Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Оренбургский государственный университет»

Кафедра программного обеспечения вычислительной
техники и автоматизированных систем

РАСЧЕТ ТРУДОЗАТРАТ ПРИ УПРАВЛЕНИИ ПРОГРАММНЫМИ ПРОЕКТАМИ

Методические указания
Составитель
Т.М. Зубкова

Рекомендовано к изданию редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по образовательным программам высшего образования по направлениям подготовки 09.04.04 Программная инженерия, 09.04.01 Информатика и вычислительная техника

Оренбург
2021

УДК 681.3 (07)
ББК 32.973.26-018я73
Р 24

Рецензент – доктор технических наук А.С. Боровский

Р 24 **Расчет трудозатрат при управлении программными проектами:** методические указания / составитель Т.М. Зубкова; Оренбургский гос. ун-т.- Оренбург: ОГУ, 2021. – 53 с.

В методических указаниях изложены задания, теоретические основы для их выполнения, а также примеры выполнения расчетов.

Методические указания предназначены для выполнения практических заданий по дисциплине «Теория и практика управления проектами» предназначены для оказания помощи студентам при выполнении индивидуальных заданий. Данная дисциплина входит в вариативную часть обязательных дисциплин магистрантов очной и заочной форм обучения по направлению подготовки 09.04.04 Программная инженерия с профилем подготовки «Разработка программно-информационных систем» и 09.04.01 Информатика и вычислительная техника с профилями подготовки «Информационное и программное обеспечение автоматизированных систем», «Системы автоматизации проектирования в машиностроении», «Распределенные информационно-вычислительные системы и телекоммуникации» по типу образовательной программы «Программа академической магистратуры».

УДК 681.3 (07)
ББК 32.973.26-018я73

Содержание

Введение

Цель освоения дисциплины: создание теоретической основы и практических навыков управления программными проектами при разработке программного обеспечения, применяемого в различных предметных областях.

Задачи:

- обучить применению различных методов управления программными проектами согласно этапам жизненного цикла;
- обучить методам оценки эффективности и затрат программного проекта;
- обучить методикам формирования команд и эффективного руководства коллективом при разработке программного обеспечения;
- обучить методам разработки технического задания, составления планов работы коллектива.

Цель практических занятий: закрепление теоретических знаний и формирование практических умений и навыков для управления программными проектами при разработке программных средств, применяемых в различных предметных областях.

1 Практическое занятие №1. Введение в управление программным проектом

На основании изучения предметной области, согласно варианту выполнить следующее:

1. Понять содержание проекта, область применения, временные рамки.

2. Определиться с процессом разработки (методы, инструменты, языки программирования, документация, поддержка).

3. Определить организационную структуру (сколько человек участвует, кто, что делает и за что отвечает).

4. Составить расписание работы с проектом.

Краткие теоретические основы для выполнения практической работы

Управление проектом заключается в управлении производством программного продукта в рамках отведенных средств и времени. Поскольку для этого требуются человеческие ресурсы, то для управления проектом необходимы не только технические и организационные навыки, по еще и искусство управления людьми.

Управление проектом охватывает:

- инфраструктуру (организационные моменты);
- управляющий процесс (права и ответственности участников);
- процесс разработки (методы, инструменты, языки, документация и поддержка);
- расписание (моменты времени, к которым должны быть представлены выполненные фрагменты работы).

Основные параметры: стоимость, функциональность, качество и расписание.

Планировщики проекта могут варьировать стоимость, возможности, качество и дату завершения проекта. Руководитель проекта может управлять следующими факторами.

1. Общая стоимость проекта.
2. Возможности продукта.
3. Качество продукта.
4. Длительность разработки проекта.

Степень контроля над этими четырьмя факторами зависит от проекта. Несмотря на то, что стоимость может быть оговоренной заранее и фиксированной, зачастую существуют различные гибкие варианты.

Типичная схема процесса управления проектом

Типичная последовательность действий, необходимая для запуска проекта.

1. Понять содержание проекта, область применения и временные рамки.
2. Определиться с процессом разработки (методы, инструменты, языки, документация и поддержка).
3. Выделить организационную структуру (привлечение отделов организации).
4. Определить управляющий процесс (ответственность участников).
5. Разработать расписание проекта (моменты сдачи частей работы).
6. Разработать план подбора кадров.
7. Начать управление рисками.
8. Определить, какие документы необходимо выработать.
9. Начать сам процесс.

К пониманию проекта (1) относится общее представление о целях и задачах, а не сбор всех требований. Последнее является длительным процессом, начинающимся после того, как набросан примерный вариант плана проекта.

Мы также должны понять общие возможности и назначение приложения. Банковское приложение может иметь как огромные возможности, например «Автоматизация всех банковских транзакций», так и совсем скромные, например «Сортировка по номеру чека», или нечто промежуточное. Необходимо прикинуть в общих чертах количество трудозатрат и времени на проект (например, два месяца или два года).

Далее (2) требуется принять решение об использовании той или иной модели процесса разработки (водопадной, спиральной, инкрементальной, комбинированной и т. д.).

Затем (3) следует провести общую организацию персонала (выяснить, какие отделы компании будут задействованы в проекте, установить количество команд, определиться с контролем качества). Как только все приготовления будут завершены, можно сосредоточиться на главной задаче для успешного выполнения работы – процессе управления (4).

Здесь необходимо будет определить, кто перед кем отчитывается, или задать уровневую структуру типа (командный процесс разработки программного обеспечения TSP).

Следующий шаг (5) – это создание общего расписания (определяющего, когда что должно быть сделано). После этого можно завершать план подряда, в котором приводится количество персонала, привлекаемого к проекту, и примерное количество человеко-дней для различных задач (например, анализа требований). Этот план не может быть конкретным (например, кто над какой функцией будет работать), так как до сих пор еще не определена архитектура приложения. Проекты практически всегда подвергаются влиянию всяких неприятностей, поэтому определение проекта и управление рисками в проекте (7) необходимо начинать одновременно. Остается подготовить соответствующий набор документов (например, документы SVVP и SRS стандарта IEEE) (8), а затем запустить непосредственно сам проект (9).

2 Практическое занятие №2. Инициация проекта

1. Оценить финансовую ценность проекта.
2. Оценить стратегическую ценность проекта.
3. Оценить уровень риска.
4. Описать концепцию проекта. Определить цель и результаты проекта.

Краткие теоретические основы для выполнения практической работы

Эффективные процессы инициации программного проекта не менее , чем на наполовину определяют его будущую успешность. Недостаточное внимание именно этой фазе проекта неизбежно приводит к существенным проблемам при планировании, реализации и завершении проекта.

Инициация состоит из процессов, способствующих формальной авторизации начала нового проекта или фазы проекта. Процессы инициации часто выполняются вне рамок проекта и связаны с организационными, программными или портфельными процессами. В ходе процесса инициации уточняются первоначальное описание содержания и ресурсы, которые организация планирует вложить. На этом этапе также выбирается менеджер проекта, если он еще не назначен, и документируются исходные допущения и ограничения. Эта информация заносится в Устав проекта и, если он одобряется, проект официально авторизуется.

Устав проекта – документ, выпущенный инициатором или спонсором проекта, который формально узаконивает существование проекта и предоставляет менеджеру проекта полномочия использовать организационные ресурсы в операциях проекта.

В российской практике данный документ чаще называется Концепция проекта.

Концепция (от лат. conceptio – понимание, система), определённый способ понимания, трактовки какого-либо предмета, явления, процесса, основная

точка зрения на предмет и др., руководящая идея для их систематического освещения.

В компании, которая принимает решение о начале того или иного проекта разработки ПО, должна существовать единая система критериев для оценки его значимости. Система критериев должна позволять из множества возможных для реализации проектов выбрать наиболее приоритетные для компании.

Приоритет любого проекта должен определяться на основе оценки трех его характеристик:

- Финансовая ценность.
- Стратегическая ценность.
- Уровень рисков.

Шкала оценки финансовой ценности проекта может выглядеть следующим образом:

- Высокая. Ожидаемая окупаемость до 1 года. Ожидаемые доходы от проекта не менее чем в 1.5 раз превышают расходы. Все допущения при проведении этих оценок четко обоснованны.
- Выше среднего. Ожидаемая окупаемость проекта от 1 года до 3 лет. Ожидаемые доходы от проекта не менее чем в 1.3 раза превышают расходы. Большинство допущений при проведении этих оценок имеют под собой определенные основания.
- Средняя. Проект позволяет улучшить эффективность производства в Компании и потенциально может снизить расходы компании не менее чем на 30%. Проект может иметь информационную ценность или помочь лучше контролировать бизнес.
- Низкая. Проект немного снижает расходы компании не менее чем на 10% и дает некоторые улучшения производительности производства.

Одной финансовой ценности для определения приоритета проекта недостаточно. Поэтому, важным показателем приоритета проекта является его соответствие стратегическим целям компании.

Шкала оценки стратегической ценности проекта может иметь следующий вид:

- Высокая. Обеспечивает стратегическое преимущество, дает устойчивое увеличение рынка или позволяет выйти на новый рынок. Решает значительные проблемы, общие для большинства важных клиентов. Повторение конкурентами затруднено или потребует от 1 до 2 лет.
- Выше среднего. Создает временные конкурентные преимущества. Выполнение обязательств перед многими важными клиентами. Конкурентное преимущество может быть удержано в течение 1 года.
- Средняя. Поддерживается доверие рынка к компании. Повышает мнение клиентов о качестве предоставляемых услуг или способствует выполнению обязательств перед несколькими клиентами. Конкуренты уже имеют или способны повторить новые возможности в пределах года.
- Низкая. Стратегическое воздействие отсутствует или незначительно. Влияние на клиентов несущественно. Конкуренты могут легко повторить результаты проекта.

Третьим обязательным показателем приоритета проекта должна быть оценка уровня его риска. ***Риск*** – неопределенное событие или условие, наступление которого отрицательно или положительно сказывается на целях проекта.

Ни один проект, который имеет даже самую высокую оценку финансовой выгодности, не будет запущен в производство, если достижение этой сверхвыгоды имеет минимальные шансы.

Примерная шкала оценки уровня рисков проекта может иметь следующий вид:

- Низкий. Цели проекта и требования хорошо поняты и документированы. Масштаб и рамки проекта заданы четко. Ресурсы требуемой квалификации доступны в полном объеме. Разрабатываемые системы не потребуют новой технологической платформы.
- Средний. Цели проекта определены более-менее четко. Хорошее понимание требований к системе. Масштаб и рамки проекта заданы достаточно

хорошо. Ресурсы требуемой квалификации доступны в основном. Системы создаются на новой, но стабильной технологической платформе.

- Выше среднего. Цели проекта недостаточно четки. Задачи системы или бизнес-приложения поняты недостаточно полно. Понимание масштаба и рамок проекта недостаточно. Ресурсы требуемой квалификации сильно ограничены. Системы создаются на новой технологической платформе, сомнения в рыночной стабильности платформы.
- Высокий. Цели проекта нечетки. Основные функциональные компоненты системы не определены. Масштаб и рамки проекта непонятны. Ресурсы требуемой квалификации практически отсутствуют. Системы создаются на новой технологической платформе, в отношении которой крайне мало ясности. Технологии имеют неподтвержденную стабильность.

Если компания уделяет мало внимания управлению приоритетами своих проектов, то это приводит к переизбытку реализуемых проектов, перегруженности исполнителей, постоянным авралам и сверхурочным работам и, как следствие, к низкой эффективности производственной деятельности. При старте нового проекта с высоким приоритетом, компания должна остановить или закрыть менее значимые проекты, чтобы обеспечить новый проект необходимыми ресурсами, а не пытаться сделать все и сразу за счет интенсификации работ, как правило, это не получается.

Концепция проекта

У каждого проекта должна быть концепция. Если проект небольшой, то для изложения концепции часто достаточно несколько абзацев. Однако, стартовать проект без концепции, это все равно, что отправлять корабль в плавание, не определив для него пункт назначения.

Концепция проекта разрабатывается на основе анализа потребностей бизнеса. Главная функция документа – подтверждение и согласование единого видения целей, задач и результатов всеми участниками проекта. Концепция определяет, что и зачем делается в проекте.

Концепция проекта это ключевой документ, который используется для принятия решений в ходе всего проекта, а также на фазе приемки – для подтверждения результата. Она содержит, как правило, следующие разделы:

- Название проекта
- Цели проекта
- Результаты проекта
- Допущения и ограничения
- Ключевые участники и заинтересованные стороны
- Ресурсы проекта
- Сроки
- Риски
- Критерии приемки
- Обоснование полезности проекта

Цели проекта должны отвечать на вопрос, зачем данный проект нужен. Цели проекта должны описывать бизнес-потребности и задачи, которые решаются в результате исполнения проекта. Целями проекта могут быть:

- Изменения в Компании. Например, автоматизация ряда бизнес-процессов для повышения эффективности основной производственной деятельности
- Реализация стратегических планов. Например, завоевание значительной доли растущего рынка за счет вывода на него нового продукта.
- Выполнение контрактов. Например, разработка программного обеспечения по заказу.
- Разрешение специфических проблем. Например, доработка программного продукта в целях приведения его в соответствие с изменениями в законодательстве.

Цели должны быть значимыми (направленными на достижение стратегических целей Компании), конкретными (специфичными для данного проекта), измеримыми (т.е иметь проверяемые количественные оценки), реальными (до-

стижимыми). Четкое определение бизнес-целей важно, поскольку существенно влияет на все процессы и решения в проекте. Проект должен быть закрыт, если признается, что достижение цели невозможно или стало нецелесообразным. Например, если реальные затраты на проект будут превосходить будущие доходы от его реализации.

Результаты проекта отвечают на вопрос, что должно быть получено после его завершения. Результаты проекта должны определять:

- Какую выгоду получит заказчик в результате проекта.
- Какой продукт или услуга будет произведен по окончании проекта.
- Высокоуровневые требования – краткое описание ключевых свойств и/или характеристик продукта/услуги.

Результаты проекта должны быть измеримыми. Это означает, что при оценке результатов проекта должна иметься возможность сделать заключение достигнуты оговоренные в концепции результаты или нет.

Допущения тесно связаны с управлением рисками. В разработке ПО часто приходится формулировать риски в виде допущений, тем самым передавая его заказчику. Например, оценивая проект разработки и внедрения по схеме с фиксированной ценой, мы должны записать в допущения предположение о том, что стоимость лицензий на стороннее ПО не изменится, до завершения проекта.

Ограничения, как правило, сокращают возможности проектной команды в выборе решений. В частности они могут содержать:

- Специфические нормативные требования. Например, обязательная сертификация продукта, услуги на соответствие определенным стандартам.
- Специфические технические требования. Например, разработка под заданную программно-аппаратную платформу.
- Специфические требования к защите информации.

Также уместно сформулировать те требования к системе, которые могут ожидаться заказчиком по умолчанию, но не включаются в рамки данного проекта.

К **ключевым участникам** программного проекта, как правило, относятся:

- Спонсор проекта – лицо или группа лиц, предоставляющая финансовые ресурсы для проекта в любом виде.
- Заказчик проекта – лицо или организация, которые будут использовать продукт, услугу или результат проекта. Следует учитывать, что заказчик и спонсор проекта не всегда совпадают.
- Пользователи результатов проекта.
- Куратор проекта – представитель исполнителя, уполномоченный принимать решение о выделении ресурсов и изменениях в проекте.
- Руководитель проекта – представитель исполнителя, ответственный за реализацию проекта в срок, в пределах бюджета и с заданным качеством.
- Соисполнители проекта. Субподрядчики и поставщики.

Ресурсы. Для того чтобы понять, сколько будет стоить реализация программного проекта, требуется определить и оценить ресурсы необходимые для его выполнения:

- Людские ресурсы и требования к квалификации персонала.
- Оборудование, услуги, расходные материалы, лицензии на ПО, критические компьютерные ресурсы.
- Бюджет проекта. План расходов и, при необходимости, предполагаемых доходов проекта с разбивкой по статьям и фазам/этапам проекта.

Специфика программного проекта заключается в том, что людские ресурсы вносят основной вклад в его стоимость. Все остальные затраты, как правило, незначительны, по сравнению с этим расходами.

3 Практическое занятие №3. Проведение расчетов по размерно-ориентированным и функционально-ориентированным метрикам

1. На основании данных предыдущих разработок посчитать размерно-ориентированные метрики: производительность, качество, удельную стоимость, документированность.

2. Сделать расчет по функционально-ориентированным метрикам. Рассчитать: производительность, качество, удельную стоимость, документированность.

Краткие теоретические основы для выполнения практической работы

Размерно-ориентированные метрики

Размерно-ориентированные метрики прямо измеряют программный продукт и процесс его разработки. Основываются размерно-ориентированные метрики на LOC-оценках (Lines Of Code). LOC-оценка – это количество строк в программном продукте.

Исходные данные для расчета этих метрик сводятся в таблицу (таблица 1).

Таблица 1 – Исходные данные для расчета LOC-метрик

Проект	Затраты, чел.-мес	Стоимость, тыс. $	KLOC, тыс. LOC	Прогр. докты, страниц	Ошибки	Люди
A1	24	158	12,1	355	27	2
B2	62	440	27,2	1224	76	6
C3	43	314	20,2	1050	54	5

Таблица содержит данные о проектах за последние несколько лет. Например, запись о проекте A1 показывает: 12 100 строк программы были разработаны за 24 человеко-месяца и стоили $158 000. Кроме того, по проекту A1 было разработано 355 страниц документации, а в течение первого года эксплуа-

тации было зарегистрировано 27 ошибок. Разрабатывали проект А1 два человека.

На основе таблицы вычисляются размерно-ориентированные метрики.

$$\text{Производительность} = \frac{\text{Длина}}{\text{Затраты}} \quad \frac{\text{тыс.LOC}}{\text{чел.-мес}} \tag{1}$$

$$\text{Качество} = \frac{\text{Ошибки}}{\text{Длина}} \quad \frac{\text{Единиц}}{\text{тыс.LOC}} \tag{2}$$

$$\text{Удельная стоимость} = \frac{\text{Стоимость}}{\text{Длина}} \quad \frac{\text{Тыс\$}}{\text{LOC}} \tag{3}$$

$$\text{Документирванность} = \frac{\text{Страниц Документа}}{\text{Длина}} \left[\frac{\text{Страниц}}{\text{тыс.LOC}} \right] \tag{4}$$

Функционально-ориентированные метрики

Функционально-ориентированные метрики косвенно измеряют программный продукт и процесс его разработки. Вместо подсчета LOC-оценки при этом рассматривается не размер, а функциональность или полезность продукта.

Используется 5 информационных характеристик.

1. *Количество внешних вводов.* Подсчитываются все вводы пользователя, по которым поступают разные прикладные данные. Вводы должны быть отделены от запросов, которые подсчитываются отдельно.

2. *Количество внешних выводов.* Подсчитываются все выводы, по которым к пользователю поступают результаты, вычисленные программным приложением. В этом контексте выводы означают отчеты, экраны, распечатки, сообщения об ошибках. Индивидуальные единицы данных внутри отчета отдельно не подсчитываются.

3. *Количество внешних запросов.* Под запросом понимается диалоговый ввод, который приводит к немедленному программному ответу в форме диалогового вывода. При этом диалоговый ввод в приложении не сохраняется, а диалоговый вывод не требует выполнения вычислений. Подсчитываются все запросы – каждый учитывается отдельно.

4. *Количество внутренних логических файлов.* Подсчитываются все логические файлы (то есть логические группы данных, которые могут быть частью базы данных или отдельным файлом).

5. *Количество внешних интерфейсных файлов.* Подсчитываются все логические файлы из других приложений, на которые ссылается данное приложение.

Вводы, выводы и запросы относят к категории *транзакция.*

Транзакция – это элементарный процесс, различаемый пользователем и перемещающий данные между внешней средой и программным приложением. В своей работе транзакции используют внутренние и внешние файлы. Приняты следующие определения.

Внешний ввод – элементарный процесс, перемещающий данные из внешней среды в приложение. Данные могут поступать с экрана ввода или из другого приложения. Данные могут использоваться для обновления внутренних логических файлов. Данные могут содержать как управляющую, так и деловую информацию. Управляющие данные не должны модифицировать внутренний логический файл.

Внешний вывод – элементарный процесс, перемещающий данные, вычисленные в приложении, во внешнюю среду. Кроме того, в этом процессе могут обновляться внутренние логические файлы. Данные создают отчеты или выходные файлы, посылаемые другим приложениям. Отчеты и файлы создаются на основе внутренних логических файлов и внешних интерфейсных файлов. Дополнительно этот процесс может использовать вводимые данные, их образуют критерии поиска и параметры, не поддерживаемые внутренними логическими файлами. Вводимые данные поступают извне, но носят временный характер и не сохраняются во внутреннем логическом файле.

Внешний запрос – элементарный процесс, работающий как с вводимыми, так и с выводимыми данными. Его результат – данные, возвращаемые из внутренних логических файлов и внешних интерфейсных файлов. Входная часть процесса не модифицирует внутренние логические файлы, а выходная часть не

несет данных, вычисляемых приложением (в этом и состоит отличие запроса от вывода).

Внутренний логический файл – распознаваемая пользователем группа логически связанных данных, которая размещена внутри приложения и обслуживается через внешние вводы.

Внешний интерфейсный файл – распознаваемая пользователем группа логически связанных данных, которая размещена внутри другого приложения и поддерживается им. Внешний файл данного приложения является внутренним логическим файлом в другом приложении.

Каждой из выявленных характеристик ставится в соответствие сложность. Для этого характеристике назначается низкий, средний или высокий ранг, а затем формируется числовая оценка ранга.

Для транзакций ранжирование основано на количестве ссылок на файлы и количестве типов элементов данных. Для файлов ранжирование основано на количестве типов элементов-записей и типов элементов данных, входящих в файл.

Тип элемента-записи – подгруппа элементов данных, распознаваемая пользователем в пределах файла.

Для определения числовых значений функциональности ПС необходимо использовать данные таблиц 2 - 9.

Таблица 2 – Примеры элементов данных

Информационная характеристика	Элементы данных
Внешние Вводы	Поля ввода данных, сообщения об ошибках, вычисляемые значения, кнопки
Внешние Выводы	Поля данных в отчетах, вычисляемые значения, сообщения об ошибках, заголовки столбцов, которые читаются из внутреннего файла
Внешние Запросы	Вводимые элементы: поле, используемое для поиска, щелчок мыши. Выводимые элементы – отображаемые на экране поля

Таблица 3 –Правила учета элементов данных из графического интерфейса пользователя

Элемент данных	Правило учета
Группа радиокнопок	Так как в группе пользователь выбирает только одну радиокнопку, все радиокнопки группы считаются одним элементом данных
Группа флажков (переключателей)	Так как в группе пользователь может выбрать несколько флажков, каждый флажок считают элементом данных
Командные кнопки	Командная кнопка может определять действие добавления, изменения, запроса. Кнопка ОК может вызывать транзакции (различных типов). Кнопка Next может быть входным элементом запроса или вызывать другую транзакцию. Каждая кнопка считается отдельным элементом данных
Списки	Список может быть внешним запросом, но результат запроса может быть элементом данных внешнего ввода

Например, GUI для обслуживания клиентов может иметь поля Имя, Адрес, Город, Страна, Почтовый Индекс, Телефон, Email. Таким образом, имеется 7 полей или семь элементов данных. Восьмым элементом данных может быть командная кнопка (добавить, изменить, удалить). В этом случае каждый из внешних вводов Добавить, Изменить, Удалить будет состоять из 8 элементов данных (7 полей плюс командная кнопка).

Таблица 4 – Ранг и оценка сложности внешних вводов

Ссылки на файлы	Элементы данных		
	1-4	5-15	>15
0-1	Низкий (3)	Низкий (3)	Средний (4)
2	Низкий (3)	Средний (4)	Высокий (6)
>2	Средний (4)	Высокий (6)	Высокий (6)

Таблица 5 – Ранг и оценка сложности внешних выводов

Ссылки на файлы	Элементы данных		
	1-4	5-19	>19
0-1	Низкий (4)	Низкий (4)	Средний (5)
2-3	Низкий (4)	Средний (5)	Высокий (7)
>3	Средний (5)	Высокий (7)	Высокий (7)

Таблица 6 – Ранг и оценка сложности внешних запросов

Ссылки на файлы	Элементы данных		
	1-4	5-19	>19
0-1	Низкий (3)	Низкий (3)	Средний (4)
2-3	Низкий (3)	Средний (4)	Высокий (6)
>3	Средний (4)	Высокий (6)	Высокий (6)

Таблица 7 – Ранг и оценка сложности внутренних логических файлов

Типы элементов-записей	Элементы данных		
	1-19	**20-50**	**>50**
1	Низкий (7)	Низкий (7)	Средний (10)
2-5	Низкий (7)	Средний (10)	Высокий (15)
>5	Средний (10)	Высокий (15)	Высокий (15)

Таблица 8 – Ранг и оценка сложности внешних интерфейсных файлов

Типы элементов-записей	Элементы данных		
	1-19	20-50	>50
1	Низкий (5)	Низкий (5)	Средний (7)
2-5	Низкий (5)	Средний (7)	Высокий (10)
>5	Средний (7)	Высокий (10)	Высокий (10)

Таблица 9 – Исходные данные для расчета FP-метрик

Имя характеристики	Ранг, сложность, количество			
	Низкий	Средний	Высокий	Итого
Внешние вводы	□x3 = __	□x4 = __	□x6 = __	= □
Внешние выводы	□x4 = __	□x5 = __	□x7 = __	= □
Внешние запросы	□x3 = __	□x4 = __	□x6 = __	= □
Внутренние логические файлы	□x7 = __ □x5 = __	□x 10= __ □x7 = __	□x15 = __ □x10 = __	= □ = □
Внешние интерфейсные файлы				
Общее количество				= □

Количество функциональных указателей вычисляется по формуле

$$FP = Общееколичество * (0{,}65 + 0{,}01 * \sum_{i=1}^{14} F_i), \qquad (5)$$

где F_i коэффициенты регулировки сложности.

Каждый коэффициент может принимать следующие значения: 0 – нет влияния, 1 – случайное, 2 – небольшое, 3 – среднее, 4 – важное, 5 – основное.

Значения выбираются эмпирически в результате ответа на 14 вопросов, которые характеризуют системные параметры приложения (таблица10).

Таблица 10 – Определение системных параметров приложения

Системный параметр	Описание
1 Передачи данных	Сколько средств связи требуется для передачи или обмена информацией с приложением или системой?
2 Распределенная обработка данных	Как обрабатываются распределенные данные и функции обработки?
3 Производительность	Нуждается ли пользователь в фиксации времени ответа или производительности?.
4 Распространенность используемой конфигурации	Насколько распространена текущая аппаратная платформа, на которой будет выполняться приложение?
5 Скорость транзакций	Как часто выполняются транзакции? (каждый день, каждую неделю, каждый месяц)
6 Оперативный ввод данных	Какой процент информации надо вводить в режиме онлайн?
7 Эффективность работы конечного пользователя	Приложение проектировалось для обеспечения эффективной работы конечного пользователя?
8 Оперативное обновление	Как много внутренних файлов обновляется в онлайновой транзакции?
9 Сложность обработки	Выполняет ли приложение интенсивную логическую или математическую обработку?
10 Повторная используемость	Приложение разрабатывалось для удовлетворения требований одного или многих пользователей?
11 Легкость инсталляции	Насколько трудны преобразование и инсталляция приложения?
12 Легкость эксплуатации	Насколько эффективны и/или автоматизированы процедуры запуска, резервирования и восстановления?
13 Разнообразные условия размещения	Была ли спроектирована, разработана и поддержана возможность инсталляции приложения в разных местах для различных организаций?
14 Простота изменений	Была ли спроектирована, разработана и поддержана в приложении простота изменений?

После вычисления FP на его основе формируются метрики производительности, качества и т. д.:

$$\text{Производитель} = \frac{\text{ФункцУказатель}}{\text{Затраты}} \quad \frac{\text{FP}}{\text{чел.} \quad \text{мес}} \; ; \tag{6}$$

$$\text{Качество} = \frac{\text{Ошибки}}{\text{ФункцУказатель}} \quad \frac{\text{Единиц}}{\text{FP}} \; ; \tag{7}$$

$$\text{Удельная стоимость} = \frac{\text{Стоимость}}{\text{ФункцУказатель}} \quad \frac{\text{Тыс.\$}}{\text{FP}} \; ; \tag{8}$$

$$\text{Документированность} = \frac{\text{СтраницДокумента}}{\text{ФункцУказатель}} \quad \frac{\text{Страниц}}{\text{FP}} \; . \tag{9}$$

Для вычисления указателя свойств добавляется одна характеристика – *количество алгоритмов.* Алгоритм здесь определяется как ограниченная подпрограмма вычислений, которая включается в общую компьютерную программу. Примеры алгоритмов: обработка прерываний, инвертирование матрицы, расшифровка битовой строки. Для формирования указателя свойств составляется таблица 11.

Таблица 11 – Исходные данные для расчета указателя свойств

Характеристика	Количество	Сложность	Итого
1 Вводы	□	x4	= □
2 Выводы	□	x5	= □
3 Запросы	□	x4	= □
4 Логические файлы	□	x7	= □
5 Интерфейсные файлы	□	x7	= □
6 Количество алгоритмов	□	x3	= □
Общее количество			= □

После заполнения таблицы по формуле (5) вычисляется значение указателя свойств. Для сложных систем реального времени это значение на 25-30% больше значения, вычисляемого по таблице для количества функциональных указателей.

FP-оценки легко пересчитать в LOC-оценки. Как показано в таблице 12, результаты пересчета зависят от языка программирования, используемого для реализации ПО.

Таблица 12 – Пересчет FP-оценок в LOC-оценки

Язык программирования	Количество операторов на один FP
Ассемблер	320
C	128
Кобол	106
Фортран	106
Паскаль	90
C++	64
Java	53
Ada 95	49
Visual Basic	32
Visual C++	34
Delphi Pascal	29
Smalltalk	22
Perl	21
HTML3	15
LISP	64
Prolog	64
Miranda	40
Haskell	38

Выполнение оценки проекта на основе LOC- и FP-метрик

Выполнение оценки проекта на основе LOC- и FP-метрик состоит из 6 шагов.

Шаг 1. Область назначения проектируемого продукта разбивается на ряд функций, каждую из которых можно оценить индивидуально:

$$f_1, f_2, ..., f_n.$$

Шаг 2. Для каждой функции f_i, планировщик формирует лучшую $LOC_{лучшуюі}\,(FP_{лучшуюі})$, худшую $LOC_{худші}\,(FP_{худші})$ и вероятную оценку $LOC_{вероятні.}(FP_{вероятні})$. Используются опытные данные (из метрического базиса) или интуиция. Диапазон значения оценок соответствует степени предусмотренной неопределенности.

Шаг 3. Для каждой функции в соответствии с β-распределением вычисляется ожидаемое значение *LOC*- (или *FP*-) оценки:

$$LOC_{ожі} = (LOC_{лучші} + LOC_{худші} + 4 \times LOC_{вероятні}) / 6. \quad (10)$$

Шаг 4. Определяется значение *LOC*- или *FP*-производительности разработки функции.

Используется один из трех подходов:

1) для всех функций принимается одна и та же метрика средней производительности $ПРОИЗВ_{ср.}$, взятая из метрического базиса;

2) для *i*-й функции на основе метрики средней производительности вычисляется настраиваемая величина производительности:

$$ПРОИЗВ_i = ПРОИЗВ_{ср.} * (LOC_{ср.} / LOC_{ожі}), \quad (11)$$

где $LOC_{ср.}$ – средняя LOC-оценка, взятая из метрического базиса (соответствует средней производительности);

3) для *i*-й функции настраиваемая величина производительности вычисляется по аналогу, взятому из метрического базиса:

$$ПРОИЗВ_i = ПРОИЗВ_{ані} * (LOC_{ані} / LOC_{ожі}). \quad (12)$$

Первый подход обеспечивает минимальную точность (при максимальной простоте вычислений), а третий подход – максимальную точность (при максимальной сложности вычислений).

Шаг 5. Вычисляется общая оценка затрат на проект: для первого подхода

$$ЗАТРАТЫ = (\sum_{i=1}^{n} LOC_{ожі}) / ПРОИЗВ_{ср} [чел. - мес.]; \quad (13)$$

для второго и третьего подходов

$$ЗАТРАТЫ = (\sum_{i=1}^{n} LOC_{ожі}) / ПРОИЗВ_i [чел. - мес.] \quad (14)$$

Шаг 6. Вычисляется общая оценка стоимости проекта: для первого и второго подходов

$$СТОИМОСТЬ = \sum_{i=1}^{n} LOC_{ожi} \times УД\_СТОИМОСТЬ_{ср}), \quad (15)$$

где $УД\_СТОИМОСТЬ_{ср}$ – метрика средней стоимости одной строки, взятая из метрического базиса.

Для третьего подхода

$$СТОИМОСТЬ = \sum_{i=1}^{n} LOC_{ожi} \times УД\_СТОИМОСТЬ_{аni}), \quad (16)$$

где $УД\_СТОИМОСТЬ_{аni}$ – метрика стоимости одной строки аналога, взятая из метрического базиса.

Пример расчета по методике LOC и FP

Предварительная оценка программного проекта

Предположим, что поступил заказ от концерна «СУПЕРАВТО». Необходимо создать ПО для рабочей станции дизайнера автомобиля (РДА). Заказчик определил проблемную область проекта в своей спецификации:

- ПО РДА должно формировать 2- и 3-мерные изображения для дизайнера;
- дизайнер должен вести диалог с РДА и управлять им с помощью стандартизованного графического пользовательского интерфейса;
- геометрические данные и прикладные данные должны содержаться в базе данных РДА;
- модули проектного анализа рабочей станции должны формировать данные для широкого класса дисплеев SVGA;
- ПО РДА должно управлять и вести диалог со следующими периферийными устройствами: мышь, дигитайзер (графический планшет для ручного ввода), плоттер (графопостроитель), сканер, струйный и лазерный принтеры.

Будем считать, что эта работа проделана и что идентифицированы следующие основные функции ПО:

1. Средства управления пользовательским интерфейсом СУПИ.

2. Анализ двухмерной графики А2Г.

3. Анализ трехмерной графики А3Г.

4. Управление базой данных УБД.

5. Средства компьютерной дисплейной графики КДГ.

6. Управление периферией УП.

7. Модули проектного анализа МПА.

Теперь нужно оценить каждую из функций количественно, с помощью LOC-оценки. По каждой функции эксперты предоставляют лучшее, худшее и вероятное значения. Ожидаемую LOC-оценку реализации функции определяем по формуле:

$$LOC_{ожі} = (LOC_{лучші} + LOC_{худші} + 4 \times LOC_{вероятні}) / 6,$$

результаты расчетов заносим в таблицу 13.

Таблица 13 – Начальная таблица оценки проекта

Функция	Лучш. [LOC]	Вероят. [LOC]	Худш. [LOC]	Ожид. [LOC]	Уд. стои-мость [$/LOC]	Стои-мость [$]	Произв. [LOC/ [чел-мес]	Затра тра-ты [чел-мес]
СУПИ	1800	2400	2650	2340				
А2Г	4100	5200	7400	5380				
А3Г	4600	6900	8600	6800				
УБД	2950	3400	3600	3350				
КДГ	4050	4900	6200	4950				
УП	2000	2100	2450	2140				
МПА	6600	8500	9800	8400				
Итого				33360				

Для определения удельной стоимости и производительности обратились в архив фирмы, где хранятся данные метрического базиса, собранные по уже выполненным проектам. Предположим, что из метрического базиса извлечены данные по функциям-аналогам, представленные в таблице 14.

Таблица 14 – Данные из метрического базиса фирмы

Функция	$LOC_{ані}$	$УД\_СТОИМОСТЬ_{ані}$ [$/ LOC]	$ПРОИЗВ_{ані}$ [LOC/чел-мес]
СУПИ	585	14	1260
А_Г	3000	20	440
УБД	1117	18	720
КДГ	2475	22	400
УП	214	28	1400
МПА	1400	18	1800

Считается, что удельная стоимость строки является константой и не изменяется от реализации к реализации. Следовательно, стоимость разработки каждой функции рассчитываем по формуле:

$$СТОИМОСТЬ_i = LOC_{ожі} \times УД\_СТОМОСТЬ_{ані}.$$

Для вычисления производительности разработки каждой функции выберем самый точный подход — подход настраиваемой производительности:

$$ПРОИЗВ_i = ПРОИЗВ_{ані} \times (LOC_{ані} / LOC_{ожі}).$$

Соответственно, затраты на разработку каждой функции будем определять по выражению:

$$ЗАТРАТЫ_i = (LOC_{ожі} / ПРОИЗВ_i) \text{ [чел.-мес].}$$

Конечные оценки проекта приведены в таблице 15.

Таблица 15 – Конечная таблица оценки проекта

Функция	Лучш.	Вероят.	Худш.	Ожид. [LOC]	Уд. стоимость [S/LOC]	Стоимость [$]	Произв. [LOC/ чел.-мес]	Затраты [чел-мес]
СУПИ	1800	2400	2650	2340	14	32760	315	7,4
А2Г	4100	5200	7400	5380	20	107600	245	21,9
А3Г	4600	6900	8600	6800	20	136000	194	35,0
УБД	2950	3400	3600	3350	18	60300	240	13,9
КДГ	4050	4900	6200	4950	22	108900	200	24,7
УП	2000	2100	2450	2140	28	59920	140	15,2
МПА	6600	8500	9800	8400	18	151200	300	28,0
Итого				33360		656680		146

Учитывая важность полученных результатов, проверим расчеты с помощью FP-указателей. На данном этапе оценивания разумно допустить, что все информационные характеристики имеют средний уровень сложности. В этом случае результаты экспертной оценки принимают вид, представленный в таблицах 16,17.

Таблица 16 – Оценка информационных характеристик проекта

Характеристика	Лучш.	Вероят.	Худш.	Ожид.	Сложность	Количество
Вводы	20	24	30	24	x 4	96
Выводы	12	15	22	16	x 5	80
Запросы	16	22	28	22	x 4	88
Логические файлы	4	4	5	4	x 10	40
Интерфейсные файлы	2	2	3	2	x 7	14
Общее количество						318

Таблица 17 – Оценка системных параметров проекта

Коэффициент регулировки сложности		Оценка
F_1	Передачи данных	2
F_2	Распределенная обработка данных	0
F_3	Производительность	4
F_4	Распространенность используемой конфигурации	3
F_5	Скорость транзакций	4
F_6	Оперативный ввод данных	5
F_7	Эффективность работы конечного пользователя	5
F_8	Оперативное обновление	3
F_9	Сложность обработки	5
F_{10}	Повторная используемость	4
F_{11}	Легкость инсталляции	3
F_{12}	Легкость эксплуатации	4
F_{13}	Разнообразные условия размещения	5
F_{14}	Простота изменений	5
	Итого	52

Таким образом, получаем:

$$FP = Общее\_количество \times (0{,}65 + 0{,}01 \times \sum_{i=1}^{14} F_i) = 318 \times 1{,}17 = 372.$$

Используя значение производительности, взятое в метрическом базисе фирмы,

$$Производительность = 2{,}55[FP / чел. - мес.],$$

вычисляем значения затрат и стоимости:

$$Затраты = FP / Призводительность = 145{,}9[чел. - мес.],$$

$$Стоимость = Затраты \times \$4500 = \$656500.$$

4 Практическое задание №4. Расчеты затрат при проектировании программного обеспечения

1. Выполнить расчеты по конструктивной модели стоимости
2. Выполнить расчеты по модели композиции приложения
3. Выполнить расчеты по модели раннего этапа проектирования
4. Выполнить расчеты по модели этапа постархитектуры

Краткие теоретические основы для выполнения практической работы

Конструктивная модель стоимости

Подмодели COCOMO 81 (Constructive Cost Model) могут применяться к трем типам программных проектов. По терминологии Боэма, их образуют:

- *распространенный тип* – небольшие программные проекты, над которыми работает небольшая группа разработчиков с хорошим стажем работы, устанавливаются мягкие требования к проекту;
- *полунезависимый тип* – средний по размеру проект, выполняется группой разработчиков с разным опытом, устанавливаются как мягкие, так и жесткие требования к проекту;
- *встроенный тип* – программный проект разрабатывается в условиях жестких аппаратных, программных и вычислительных ограничений.

Уравнения базовой подмодели имеют вид

$$E = a_b * (KLOC)^{b_b} \text{ [чел-мес];} \quad (1)$$

$$D = c_b * (E)^{d_b} \text{ [мес],} \quad (2)$$

где E – затраты в человеко-месяцах,

D – время разработки, KLOC –количество строк в программном продукте.

Коэффициенты a_b, b_b, c_b, d_b берутся из таблицы 1.

Таблица 1 – Коэффициенты для базовой подмодели COCOMO 81

Тип проекта	a_b	b_b	c_b	d_b
Распространенный	2,4	1,05	2,5	0,38
Полунезависимый	3,0	1,12	2,5	0,35
Встроенный	3,6	1,20	2,5	0,32

COCOMO II

В состав COCOMO II входят:

- модель композиции приложения;
- модель раннего этапа проектирования;
- модель этапа пост-архитектуры.

Модель композиции приложения

Модель композиции приложения ориентирована на применение объектных указателей.

Объектный указатель – средство косвенного измерения ПО, для его расчета определяется количество экранов (как элементов пользовательского интерфейса), отчетов и компонентов, требуемых для построения приложения. Как показано в таблице 2, каждый объектный экземпляр (экран, отчет) относят к одному из трех уровней сложности. Здесь места подстановки измеренных и вычисленных значений отмечены прямоугольниками (прямоугольник играет роль метки-заполнителя). В свою очередь, сложность является функцией от параметров клиентских и серверных таблиц данных (см. таблицы 3 и 4), которые требуются для генерации экрана и отчета, а также от количества представлений и секций, входящих в экран или отчет.

Таблица 2 – Оценка количества объектных указателей

Тип объекта	Количество	Вес			Итого
		Простой	Средний	Сложный	
Экран	□	*1	*2	*3	= □
Отчет	□	*2	*5	*8	= □
3GL компонент	□			*10	= □
Объектные указатели					= □

Таблица 3 – Оценка сложности экрана

Экраны	Количество серверных (срв) и клиентских (клт) таблиц данных		
Количество представлений	Всего < 4 (< 2 срв, <3клт)	Всего < 8 (2-3 срв, 3-5 клт)	Всего > 8 (>3срв, >5клт)
<3	Простой	Простой	Средний
3-7	Простой	Средний	Сложный
>8	Средний	Сложный	Сложный

Таблица 4 – Оценка сложности отчета

Отчеты	Количество серверных (срв) и клиентских (клт) таблиц данных		
Количество представлений	Всего < 4 (< 2 срв, < 3 клт)	Всего < 8 (2-3 срв, 3-5 клт)	Всего > 8 (>3срв, > 5 клт)
0 или 1	Простой	Простой	Средний
2 или 3	Простой	Средний	Сложный
>4	Средний	Сложный	Сложный

После определения сложности количество экранов, отчетов и компонентов взвешивается в соответствии с таблицей 1. Количество объектных указате-

лей определяется перемножением исходного числа объектных экземпляров на весовые коэффициенты и последующим суммированием промежуточных результатов.

Для учета реальных условий разработки вычисляется процент повторного использования программных компонентов %REUSE и определяется количество новых объектных указателей NOP:

$$NOP = (Объектные\_указатели) * [(100 - \%REUSE)/100]. \quad (3)$$

Для оценки затрат, основанной на величине NOP, надо знать скорость разработки продукта PROD. Эту скорость определяют по таблице 5, учитывающей уровень опытности разработчиков и зрелость среды разработки.

Проектные затраты оцениваются по формуле:

$$ЗАТРАТЫ = NOP / PROD \text{ [чел.-мес]}, \quad (4)$$

где PROD – производительность разработки, выраженная в терминах объектных указателей.

Таблица 5 – Оценка скорости разработки

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

В более развитых моделях дополнительно учитывается множество масштабных факторов, формирователей затрат, процедур поправок.

Модель раннего этапа проектирования

Модель раннего этапа проектирования используется в период, когда стабилизируются требования и определяется базисная программная архитектура.

Основное уравнение этой модели имеет следующий вид:

$$ЗАТРАТЫ = A * РАЗМЕР^{B} * M_e + ЗАТРАТЫ_{auto} \text{ [чел.-мес]}, \quad (5)$$

где:

- масштабный коэффициент A = 2,5;
- показатель B отражает нелинейную зависимость затрат от размера проекта (размер системы *РАЗМЕР* выражается в тысячах *LOC*);
- множитель поправки M_e зависит от 7 формирователей затрат, характеризующих продукт, процесс и персонал;
- слагаемое $ЗАТРАТЫ_{auto}$ отражает затраты на автоматически генерируемый программный код.

Значение показателя степени B изменяется в диапазоне 1,01... 1,26, зависит от пяти масштабных факторов W_i и вычисляется по формуле

$$B = 1{,}01 + 0{,}01\sum_{i=1}^{5} Wi. \quad (6)$$

Общая характеристика масштабных факторов приведена в таблице 6, а таблица 7 позволяет определить оценки этих факторов. Оценки принимают 6 значений: от очень низкой (5) до сверхвысокой (0).

Таблица 6 – Характеристика масштабных факторов

Масштабный фактор (W_i)	Пояснение
Предсказуемость PREC	Отражает предыдущий опыт организации в реализации проектов этого типа. Очень низкий означает отсутствие опыта. Сверхвысокий означает, что организация полностью знакома с этой прикладной областью
Гибкость разработки FLEX	Отражает степень гибкости процесса разработки. Очень низкий означает, что используется заданный процесс. Сверхвысокий означает, что клиент установил только общие цели
Разрешение архитектуры /риска RESL	Отражает степень выполняемого анализа риска. Очень низкий означает малый анализ. Сверхвысокий означает полный и сквозной анализ риска
Связность группы TEAM	Отражает, насколько хорошо разработчики группы знают друг друга и насколько удачно они совместно работают. Очень низкий означает очень трудные взаимодействия. Сверхвысокий, означает интегрированную группу, без проблем взаимодействия
Зрелость процесса PMAT	Означает зрелость процесса в организации. Вычисление этого фактора может выполняться по вопроснику CMM

В качестве иллюстрации рассмотрим компанию, которая берет проект в малознакомой проблемной области. Положим, что заказчик не определил используемый процесс разработки и не допускает выделения времени на всесторонний анализ риска. Для реализации этой программной системы нужно создать новую группу разработчиков. Компания имеет возможности, соответ-

ствующие 2-му уровню зрелости согласно модели CMM. Возможны следующие значения масштабных факторов:

- предсказуемость. Это новый проект для компании – значение Низкий (4);
- гибкость разработки. Заказчик требует некоторого согласования – значение Очень высокий (1);
- разрешение архитектуры/риска. Не выполняется анализ риска, как следствие, малое разрешение риска – значение Очень низкий (5);
- связность группы. Новая группа, нет информации – значение Номинальный (3);
- зрелость процесса. Имеет место некоторое управление процессом – значение Номинальный (3).

Таблица 7 – Оценка масштабных факторов

Масштабный фактор (W_i)	Очень низкий 5	Низкий 4	Номинальный 3	Высокий 2	Очень высокий 1	Сверхвысокий 0
PREC	Полностью непредсказуемый проект	Главным образом, в значительной степени непредсказуемый	Отчасти непредсказуемый	Большей частью знакомый	В значительной степени знакомый	Полностью знакомый
FLEX	Точный, строгий процесс разработки	Редкое расслабление в работе	Некоторое расслабление в работе	Большей частью согласованный процесс	Некоторое согласование 36процесса	Заказчик определил только общие цели
RESL	Малое разрешение риска (20%)	Некоторое (40%)	Частое (60%)	Большей частью (75%)	Почти всегда (90%)	Полное (100%)

Продолжение таблицы 7

Масштабный фактор (W_i)	Очень низкий 5	Низкий 4	Номинальный 3	Высокий 2	Очень высокий 1	Сверхвысокий 0
TEAM	Очень трудное взаимодействие	Достаточно трудное взаимодействие	Среднее взаимодействие	Главным образом кооперативность	Высокая кооперативность	Безукоризненное взаимодействие
PREC	Полностью непредсказуемый проект	В значительной степени непредсказуемый	Отчасти непредсказуемый	Большей частью знакомый	В значительной степени знакомый	Полностью знакомый
PMAT	Взвешенное среднее значение от количества ответов «Yes» на вопросник CMM Maturity		Взвешенное среднее значение от количества ответов «Yes» на вопросник CMM Maturity			

Сумма этих значений равна 16, поэтому конечное значение степени B= 1,17. Вернемся к обсуждению основного уравнения модели раннего этапа проектирования. Множитель поправки M_e зависит от набора формирователей затрат, перечисленных в таблице 8.

Для каждого формирователя затрат определяется оценка (по 6-балльной шкале), где 1 соответствует очень низкому значению, а 6 – сверхвысокому значению. На основе оценки для каждого формирователя по таблице Боэма (приложение А) определяется множитель затрат EM_i Перемножение всех множителей затрат формирует множитель поправки:

$$M_e = \prod_{i=1}^{7} EM_i \,. \qquad (7)$$

Слагаемое $ЗАТРАТЫ_{auto}$ используется, если некоторый процент программного кода генерируется автоматически. Поскольку производительность такой работы значительно выше, чем при ручной разработке кода, требуемые затраты вычисляются отдельно, по следующей формуле:

$$ЗАТРАТЫ_{auto} = (KALOC * (AT / 100)) / ATPROD, \qquad (8)$$

где:

- KALOC – количество строк автоматически генерируемого кода (в тысячах строк);
- AT – процент автоматически генерируемого кода (от всего кода системы);
- ATPROD – производительность автоматической генерации кода.

Сомножитель AT в этой формуле позволяет учесть затраты на организацию взаимодействия автоматически генерируемого кода с оставшейся частью системы.

Далее затраты на автоматическую генерацию добавляются к затратам, вычисленным для кода, разработанного вручную.

Таблица 8 – Формирователи затрат для раннего этапа проектирования

Обозначение	Название
PERS	Возможности персонала (Personnel Capability)
RCPX	Надежность и сложность продукта (Product Reliability and Complexity)
RUSE	Требуемое повторное использование (Required Reuse)
PDIF	Трудность платформы (Platform Difficulty)
PREX	Опытность персонала (Personnel Experience)
FCIL	Средства поддержки (Facilities)
SCED	График (Schedule)

Модель этапа постархитектуры

Модель этапа постархитектуры используется в период, когда уже сформирована архитектура и выполняется дальнейшая разработка программного продукта.

Основное уравнение постархитектурной модели является развитием уравнения предыдущей модели и имеет следующий вид:

$$ЗАТРАТЫ = A * К_{\sim req} * РАЗМЕР^{B} * М_{p} + ЗАТРАТЫ_{auto} \text{ [чел.-мес]}, \quad (9)$$

где

- коэффициент $К_{\sim req}$ учитывает возможные изменения в требованиях;
- показатель B отражает нелинейную зависимость затрат от размера проекта (размер выражается в *KLOC*), вычисляется так же, как и в предыдущей модели;
- в размере проекта различают две составляющие – новый код и повторно используемый код;
- множитель поправки $М_{p}$ зависит от 17 факторов затрат, характеризующих продукт, аппаратуру, персонал и проект.

Изменчивость требований приводит к повторной работе, требуемой для учета предлагаемых изменений, оценка их влияния выполняется по формуле

$$K_{\sim reg} = 1 + (BRAK/100), \quad (10)$$

где *BRAK* – процент кода, отброшенного (модифицированного) из-за изменения требований.

Размер проекта и продукта определяют по выражению

$$РАЗМЕР = РАЗМЕР_{new} + РАЗМЕР_{reuse}[KLOC], \quad (11)$$

где

- $РАЗМЕР_{new}$ – размер нового (создаваемого) программного кода;
- $РАЗМЕР_{reuse}$ – размер повторно используемого программного кода.

Формула для расчета размера повторно используемого кода записывается следующим образом:

$$РАЗМЕР_{reuse} = KASLOC * ((100 - AT)/100) * (AA + SU + 0{,}4DM + 0{,}3CM + 0{,}3IM)/100, \quad (12)$$

где

- *KASLOC* – количество строк повторно используемого кода, который должен быть модифицирован (в тысячах строк);
- *AT* – процент автоматически генерируемого кода;
- *DM* – процент модифицируемых проектных моделей;
- *CM* – процент модифицируемого программного кода;
- *IM* – процент затрат на интеграцию, требуемых для подключения повторно используемого ПО;
- *SU* – фактор, основанный на стоимости понимания добавляемого ПО; изменяется от 50 (для сложного неструктурированного кода) до 10 (для хорошо написанного объектно-ориентированного кода);
- *AA* – фактор, который отражает стоимость решения о том, может ли ПО быть повторно используемым; зависит от размера требуемого тестирования и оценивания (величина изменяется от 0 до 8).

Правила выбора этих параметров приведены в руководстве по COCOMO II.

Для определения множителя поправки M_p основного уравнения используют 17 факторов затрат, которые могут быть разбиты на 4 категории. Перечислим факторы затрат, сгруппировав их по категориям.

Факторы продукта:

1) требуемая надежность ПО – RELY;

2) размер базы данных – DATA;

3) сложность продукта – CPLX;

4) требуемая повторная используемость – RUSE;

5) документирование требований жизненного цикла – DOCU.

Факторы платформы (виртуальной машины):

6) ограничения времени выполнения – TIME;

7) ограничения оперативной памяти – STOR;

8) изменчивость платформы – PVOL.

Факторы персонала:

9) возможности аналитика – ACAP;

10) возможности программиста – PCAP;

11) опыт работы с приложением – AEXP;

12) опыт работы с платформой – PEXP;

13) опыт работы с языком и утилитами – LTEX;

14) непрерывность персонала – PCON.

Факторы проекта:

15) использование программных утилит – TOOL;

16) мультисетевая разработка – SITE;

17) требуемый график разработки – SCED.

Для каждого фактора определяется оценка (по 6-балльной шкале). На основе оценки для каждого фактора по таблице Боэма (приложение А) определяется множитель затрат EM_i. Перемножение всех множителей затрат дает множитель поправки пост-архитектурной модели:

$$M_p = \prod_{i=1}^{17} EM_i \,. \tag{13}$$

Значение M_p отражает реальные условия выполнения программного проекта и позволяет троекратно увеличить (уменьшить) начальную оценку затрат.

От оценки затрат можно перейти к стоимости проекта. Переход выполняют по формуле:

$$СТОИМОСТЬ = ЗАТРАТЫ \times РАБ\_КОЭФ, \tag{14}$$

где среднее значение рабочего коэффициента составляет \$15 000 за человеко-месяц.

После определения затрат и стоимости можно оценить длительность разработки. Модель COCOMO II содержит уравнение для оценки календарного

времени TDEV, требуемого для выполнения проекта. Для моделей всех уровней справедливо:

$$\textit{Длительность}(TRED) = [3{,}0 \times (\textit{ЗАТРАТЫ})^{(0{,}33+0{,}2(B-1{,}01))}] * SCEDPercentage / 100 \quad \text{[мес]}, \qquad (15)$$

где

- B – ранее рассчитанный показатель степени;
- *SCEDPercentage* – процент увеличения (уменьшения) номинального графика.

Если нужно определить номинальный график, то принимается SCEDPercentage =100 и правый сомножитель в уравнении обращается в единицу. Следует отметить, что COCOMO II ограничивает диапазон уплотнения/растягивания графика (от 75% до 160%). Причина проста – если планируемый график существенно отличается от номинального, это означает внесение в проект высокого риска.

Пример

Проверка с помощью модели COCOMO II

Примем, что все масштабные факторы и факторы затрат имеют номинальные значения. В силу этого показатель степени B = 1,16, а множитель поправки M_p= 1. Кроме того, будем считать, что автоматическая генерация кода и повторное использование компонентов не предусматриваются. Следовательно, мы вправе применить формулу

$$\textit{ЗАТРАТЫ} = A \times \textit{РАЗМЕР}^{B} \text{ [чел.-мес]}$$

и получаем:

$$\textit{ЗАТРАТЫ} = 2{,}5 \times (33{,}3)^{1{,}16} = 145{,}87 \text{ [чел.-мес].}$$

Соответственно, номинальная длительность проекта равна

$$\textit{Длительность} = [3{,}0 \times (\textit{ЗАТРАТЫ})^{(0{,}33+0{,}2(B-1{,}01))}] = 3 \times (145{,}87)^{0{,}36} = 18 \text{ [мес].}$$

Заключение

В методических указаниях приведены оценки затрат при проектировании программного проекта. Показаны размерно-ориентированные и функционально-ориентированные метрики затрат, методика их применения. Описана модель для оценивания затрат – COCOMO II. В качестве иллюстраций приводятся примеры предварительного оценивания проекта, анализа влияния на проект конкретных условий разработки.

В результате выполнения практических заданий студенты закрепляют теоретические знания и формируют практические навыки оценки затрат при разработки программного обеспечения.

Представленный теоретический и справочный материал позволяет использовать рекомендации по предварительной оценки затрат при проектировании программного обеспечения.

Список использованных источников

1 Кулямин, В. В. Технологии программирования. Компонентный подход: / В. В. Кулямин . - М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2011. - 463 с. : ил. ISBN 978-5-94774-544-3 (БИНОМ.ЛЗ), ISBN 5-9556-0067-1 (ИНТРУИТ.РУ)

2 Орлов, С. А. Технологии разработки программного обеспечения: разработка сложных программных систем: учеб. для вузов / С. А. Орлов .- 3-е изд. - СПб. [и др.] : Питер, 2004. - 527 с.: ил. ISBN 5-94723-145-X

3 Пилон, Д. Управление разработкой ПО/ Д. Пилон, Р. Майлз. – СПб.: Питер, 2011. – 464 с.: ил. ISBN 978-5-459-00522-6

4 Брауде, Э. Технология разработки программного обеспечения/ Э. Брауде. – СПб.: Питер, 2004. – 655 с.: ил. ISBN 5-94723-663-X

5 Гагарина, Л. Г. Технология разработки программного обеспечения: учеб. пособие для вузов по направлению "Информатика и вычислительная техника" / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Виснадул; под ред. Л. Г. Гагариной. - М. : ИД «ФОРУМ»: ИНФРА-М, 2008. – 400 с: ил. ISBN 978-5-8199-0342-1 (ИД «ФОРУМ») ISBN 978-5-16-003193-4 (ИНФРА-М)

6 Иванова, Г. С. Технология программирования: учебник для вузов / Г. С. Иванова .- 3-е изд., перераб. и доп. - М. : МГТУ им. Н.Э. Баумана, 2006. - 336 с.: ил. ISBN ISBN 5-7038-2077-4

Приложение А
(справочное)
Факторы затрат постархитектурной модели COCOMO II

Значительную часть времени при использовании модели COCOMO II занимает работа с факторами затрат. Это приложение содержит описание таблиц Боэма, обеспечивающих оценку факторов затрат.

Факторы продукта

Таблица А.1 –Требуемая надежность ПО (Required Software Reliability) RELY

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверх высокий
RELY	Легкое беспокойство	Низкая, легко восстанавливаемые потери	Умеренная, легко восстанавливаемые потери	Высокая, финансовые потери	Риск для человеческой жизни	

Таблица А.2 – Размер базы данных (Data Base Size) DATA

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверх-высокий
DATA		Байты БД/ LOC-прогр. < 10	$10 \le D/P < 100$	$100 \le D/P < 1000$	$D/P \ge 1000$	

Примечание: Фактор DATA определяется делением размера БД (D) на длину кода программы (P). Длина программы представляется в LOC-оценках.

Сложность продукта (Product Complexity) CPLX

Сложность продукта определяют по двум следующим таблицам. Выделяют 5 областей применения продукта: операции управления, вычислительные операции, операции с приборами (устройствами), операции управления данными, операции управления пользовательским интерфейсом. Выбирается область или комбинация областей, которые характеризуют продукт или подсистему продукта. Сложность рассматривается как взвешенное среднее значение для этих областей.

Таблица А.3 – Сложность модуля в зависимости от области применения

CPLX	Операции управления	Вычислительные операции	Операции с приборами
Очень низкий	Последовательный код с небольшим количеством структурированных операторов: DO, CASE, IF-THEN-ELSE.Простая композиция модулей с помощью вызовов процедур и простых сценариев	Вычисление простых выражений, например, A=B+C*(D-E)	Простые операторы чтения и записи, использующие простые форматы
Низкий	Несложная вложенность структурированных операторов. В основном простые предикаты	Вычисление выражений средней сложности, например D=SQRT(B**2-4*A*C)	Не требуется знание характеристик конкретного процессора или устройства ввода-вывода. Ввод-вывод выполняется на уровне GET/PUT
Номинальный	В основном простая вложенность. Некоторое межмодульное управление. Таблицы решений. Простые обратные вызовы (callbacks) или передачи сообщений, включение среднего уровня – поддержка распределенной обработки	Использование стандартных математических и статистических подпрограмм. Базовые матричные / векторные операции	Обработка ввода-вывода, включающая выбор устройства, проверку состояния и обработку ошибок

Продолжение таблицы А.3

Высокий	Высокая вложенность операторов с составными предикатами. Управление очередями и стеками. Однородная распределенная обработка. Управление ПО реального времени на единственном процессоре	Базовый численный анализ: мультивариантная интерполяция, обычные дифференциальные уравнения. Базисное усечение, учет потерь точности	Операции ввода-вывода физического уровня (определение адресов физической памяти; поиски, чтения и т. д.). Оптимизированный совмещенный ввод-вывод
Очень высокий	Реентерабельное и рекурсивное программирование. Обработка прерываний с фиксированными приоритетами Синхронизация задач, сложные обратные вызовы, гетерогенная распределенная обработка. Управление однопроцессорной системой в реальном времени	Сложный, но структурированный численный анализ: уравнения с плохо обусловленными матрицами, уравнения в частных производных. Простой параллелизм	Процедуры для диагностики по прерыванию, обслуживание и маскирование прерываний. Обслуживание линий связи. Высокопроизводительные встроенные системы
Сверхвысокий	Планирование множественных ресурсов с динамически изменяющимися приоритетами. Управление на уровне микропрограмм. Управление распределенной аппаратурой в реальном времени	Сложный и неструктурированный численный анализ: высокоточный анализ стохастических данных с большим количеством шумов. Сложный параллелизм	Программирование с учетом временных характеристик приборов, микропрограммные операции. Критические к производительности встроенные системы

Таблица А.4 – Сложность модуля в зависимости от области применения

CPLX	Операции управления данными	Операции управления пользовательским интерфейсом
Очень низкий	Простые массивы в оперативной памяти. Простые запросы к БД, обновления	Простые входные формы, генераторы отчетов
Низкий	Использование одного файла без изменения структуры данных, без редактирования и промежуточных файлов. Умеренно сложные запросы к БД, обновления	Использование билдеров для простых графических интерфейсов
Номинальный	Ввод из нескольких файлов и вывод в один файл. Простые структурные изменения, простое редактирование. Сложные запросы БД, обновления	Простое использование набора графических объектов (widgets)
Высокий	Простые триггеры, активизируемые содержимым потока данных. Сложное изменение структуры данных	Разработка набора графических объектов, его расширение. Простой голосовой ввод-вывод, мультимедиа
Очень высокий	Координация распределенных БД. Сложные триггеры. Оптимизация поиска	Умеренно сложная 2D/3D-графика, динамическая графика, мультимедиа
Сверхвысокий	Динамические реляционные и объектные структуры с высоким сцеплением. Управление данными с помощью естественного языка	Сложные мультимедиа, виртуальная реальность

Таблица А.5 – Требуемая повторная используемость (Required Reusability) RUSE

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
RUSE		Нет	На уровне проекта	На уровне программы	На уровне семейства продуктов	На уровне нескольких семейств продуктов

Таблица А.6 – Документирование требований жизненного цикла (Documentation match to life-cycle needs) DOCU

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
DOCU	Многие требования жизненного цикла не учтены	Некоторые требования жизненного цикла не учтены	Оптимизированы к требованиям жизненного цикла	Избыточны по отношению к требованиям жизненного цикла	Очень избыточны по отношению к ребованиям жизненного цикла	

Факторы платформы (виртуальной машины)

Таблица А.7 – Ограничения времени выполнения (Execution Time Constraint) TIME

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
TIME			Используется ≤ 50% возможного времени выполнения	70%	85%	95%

Таблица А.8 – Ограничения оперативной памяти (Main Storage Constraint) STOR

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
STOR			Используется ≤ 50% доступной памяти	70%	85%	95%

Таблица А.9 – Изменчивость платформы (Platform Volatility) PVOL

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
PVOL		Значительные изменения - каждые 12мес.; незначительные - каждый месяц.	Значительные изменения - каждые 6 мес.; незначительные - каждые 2 недели.	Значительные изменения - 2 мес.; незначительные - 1 неделя	Значительные изменения - 2нед.; незначительные - 2 дня	

Факторы персонала

Таблица А. 10 – Возможности аналитика (Analyst Capability) ACAP

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
ACAP	15%	35%	55%	75%	90%	

Таблица А.11– Возможности программиста (Programmer Capability) PCAP

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
PCAP	15%	35%	55%	75%	90%	

Таблица А. 12 – Опыт работы с приложением (Applications Experience) AEXP

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
AEXP	2 месяца	6 месяцев	1 год	3 года	6 лет	

Таблица А. 13 – Опыт работы с платформой (Platform Experience) PEXP

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
PEXP	2 месяца	6 месяцев	1 год	3 года	6 лет	

Таблица А. 14 – Опыт работы с языком и утилитами (Language and Tool Experience) LTEX

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
LTEX	2 месяца	6 месяцев	1 год	Згода	6 лет	

Таблица А. 15 – Непрерывность персонала (Personnel Continuity) PCON

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
PCON	48%/год	24%/ год	12%/ год	6%/ год	3%/год	

Примечание: С помощью фактора учитывается процент смены персонала

Факторы проекта

Таблица А. 16 – Использование программных утилит (Use of Software Tools) TOOL

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
TOOL	Редактирование, кодирование, отладка	Простая входная, выходная CASE-утилита, малая интеграция	Базовые утилиты жизненного цикла, умеренная интеграция	Развитые утилиты жизненного цикла, умеренная интеграция	Развитые утилиты жизненного цикла, хорошо интегрированные с процессами, методами, повторным использованием	

Таблица А. 17 – Мультисетевая разработка (Multisite Development) SITE

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
SITE: коммуникации	Один телефон, почта	Индивидуаль-ные телефоны, FAX	Узкополосный e- mail	Широкополосные электронные коммуникации	Широкополосные электронные коммуникации, видеоконференции от случая к случаю	Интерактивные мультимедиа

Таблица А. 18 – Требуемый график разработки (Required Development Schedule) SCED

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
SCED	75% от номинального срока	85%	100%	130%	160%	

Таблица А. 19 – Числовые значения множителей затрат

Фактор	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
RELY	Легкое беспокойство 0,75	Низкая, легко восстанавливаемые потери 0,88	Умеренная, легко восстанавливаемые потери 1,00	Высокая, финансовые потери 1,15	Риск для человеческой жизни 1,39	

Продолжение таблицы А.19

DATA		байты БД/LOC прогр. <10 0,93	10 ≤ D/P<100 1,00	100 ≤ D/P< <1000 1,09	D/P ≥1000 1,19	
CPLX	0,75	0,88	1,00	1,15	1,30	1,66
RUSE		Нет 0,91	На уровне проекта 1,00	На уровне программы 1,14	На уровне семейства продуктов 1,29	На уровне нескольких семейств продуктов 1,49
DOCU	Многие требования жизненного цикла не учтены 0,89	Некоторые требования жизненного цикла не учтены 0,95	Оптимизированы к требованиям жизненного цикла 1,00	Избыточны по отношению к требованиям жизненного цикла 1,06	Очень избыточны по отношению к требованиям жизненного цикла 1,13	
TIME			Используется < 50% возможного времени выполнения 1,00	70% 1,11	85% 1,31	95% 1,67
STOR			Используется < 50% доступной памяти 1,00	70% 1,06	85% 1,21	95% 1,57
PVOL		Значиельные изменения - через 1 год; незначетельные - через1 мес. 0,87	Значительные - через 6 мес.; незначительные - через 2 недели 1,00	Значительные – через 2 мес.; незначительные - через 1 неделю 1,15	Значительные – через 2 нед. незначительные - через 2 дня 1,30	
ACAP	15% 1,50	35% 1,22	55% 1,00	75% 0,83	90% 0,67	

Продолжение таблицы А.19

PCAP	15% 1,37	35% 1,16	55% 1,00	75% 0,87	90% 0,74	
PCON	48%/год 1,24	24%/год 1,10	12%/год 1,00	6%/год 0,92	3%/год 0,84	
AEXP	≤2 месяцев 1,22	6 месяцев 1,10	1 год 1,00	3 года 0,89	6 лет 0,81	
PEXP	≤ 2 месяцев 1,25	6 месяцев 1,12	1 год 1,00	3года 0,88	6 лет 0,81	
LTEX	≤2 месяцев 1,22	6 месяцев 1,10	1 год 1,00	3 года 0,91	6 лет 0,84	
TOOL	Редактирование, кодирование, отладка 1,24	Простая входная, выходная CASE утилита, малая интеграция 1,12	Базовые утилиты жизненного цикла, умеренная интеграция 1,00	Развитые утилиты жизненного цикла, умеренная интеграция 0,86	Развитые утилиты жизненного цикла, хорошо интегрированы с процессами, методами, повторным использованием 0,72	
SITE коммуникации	Один телефон, почта 1,25	Индивидуальные телефоны, FAX 1,10	Узкополосный e-mail 1,00	Широкополосные коммуникации 0,92	Широкополосные коммуникации, иногда видеоконференции 0,84	Интерактивные мультимедиа 0,78
SCED	75% от номин. 1,29	85% 1,10	100% 1,00	130% 1,00	160% 1,00	

Таблица А. 19 обеспечивает перевод оценок факторов затрат в числовые значения множителей затрат. Имя фактора определяет строку таблицы, оценка фактора – столбец. На пересечении строки и столбца находим числовое значение множителя. Например, фактору TIME с оценкой Высокий соответствует множитель со значением 1,11.