Экономика программной инженерии

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

baryshnikovam@mail.ru

Лекция 7

Влияние эффектов повторного использования кода на размер ПО. Модель COCOMO II

https://csse.usc.edu/ - Центр системной инженерии и программного обеспечения имени Боэма

Концепция повторного использования

- Многие программы происходят от предыдущих версий этих же программ
- В результате может быть достигнута экономия средств и/или времени, а также повышен уровень качества
- Повторному использованию подлежат: программный код, тестовый код, тестовые процедуры, документация, дизайн, спецификации требований и др.
- Программное обеспечение, повторно используемое в полном объеме, помимо собственно программного кода 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

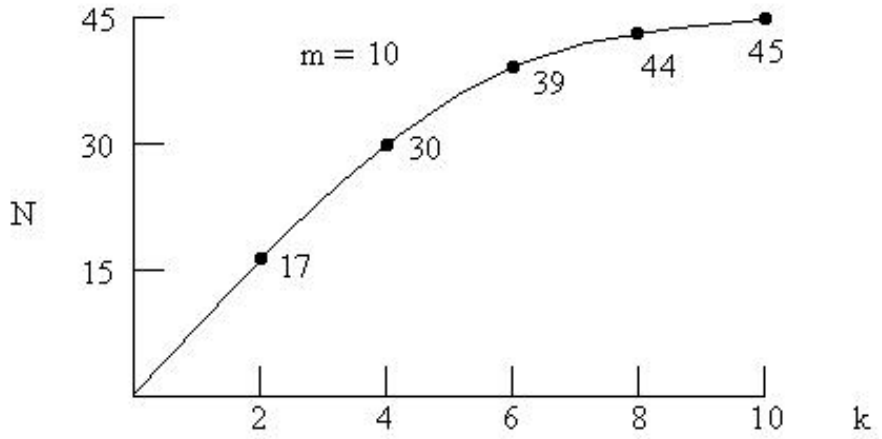
Элемент	LOC для нового кода	LOC для модифицируемого кода	LOC для повторно используемого кода	Итого LOC
Итого	2138	1782	1201	5121
Множитель	100%	60%	30%	
Результат	2138	1069	360	3567

Нелинейные эффекты повторного использования

Проведенный в 80-90-х годах анализ затрат на повторное использование почти 3000 программных модулей показывает, что функция приведенных затрат нелинейная в двух аспектах:

• Как правило, затраты на оценку, выбор и интеграцию компонента многократного использования составляют около 5%.

• Небольшие модификации приводят к непропорционально большим затратам. В первую очередь это связано с двумя факторами: стоимостью понимания программного обеспечения, подлежащего модификации, и относительной стоимостью проверки интерфейса



если модифицировать k из m программных модулей, количество N необходимых проверок интерфейса модуля равно N = k * (m-k) + k * (k-1)/2

Как только происходит переход от немодифицированного повторного использования («черный ящик») к модифицированному программному обеспечению («белый ящик»), человек сталкивается с недостатком понимания программного обеспечения. Проведенные исследования показывают, что 47% усилий по сопровождению программного обеспечения связано с пониманием программного кода, подлежащего модификации

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

Модель композиции приложения следует рассматривать как некую рабочую гипотезу, основанную на концепции продукта

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

Множитель трудоемкости	Идентификатор	Составные драйверы затрат
Надежность и сложность продукта	RCPX	RELY-DATA-CPLX-DOCU
Повторное использование компонентов	RUSE	RUSE
Сложность платформы	PDIF	TIME-STOR-PVOL
Опытность персонала	PERS	AEXP-PEXP-LTEX
Способности персонала	PREX	ACAP-PCAP-PCON
Возможности среды	FCIL	TOOL-SITE
Сроки	SCED	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						
	Сверхнизкий	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
PERS	2,12	1,62	1,26	1,00	0,83	0,63	0,5
RCPX	0,49	0,60	0,83	1,00	1,33	1,91	2,72
RUSE	n/a	n/a	0,95	1,00	1,07	1,15	1,24
PDIF	n/a	n/a	0,87	1,00	1,29	1,81	2,61
PREX	1,59	1,33	1,22	1,00	0,87	0,73	0,62
FCIL	1,43	1,30	1,10	1,00	0,87	0,73	0,62
SCED	n/a	1,43	1,14	1,00	1,00	1,00	n/a

Комментарии к множителям трудоемкости

- **PERS** — квалификация персонала (Сверхнизкий — аналитики и программисты имеют низшую квалификацию, текучесть больше 45%; Сверхвысокий — аналитики и программисты имеют высшую квалификацию, текучесть меньше 4%)
- **RCPX** — сложность и надежность продукта (Сверхнизкий — продукт простой, специальных требований по надежности нет, БД маленькая, документация не требуется; Сверхвысокий — продукт очень сложный, требования по надежности жесткие, БД сверхбольшая, документация требуется в полном объеме)
- **RUSE** — разработка для повторного использования (Низкий — не требуется; Сверхвысокий — требуется переиспользование в других продуктах)
- **PDIF** — сложность платформы разработки (Сверхнизкий — специальные ограничения по памяти и быстродействию отсутствуют, платформа стабильна; Сверхвысокий — жесткие ограничения по памяти и быстродействию, платформа нестабильна)
- **PREX** — опыт персонала (Сверхнизкий — новое приложение, инструменты и платформа; Сверхвысокий — приложение, инструменты и платформа хорошо известны)
- **FCIL** — оборудование (Сверхнизкий — инструменты простейшие, коммуникации затруднены; Сверхвысокий — интегрированные средства поддержки жизненного цикла, интерактивные мультимедиа коммуникации)
- **SCED** — сжатие расписания (Очень низкий — 75% от номинальной длительности; Очень высокий — 160% от номинальной длительности)

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

Драйверы затрат в модели COCOMO II

Идентификатор	Уточняющий фактор работ	Диапазон изменения параметра	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Чрезвычайно высокий
RELY	Требуемая надежность	0.81 – 1.26	0.81	0.91	1.0	1.1	1.26	
DATA	Размер базы данных	0.9 – 1.28		0.9	1.0	1.14	1.28	
CPLX	Сложность продукта	0.73 – 1.74	0.73	0.87	1.0	1.17	1.34	1.74
RUSE	Требуемый уровень повторного использования	0.98 – 1.34		0.98	1.0	1.07	1.15	1.34
DOCU	Соответствие документации требованиям жизненного цикла	0.81 – 1.23	0.81	0.91	1.0	1.11	1.23	
TIME	Ограничение времени выполнения	1.0 – 1.63			1.0	1.11	1.29	1.63
STOR	Ограничение объема основной памяти	1.0 – 1.46			1.0	1.05	1.17	1.46
PVOL	Изменчивость платформы	0.87 – 1.3		0.87	1.0	1.15	1.3	
ACAP	Способности аналитика	0.71 – 1.42	1.42	1.19	1.0	0.85	0.71	
AEXP	Знание приложений	0.81 – 1.22	1.22	1.1	1.0	0.88	0.81	
PCON	Преемственность персонала	0.81 – 1.29	1.29	1.12	1.0	0.9	0.81	
PCAP	Способности программиста	0.76 – 1.34	1.34	1.15	1.0	0.88	0.76	
PEXP	Опыт работы с платформой	0.85 – 1.19	1.19	1.09	1.0	0.91	0.85	
LTEX	Знание языка программирования и инструментария	0.84 – 1.2	1.2	1.09	1.0	0.91	0.84	
TOOL	Использование программных инструментов	0.78 – 1.17	1.17	1.09	1.0	0.9	0.78	
SITE	Распределенная разработка	0.8 – 1.22	1.22	1.09	1.0	0.93	0.86	0.8
SCED	Требуемые сроки разработки	1.43 – 1	1.43	1.14	1.0	1.0	1.0	

Краткое пояснение к драйверам затрат

	Драйвер	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Чрезвычайно высокий
RELY	*Требуемая надежность*	Небольшие ограничения	Низкие, легко возмещаемые потери	Умеренные, легко возмещаемые потери	Большие финансовые потери	Риск для человеческой жизни	
DATA	*Размер базы данных*		Размер БД (байт) / размер программы (SLOC) < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 100	
CPLX	*Сложность продукта*	См. слайды №24-26					
RUSE	*Требуемый уровень повторного использования*		нет	По всему проекту	По всей программе	По всей продуктовой линейке	В нескольких продуктовых линейках

Оценки сложности (CPLX) в зависимости от типа модуля

	Операции по управлению	Вычислительные операции	Аппаратно-зависимые операции	Операции по управлению данными	Операции по управлению пользовательским интерфейсом
Очень низкий	Линейный код с несколькими не вложенными операторами структурированного программирования. Простая компоновка модуля с помощью вызовов подпрограмм или простых скриптов	Вычисление простых выражений: например, A=B+C*(DE)	Простое чтение, запись инструкций в простых форматах	Простые массивы в основной памяти. Простые запросы и обновления БД	Простые формы ввода, генератор отчетов
Низкий	Простое вложение операторов структурного программирования. В основном простые предикаты	Вычисление выражений среднего уровня: например, $d = \sqrt{b^2 - 4 * a * c}$	Не требуется знаний характеристик конкретного процессора или устройства ввода-вывода. Ввод-вывод выполняется на уровне GET/PUT	Единичный файл без изменений структуры данных, без правок, без промежуточных файлов. Умеренно сложные запросы и обновления БД	Использование простого графического пользовательского интерфейса (GUI)

Оценки сложности в зависимости от типа модуля

	Операции по управлению	Вычислительные операции	Аппаратно-зависимые операции	Операции по управлению данными	Операции по управлению пользовательским интерфейсом
Номинальный	В основном простая вложенность. Некоторый межмодульный контроль. Простые вызовы или передача сообщений, включая распределенную обработку	Использование стандартных математических и статистических функций. Основные матрично-векторные операции	Обработка ввода-вывода включает в себя выбор устройства, проверку состояния и обработку ошибок	Многофайловый ввод и однофайловый вывод. Простые структурные изменения, простые правки. Сложные запросы к базе данных и обновления	Использование простого набора виджетов
Высокий	Сильно вложенные операторы структурного программирования со множеством составных предикатов. Однородная, распределенная обработка. Однопроцессорное программное управление в режиме реального времени	Базовый численный анализ: многомерная интерполяция, обыкновенные дифференциальные уравнения	Операции на физическом уровне ввода -вывода (преобразование адресов физического хранилища; поиск, считывание). Оптимизированное перекрытие ввода-вывода	Простые триггеры, активируемые содержимым потока данных. Сложная реструктуризация данных	Разработка и расширение набора виджетов. Простой голосовой ввод-вывод, мультимедиа

Оценки сложности в зависимости от типа модуля

	Операции по управлению	Вычислительные операции	Аппаратно-зависимые операции	Операции по управлению данными	Операции по управлению пользовательским интерфейсом
Очень высокий	Реентерабельное и рекурсивное кодирование. Обработка прерываний с фиксированным приоритетом. Синхронизация задач, сложные обратные вызовы, гетерогенная распределенная обработка. Однопроцессорное жесткое управление в режиме реального времени.	Сложный, но структурированный численный анализ: почти сингулярные матричные уравнения, уравнения в частных производных. Простое распараллеливание	Процедуры для диагностики прерываний, обслуживания, маскировки. Встраиваемые системы с высокой производительностью	Координация распределенной базы данных. Сложные триггеры. Оптимизация поиска	Умеренно сложная 2D / 3D динамическая графика, мультимедиа
Чрезвычайно высокий	Планирование нескольких ресурсов с динамически меняющимися приоритетами. Управление на уровне микрокода. Распределенный жесткий контроль в режиме реального времени	Сложный и неструктурированный численный анализ: высокоточный анализ зашумленных, стохастических данных. Сложное распараллеливание	Кодирование, зависящее от времени работы устройства, микропрограммируемые операции. Встраиваемые системы, критически важные для производительности	Высоко-связные, динамичные реляционные и объектные структуры. Управление данными на естественном языке	Сложные мультимедиа, виртуальная реальность

Краткое пояснение к драйверам затрат

	Драйвер	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Чрезвычайно высокий
DOCU	*Соответствие документации требованиям жизненного цикла*	Многие аспекты жизненного цикла не покрыты документацией	Некоторые аспекты жизненного цикла не покрыты документацией	Объем документации соответствует потребностям жизненного цикла	Объем документации избыточен для потребностей жизненного цикла	Очень избыточный объем документации для нужд жизненного цикла	
TIME	*Ограничение времени выполнения*			50% использования доступного времени выполнения exe-файла	70%	85%	95%
STOR	*Ограничение объема основной памяти*			50% использования доступного хранилища	70%	85%	95%
PVOL	*Изменчивость платформы*		Существенные изменения каждые 12 месяцев; незначительные изменения каждые 1 месяц	Существенные изменения каждые 6 месяцев; незначительные изменения каждые 2 недели	Существенные изменения каждые 2 месяца; незначительные изменения каждую неделю	Существенные изменения каждые 2 недели; незначительные изменения каждые 2 дня	

Краткое пояснение к драйверам затрат

	Драйвер	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий
ACAP	*Способности аналитика*	15 проц.	35 проц.	55 проц.	75 проц.	90 проц.
AEXP	*Знание приложений*	≤ 2 мес.	6 мес.	1 год	3 года	6 лет
PCON	*Преемственность персонала*	48% / год	24% / год	12% /год	6% / год	3% / год
PCAP	*Способности программиста*	15 проц.	35 проц.	55 проц.	75 проц.	90 проц.
PEXP	*Опыт работы с платформой*	≤ 2 мес.	6 мес.	1 год	3 года	6 лет
LTEX	*Знание языка программирования и инструментария*	≤ 2 мес.	6 мес.	1 год	3 года	6 лет

Примечания:

1) Процентиль – это значение, которое заданная случайная величина не превышает с фиксированной вероятностью, заданной в процентах. *n*-й процентиль набора данных - это значение, при котором *n* процентов данных ниже его

2) Драйвер PCON оценивается на основе годовой текучести кадров

3) Оценка драйвера PCAP основывается на способностях программистов как команды, а не отдельных людей

Краткое пояснение к драйверам затрат

	Драйвер	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий
TOOL	*Использование программных инструментов*	Редактирование, кодирование, отладка	Простой, внешний интерфейс, небольшая интеграция	Базовые инструменты поддержки жизненного цикла, интегрированные сравнительно недавно	Умеренно интегрированные зрелые инструменты поддержки жизненного цикла	Зрелые инструменты проактивного жизненного цикла, хорошо интегрированные с методами повторного использования
SITE	*Распределенная разработка*	См. следующий слайд				
SCED	*Требуемые сроки разработки*	75% от номинального	85% от номинального	100% от номинального	130% от номинального	160% от номинального

Учет влияния распределенной разработки и коммуникационной поддержки (драйвер SITE)

	Драйвер	**Очень низкий**	**Низкий**	**Номинальный**	**Высокий**	**Очень высокий**	**Чрезвычайно высокий**
SITE	*Распределенная разработка (колокация)*	Международный уровень	В нескольких городах и в разных компаниях	В нескольких городах или в разных компаниях	Один город или область	Одно здание или комплекс	Полностью совмещенная
SITE	*Распределенная разработка (коммуникация)*	Телефон, почта	Телефон, факс	Электронная почта	Широкополосная электронная почта	Видеоконференции	Интерактивные мультимедиа

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 восьми балльной 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	7,8
Уровень 1+ CMM	6,24
Уровень 2 CMM	4,68
Уровень 3 CMM	3,12
Уровень 4 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 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 примерно котироваться как организация второго уровня в соответствии с моделью оценки уровня зрелости.

Предварительный размер проекта оценен в **28 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 = 1,26 * 1,34 * 1, 05 * 1,09 * 0,75 = 1,45

Трудозатраты = 2,45 * 1,45 *28 ^{1,207} = **198** чел.-мес.

Время = 3 * (Трудозатраты) $^{(0.33 + 0.2 * (p-1.01))}$ = 3 * 198 $^{(0,33 + 0,2* (1,207 – 1,01))}$ = **21,2** 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
- недостаточно 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!