИТОГО	2138	1782	1201	5121

Типичные множители повторного использования кода

Степень простоты использования	Повторно используемый код	Модифицируемый код
Простой	10%	25%
Средний	30%	60%
Сложный	40%	75%

Применение множителей повторного использования кода

Элемент	LOC для нового кода	LOC для модифицируемого кода	LOC для повторно используемого кода	Итого LOC
Итого	2138	1782	1201	5121
Множитель	100%	60%	30%	
Результат	2138	1069	360	3567

COCOMO II

Проект COCOMO II стартовал в 1995 году в центре по разработке ПО USC при финансовой и технической поддержке большого количества промышленных предприятий, таких как AT&T, BELL Labs, Hewlett-Packard, Rational, Texas Instruments, Lockheed Martin, Motorola, Xerox и др.

Проект имел триединую задачу:

- разработать модель для оценки стоимости и сроков создания ПО для того жизненного цикла, который будет применяться в конце 20 – начале 21 века
- разработать базу данных стоимости программных проектов и осуществить инструментальную поддержку методов усовершенствования модели стоимости
- создать количественную аналитическую схему для оценки технологий создания ПО и их экономического эффекта

Три различные модели оценки стоимости в COCOMO II

- Модель композиции приложения – это модель, которая подходит для проектов, созданных с помощью современных инструментальных средств. Единицей измерения служит объектная точка
- Модель ранней разработки архитектуры. Эта модель применяется для получения приблизительных оценок проектных затрат периода выполнения проекта перед тем как будет определена архитектура в целом. В этом случае используется небольшой набор новых драйверов затрат и новых уравнений оценки. В качестве единиц измерения используются функциональные точки либо KSLOC
- Постархитектурная модель – наиболее детализированная модель COCOMO II, которая используется после разработки архитектуры проекта. В состав этой модели включены новые драйверы затрат, новые правила подсчета строк кода, а также новые уравнения

Характеристики моделей оценки стоимости COCOMO II

Модель композиции приложения	**Модель ранней разработки архитектуры**	**Постархитектурная модель**
Грубые входные данные	Ясно понимаемые особенности проекта	Детальное описание проекта
Оценки низкой точности	Оценки умеренной точности	Высокоточные оценки
Приблизительные требования	Ясно понимаемые требования	Стабилизировавшиеся основные требования
Концепция архитектуры	Ясно понимаемая архитектура	Стабильная базовая архитектура

Модель композиции приложения

Данная модель используется на ранней стадии конструирования ПО, когда:

- рассматривается макетирование пользовательских интерфейсов
- оценивается производительность
- определяется степень зрелости технологии

Модель ориентирована на применение объектных точек. Объектная точка — средство косвенного измерения ПО. Подсчет количества объектных точек производится с учетом количества экранов (как элементов пользовательского интерфейса), отчетов и компонентов, требуемых для построения приложения

Правила подсчета объектных точек

- количество изображений на дисплее (экранных форм). Простые изображения принимаются за 1 объектную точку, изображения умеренной сложности принимаются за 2 точки, очень сложные изображения принято считать за 3 точки
- количество представленных отчетов. Для простых отчетов назначаются 2 объектные точки, умеренно сложным отчетам назначаются 5 точек. Написание сложных отчетов оценивается в 8 точек
- количество модулей, которые написаны на языках третьего поколения и разработаны в дополнение к коду, написанному на языке программирования четвертого поколения. Каждый модуль на языке третьего поколения считается за 10 объектных точек

Модель композиции приложения

NOP = (Объектные точки) x [(100 - %RUSE) /100] – новые объектные точки

ТРУДОЗАТРАТЫ = NOP /PROD [чел.-мес.]

PROD - оценка скорости разработки

Модель композиции приложения используется на этапе создания прототипов и анализа осуществимости

Опытность/ возможности разработчика	**Зрелость/ возможности среды разработки**	**PROD**
Очень низкая	Очень низкая	4
Низкая	Низкая	7
Номинальная	Номинальная	13
Высокая	Высокая	25
Очень высокая	Очень высокая	50

Пример использования модели композиции приложения

Найти трудозатраты и календарное время работы над следующим проектом: приложение формирует 2 формы средней сложности (запросы к базе данных), 3 отчета высокой сложности и имеет 2 программных модуля на языке 3 уровня. Процент повторного использования кода программы – 10%. Над проектом работает программист средней квалификации, однако имеющий опыт работы в данной предметной области

1. Найдем количество объектных точек и их суммарную балльную оценку

№	Наименование объекта	Уровень сложности (коэф-нт)	Количество	Число точек
1	Форма	Средний (2)	2	4
2	Отчет	Высокий (8)	3	24
3	Модуль		2	20
всего			7	48

2. Определим трудозатраты на уровне композиции приложения, приняв среднюю производительность программиста 13 точек в месяц:
Трудозатраты = (NOP * (100 - RUSE) / 100)) / PROD = (48 * (100 – 10)/100) / 13 = 3,32 (чел./мес.)

3. Определим длительность выполнения проекта на уровне композиции приложения:
Время = 3 * (Трудозатраты)$^{(0.33 + 0.2 * (p-1.01))}$ = 3 * 3,32$^{(0,33 +0.2 * (p-1.01))}$ = 3 * 1,482 = 4,45 (месяца),

где p = 1

Модель ранней разработки архитектуры

Трудозатраты = 2,45*EArch*(Размер)^{p},

где

Трудозатраты (работа) — число человеко-месяцев

EArch = PERS * RCPX * RUSE * PDIF * PREX * FCIL * SCED

Размер — KSLOC (предпочтительно для подсчета KSLOC предварительно подсчитать количество функциональных точек)

P — показатель степени

Время = 3,0 * (Трудозатраты) $^{(0.33 + 0.2 * (p-1.01))}$

Примечание: Множитель EArch является произведением семи показателей, характеризующих проект и процесс создания ПО, а именно: надежность и уровень сложности разрабатываемой системы (RCPX), повторное использование компонентов (RUSE), сложность платформы разработки (PDIF), возможности персонала (PERS), опыт персонала (PREX), график работ (SCED) и средства поддержки (FCIL). Каждый множитель может быть оценен экспертно, либо его можно вычислить путем комбинирования значений более детализированных показателей, которые используются на постархитектурном уровне (см. след. слайд)

Модель ранней разработки архитектуры

Идентификатор	Составные драйверы затрат
Сложность продукта	RELY-DATA-CPLX-DOCU
Необходимость повторного использования	RUSE
Сложность платформы	TIME-STOR-PVOL
Опытность персонала	AEXP-PEXP-LTEX
Способности персонала	ACAP-PCAP-PCON
Возможности среды	TOOL-SITE
Сроки	SCED

Примечание: высокий опыт персонала и интенсивное повторное использование компонентов ведут к снижению затрат

Значения множителей трудоемкости, в зависимости от их уровня

	Оценка уровня множителя трудоемкости					
	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
PERS	1.62	1.26	1.00	0.83	0.63	0.5
RCPX	0.60	0.83	1.00	1.33	1.91	2.72
RUSE	n/a	0.95	1.00	1.07	1.15	1.24
PDIF	n/a	0.87	1.00	1.29	1.81	2.61
PREX	1.33	1.22	1.00	0.87	0.74	0.62
FCIL	1.30	1.10	1.00	0.87	0.73	0.62
SCED	1.43	1.14	1.00	1.00	1.00	n/a

Модель этапа постархитектуры

Трудозатраты = 2,45*EApp*(Размер)^{p},

где:

Трудозатраты (работа) — число человеко-месяцев;

EApp — результат применения семнадцати уточняющих факторов постархитектурных этапов разработки

Время = 3,0 * (Трудозатраты) $^{(0.33 + 0.2 * (p-1.01))}$

Усовершенствованная постархитектурная модель COCOMO II

Идентификатор	Уточняющий фактор работ	Изменение в COCOMO II
RELY	Требуемая надежность	Без изменений относительно COCOMO
DATA	Размер базы данных	Без изменений относительно COCOMO
CPLX	Сложность продукта	Без изменений относительно COCOMO
RUSE	Требуемый уровень повторного использования	
DOCU	Документация	Добавлен. Определяет насколько документация соответствует требованиям жизненного цикла
TIME	Ограничение времени выполнения	Без изменений относительно COCOMO
STOR	Ограничение объема основной памяти	Без изменений относительно COCOMO

Усовершенствованная постархитектурная модель COCOMO II

Идентификатор	Уточняющий фактор работ	Изменение в COCOMO II
PVOL	Изменчивость платформы	Фактор изменчивости платформы
ACAP	Способности аналитика	Без изменений относительно COCOMO
AEXP	Знание приложений	Без изменений относительно COCOMO
PCAP	Способности программиста	Без изменений относительно COCOMO
PCON	Преемственность персонала	Новый параметр
LTEX	Знание языка программирования и инструментария	Изменен с целью охвата знаний инструментария и языка
SITE	Распределенная разработка. Взаимодействие между командами разработчиков	Новые параметры, определяющие степень взаимной удаленности команд разработчиков и степень автоматизации их деятельности
TOOL	Использование программных инструментов	Без изменений относительно COCOMO
SCED	Требуемые сроки разработки	Без изменений относительно COCOMO

Правила вычисления показателя степени в модели COCOMO II

- Значение показателя степени изменяется от 1.1 до 1.24
- Значение показателя степени зависит от того, насколько новаторским является данный проект, от гибкости процесса разработки ПО, от применяемых процессов управления рисками, сплоченности команды программистов и уровня управления организацией-разработчиком
- Значение показателя степени рассчитывается с учетом пяти показателей по восьмибалльной шкале от низшего (7 баллов) до наивысшего (0 баллов) уровня
- Значения всех пяти показателей суммируются, сумма делится на 100, результат прибавляется к числу 1.01

Факторы, влияющие на показатель степени в модели COCOMO II

- Наличие прецедентов у приложения: уровень опыта организации-разработчика в данной области
- Гибкость процесса: степень строгости контракта, порядок его выполнения, присущая контракту свобода внесения изменений, виды деятельности в течение жизненного цикла и взаимодействие между заинтересованными сторонами
- Разрешение рисков, присущих архитектуре: степень технической осуществимости, продемонстрированной до перехода к полномасштабному производству
- Сплоченность команды: степень сотрудничества и того, насколько все заинтересованные стороны (покупатели, разработчики, пользователи, ответственные за сопровождение и др.) разделяют общую концепцию
- Зрелость процесса: уровень зрелости организации-разработчика, определяемая в соответствии с моделью CMM

Новизна проекта (PREC)

Отражает предыдущий опыт организации в реализации проектов данного типа. Очень низкий уровень этого показателя означает отсутствие опыта, наивысший уровень указывает на компетентность организации-разработчика в данной области ПО

Характеристика	Рекомендуемое значение
Полное отсутствие прецедентов, полностью непредсказуемый проект	6,2
Почти полное отсутствие прецедентов, в значительной мере непредсказуемый проект	4,96
Наличие некоторого количества прецедентов	3,72
Общее знакомство с проектом	2,48
Значительное знакомство с проектом	1,24
Полное знакомство с проектом	0

Гибкость процесса разработки (FLEX)

Отображает возможность изменения процесса разработки ПО. Очень низкий уровень этого показателя означает, что процесс определен заказчиком заранее, наивысший — заказчик определил лишь общие задачи без указания конкретной технологии процесса разработки ПО

Характеристика	Рекомендуемое значение
Точный, строгий процесс разработки	5,07
Случайные послабления в процессе	4,05
Некоторые послабления в процессе	3,04
Большей частью согласованный процесс	2,03
Некоторое согласование процесса	1,01
Заказчик определил только общие цели	0

Разрешение рисков в архитектуре системы (RESL)

Отображает степень детализации анализа рисков, основанного на анализе архитектуры системы. Очень низкий уровень данного показателя соответствует поверхностному анализу рисков, наивысший уровень означает, что был проведен тщательный и полный анализ всевозможных рисков

Характеристика	Рекомендуемое значение
Малое (20 %)	7
Некоторое (40 %)	5,65
Частое (60 %)	4,24
В целом (75 %)	2,83
Почти полное (90 %)	1,41
Полное (100%)	0

Сплоченность команды (TEAM)

Отображает степень сплоченности команды и их способность работать совместно. Очень низкий уровень этого показателя означает, что взаимоотношения в команде сложные, а наивысший — что команда сплоченная и эффективная в работе, не имеет проблем во взаимоотношениях

Характеристика	Рекомендуемое значение
Сильно затрудненное взаимодействие	5,48
Несколько затрудненное взаимодействие	4,38
Некоторая согласованность	3,29
Повышенная согласованность	2,19
Высокая согласованность	1,1
Взаимодействие как в едином целом	0

Уровень зрелости процесса разработки (PMAT)

Отображает уровень развития процесса создания ПО в организации-разработчике	
Характеристика	**Рекомендуемое значение**
Уровень 1 CMM	7
Уровень 1+ CMM	6,24
Уровень 2 CMM	4,68
Уровень 3 CMM	1,12
Уровень 7 CMM	1,56
Уровень 5 CMM	0

Пример использования модели COCOMO II

Предположим, что организация-разработчик выполняет программный проект в той области, в которой у нее мало опыта разработок. Заказчик ПО не определил строгий технологический процесс, который будет использоваться при создании программного продукта, но некоторые согласования были проведены. Заказчик не выделил в плане работ времени на анализ возможных рисков в архитектуре системы. Для создания программной системы необходимо сформировать новую команду специалистов. Организация-разработчик недавно привела в действие программу усовершенствования технологического процесса разработки ПО и может котироваться как организация второго уровня в соответствии с моделью оценки уровня зрелости.

Предварительный размер проекта оценен в 128 KSLOC.

Предполагается очень высокая надежность системы и сложность системных модулей, высокие ограничения объема памяти, низкий уровень использования программных инструментов и ускоренный график работы. Остальные драйверы затрат номинальные

Показатели проекта

Показатель проекта	Характеристика показателя	Значение
Новизна проекта	Это новый проект для организации, но предметная область является знакомой; данный показатель имеет низкий уровень	3,72
Гибкость процесса разработки	Некоторое согласование процесса разработки с заказчиком дает высокое значение показателю	1,01
Анализ архитектуры системы и рисков	Анализ не был проведен — уровень данного показателя очень низкий	7
Сплоченность команды	Команда разработчиков новая, информация о ней отсутствует - уровень этого показателя оценивается как обычный	3,29
Уровень развития процесса разработки	Организация имеет второй уровень зрелости процессов разработки	4,68
	Итого	19,7

Значение показателя степени p = 19,7/100 + 1,01 = 1,207

Выполнение расчетов

Трудозатраты=2,45*EApp*(Размер)^{p}

Eapp = RELY * CPLX * STOR * TOOL * SCED = 1,25 * 1,3 * 1, 21 * 1,12 * 1,29 = 2,841

Трудозатраты = 2,45 * 2,841 *128 ^{1,207} = 2432 чел.-мес.

Время = 3 * (Трудозатраты) $^{(0.33 + 0.2 * (p-1.01))}$ =

= 3 * 2432 $^{(0,33 + 0,2* (1,207 – 1,01))}$ = 53,44 мес.

Достоинства модели COCOMO II

- возможен учет достаточно полной номенклатуры факторов, влияющих на экономические характеристики производства сложных программных продуктов
- метод является достаточно универсальным и может поддерживать различные размеры, режимы и уровни качества продуктов
- фактические данные подбираются в соответствии с реальными проектами и факторами корректировки, которые могут соответствовать конкретному проекту и организации
- прогнозы производственных процессов являются варьируемыми и повторяемыми
- метод позволяет добавлять уникальные факторы для корректировки экономических характеристик, связанные со специфическим проектом и организацией
- возможна высокая степень достоверности калибровки с опорой на предыдущий опыт коллектива специалистов
- результаты прогнозирования сопровождаются обязательной документацией
- модель относительно проста в освоении и применении

Недостатки модели COCOMO II

- все результаты зависят от размера программного продукта: точность оценки размера, оказывает определяющее влияние на точность прогноза трудозатрат, длительности разработки и численности специалистов
- игнорируются требования к характеристикам качества программного продукта
- не учитывается зависимость между интегральными затратами и количеством времени, затрачиваемым на каждом этапе проекта
- игнорируется изменяемость требований к программному продукту в процессе производства
- не достаточно учитывается внешняя среда производства и применения программного продукта
- игнорируются многие особенности, связанные с аппаратным обеспечением проекта

Модель COCOMO II изначально представляет трудозатраты по сумме всех этапов производства (от этапа планирования концепции до этапа поставки программного продукта). При этом не учитываются проблемы сопровождения, переноса и повторного использования компонентов, которые трудно описывать в рамках одной и той же модели

Последовательное уточнение технико-экономических параметров программного проекта

1. Экспертные оценки экономических характеристик по прототипам – достоверность оценивания размера продукта 40 – 50%

2. Этап предварительного проектирования

- оценивание экономических характеристик в базовой модели COCOMO – достоверность оценивания размера продукта 20 – 30%
- оценивание экономических характеристик в детализированной модели COCOMO – достоверность оценивания размера продукта 10 – 20%

3. Этап детального проектирования

- оценивание экономических характеристик в упрощенной предварительной модели COCOMO достоверность оценивания размера продукта 5 – 10% (учитывается 7 параметров)
- оценивание экономических характеристик в детальной модели COCOMO II – достоверность оценивания размера продукта 3 – 5% (учитывается 17 параметров)

Спасибо за внимание!