Экономика программной инженерии

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

baryshnikovam@mail.ru

Лекция 6

Экономическая модель разработки ПО. Оценка технико-экономических показателей проекта. Оценка размера программного продукта в строках программного кода. Модель COCOMO

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

Прогнозирование технико-экономических параметров проекта — это анализ и расчёт экономической целесообразности его осуществления, основанный на сравнительной оценке затрат и результатов, эффективности использования, срока окупаемости вложений…

Задача прогнозирования технико-экономических параметров проекта разработки ПО (трудоемкости, стоимости, длительности работ) является достаточно сложной, так как решается в условиях неопределенности. Основная проблема заключается в широком спектре количественных и качественных показателей, которые с различных сторон характеризуют проект разработки ПО, и невысокая достоверность оценки их значений экспертами

При этом наличие корректных прогнозных оценок на этапе приятия решения о целесообразности реализации проекта в условиях ограничений сроков и ресурсов, предлагаемых заказчиком, важно с точки зрения отображения их в условиях будущего договора. Следствием недостатков или отсутствия технико-экономического обоснования является неверная оценка преимуществ новой программной разработки, недооценка роли других конкурирующих предложений, неизбежный перерасход средств и снижение качества программного продукта

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 своевременно идентифицировать эти изменения и выполнять реалистичное обновление оценок затрат

Факторы, повышающие точность оценок

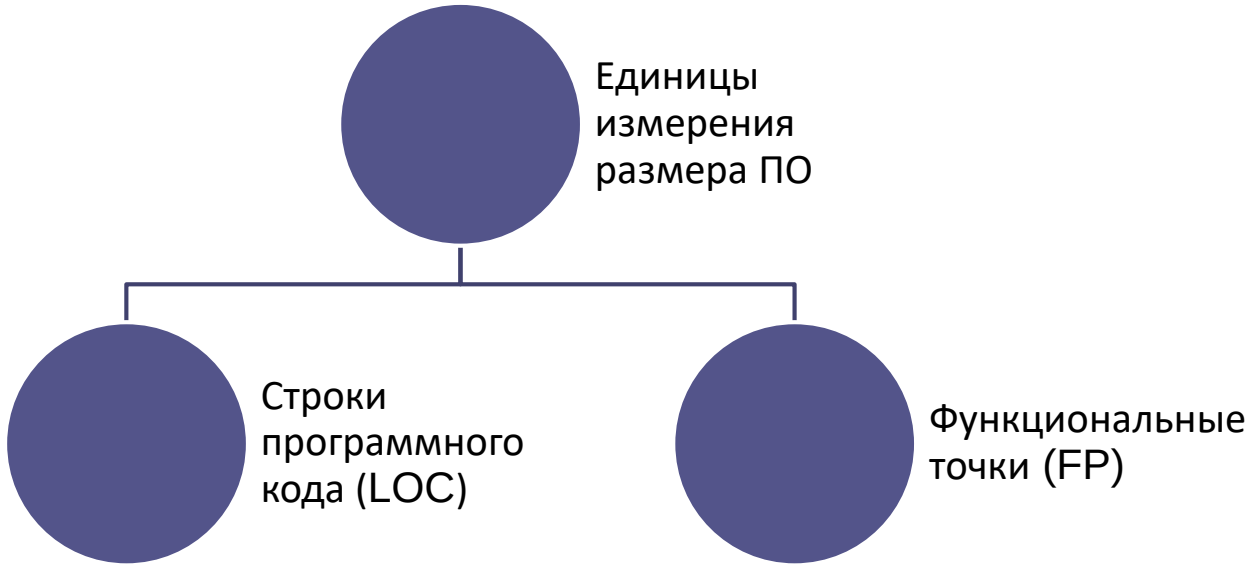
- цели оценивания технико-экономических показателей должны быть согласованы с потребностями в информации, способствующей принятию решений на соответствующем этапе программного проекта

- 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 решений (прежде всего – касающихся бюджета), принимаемых на ранних этапах и влияющих на следующие этапы

Основные параметры оценки при создании ПО

- 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 соглашение о методе подсчета строк кода. Языковые конструкции, используемые, например, в Visual C++, Ассемблере 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 программному продукту 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 А, размер которого составляет 2345 LOC. Мы хотим создать новый модуль А’, который будет во многом схож с модулем А, но в него будут добавлены некоторые дополнительные свойства. Кроме того, мы знаем как сделать программный код более компактным. В результате этого размер модуля А’ 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 трудоемкость (трудозатраты)
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, при этом в процессе разработки необходимы определенные инновации	Среда характеризуется незначительной нестабильностью
Встроенный	Более 500 KLOC	Большая команда разработчиков трудится над крупным проектом, в котором 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)

Примечание: Множители и показатели степени приведены для модели COCOMO 81: Intermediate Model, которая известна как промежуточная модель

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

Несколько слов о сжатии расписания и драйвере SCED

- Сжатие расписания — это метод управления проектом, используемый для сокращения продолжительности проекта без ущерба для его объема или качества. Потребность в нем возникает в связи с обновленной датой поставки, новой возможностью или задержкой графика.
- Одним из способов ускорить время выполнения проекта является выделение на него большего количества ресурсов. Однако следует помнить, что любые дополнительные затраты, связанные с ускорением проекта, сопоставляются с любыми потенциальными выгодами от его более раннего завершения.
- Еще один способ сжать расписание – это запараллелить выполнение действий или этапов, которые изначально были запланированы последовательно. Действия могут перекрываться, начинаться раньше, чем предлагалось, а также могут объединяться в расписании. Но этот процесс добавляет риск к графику и должен выполняться осторожно
- Наконец еще один способ сжать график – это перераспределить ресурсы с работ, которые не находятся на критическом пути, на те, которые находятся на критическом пути. Но это работает только в том случае, если такая возможность есть

- Драйвер SCED (ограничение сроков разработки) определяет значимость даты поставки продукта. Высокая степень значимости подразумевает, что продукт желательно или необходимо поставить как можно раньше
- Значение драйвера показывает сжатие расписания: очень низкий драйвер соответствует 75% от номинальной длительности; очень высокий – 160% от номинальной длительности

Таким образом, планируя сжатие расписания, следует учитывать, что это вызовет необходимость добавления дополнительных ресурсов в проект, увеличение сверхурочной работы и доплату за более быструю доставку важных компонентов. Было эмпирически доказано, что увеличение степени сжатия увеличивает стоимость проекта в геометрической прогрессии. По этой причине было включено максимальное сжатие в 25%. При превышении этих показателей сжатия проект считается однозначно «провальным»

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

Распределение бюджета по видам 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 государственной 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 = $2,8*EAF*(Размер)^{1,2}=2,8*1,28*(100)^{1,2}$ = 900 человеко-месяцев на разработку+72 человеко-месяца на планирование, определение требований=972 человеко-месяца
- Время = $2,5*(Трудозатраты)^{0,32} = 2,5*(900)^{0,32}$ = 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 и времени по этапам жизненного цикла

Вид деятельности	Трудозатраты (чел-мес.)	Время (мес.)
Планирование и определение требований	72	8
Проектирование продукта	162	8
Детальное проектирование	225	4
Кодирование и тестирование отдельных модулей	234	4
Интеграция и тестирование	279	6
ИТОГО:	***972***	***30***

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

Спасибо за внимание!