МЕТРИКИ ПРОГРАММНОГО ПРОЕКТА

В этой главе детально рассматривается такой элемент процесса конструирования ПО, как руководство программным проектом. Читатель знакомится с вопросами планирования проекта, оценки затрат проекта. В данной главе обсуждаются размерно-ориентированные и функционально-ориентированные метрики затрат, методика их применения. Достаточно подробно описывается наиболее популярная модель для оценивания затрат — COCOMO II. В качестве иллюстраций приводятся примеры предварительного оценивания проекта, анализа влияния на проект конкретных условий разработки.

Процесс руководства проектом

Руководство программным проектом — первый слой процесса конструирования ПО. Термин «слой» подчеркивает, что руководство определяет сущность процесса разработки от его начала до конца. Принцип руководства иллюстрирует рис. 2.1.

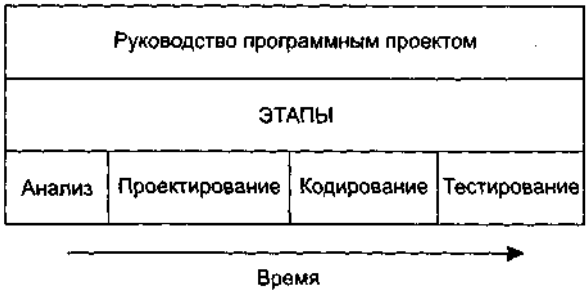

Рис. 2.1. Руководство в процессе конструирования ПО

На этом рисунке прямоугольник обозначает процесс конструирования, в нем выделены этапы, а вверху, над каждым из этапов, размещен слой деятельности *«руководство программным проектом».*

Для проведения успешного проекта нужно понять объем предстоящих работ, возможный риск, требуемые ресурсы, предстоящие задачи, прокладываемые вехи, необходимые усилия (стоимость), план работ, которому желательно следовать. Руководство программным проектом обеспечивает такое понимание. Оно начинается перед технической работой, продолжается по мере развития ПО от идеи к реальности и достигает наивысшего уровня к концу работ [32], [64], [69].

Начало проекта

Перед планированием проекта следует:

- установить цели и проблемную область проекта;
- обсудить альтернативные решения;
- выявить технические и управленческие ограничения.

Измерения, меры и метрики

Измерения помогают понять как процесс разработки продукта, так и сам продукт. Измерения процесса производятся в целях его улучшения, измерения продукта — для повышения его качества. В результате измерения определяется *мера* — количественная характеристика какого-либо свойства объекта. Путем непосредственных измерений могут определяться только опорные свойства объекта. Все остальные свойства оцениваются в результате вычисления тех или иных функций от значений опорных характеристик. Вычисления этих функций проводятся по формулам, дающим числовые значения и называемым *метриками.*

В *IEEE Standard Glossary of Software Engineering Terms* метрика определена как мера степени обладания свойством, имеющая числовое значение. В программной инженерии

понятия *мера* и *метрика* очень часто рассматривают как синонимы.

Процесс оценки

При планировании программного проекта надо оценить людские ресурсы (в человеко-месяцах), продолжительность (в календарных датах), стоимость (в тысячах долларов). Обычно исходят из прошлого опыта. Если новый проект по размеру и функциям похож на предыдущий проект, вполне вероятно, что потребуются такие же ресурсы, время и деньги.

Анализ риска

На этой стадии исследуется область неопределенности, имеющаяся в наличии перед созданием программного продукта. Анализируется ее влияние на проект. Нет ли скрытых от внимания трудных технических проблем? Не станут ли изменения, проявившиеся в ходе проектирования, причиной недопустимого отставания по срокам? В результате принимается решение — выполнять проект или нет.

Планирование

Определяется набор проектных задач. Устанавливаются связи между задачами, оценивается сложность каждой задачи. Определяются людские и другие ресурсы. Создается сетевой график задач, проводится его временная разметка.

Трассировка и контроль

Каждая задача, помеченная в плане, отслеживается руководителем проекта. При отставании в решении задачи применяются утилиты повторного планирования. С помощью утилит определяется влияние этого отставания на промежуточную веху и общее время конструирования. Под вехой понимается временная метка, к которой привязано подведение промежуточных итогов.

В результате повторного планирования:

- могут быть перераспределены ресурсы;
- могут быть реорганизованы задачи;
- могут быть пересмотрены выходные обязательства.

Планирование проектных задач

Основной задачей при планировании является определение WBS — Work Breakdown Structure (структуры распределения работ). Она составляется с помощью утилиты планирования проекта. Типовая WBS приведена на рис. 2.2.

Первыми выполняемыми задачами являются системный анализ и анализ требований. Они закладывают фундамент для последующих параллельных задач.

Системный анализ проводится с целью:

1) выяснения потребностей заказчика;
2) оценки выполнимости системы;
3) выполнения экономического и технического анализа;
4) распределения функций по элементам компьютерной системы (аппаратуре, программам, людям, базам данных и т. д.);
5) определения стоимости и ограничений планирования;
6) создания системной спецификации.

В *системной спецификации* описываются функции, характеристики системы, ограничения разработки, входная и выходная информация.

Анализ требований дает возможность:

1) определить функции и характеристики программного продукта;

2) обозначить интерфейс продукта с другими системными элементами;
3) определить проектные ограничения программного продукта;
4) построить модели: процесса, данных, режимов функционирования продукта;
5) создать такие формы представления информации и функций системы, которые можно использовать в ходе проектирования.

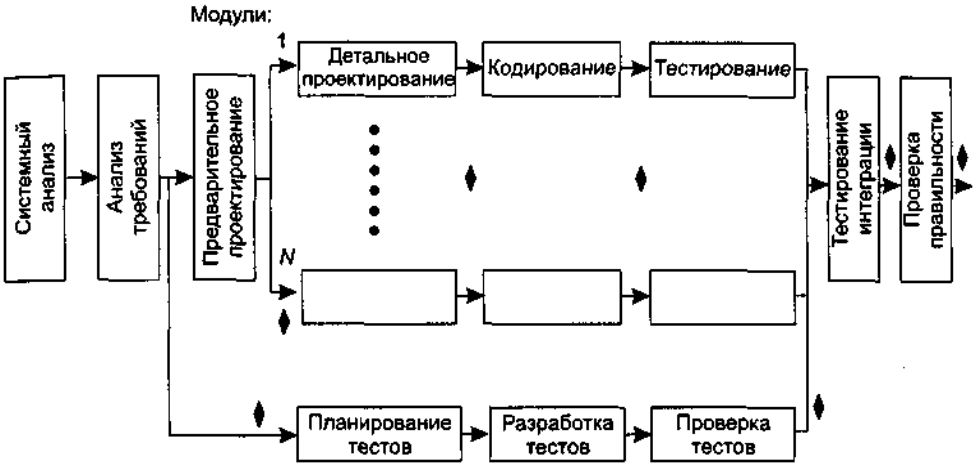

Рис. 2.2. Типовая структура распределения проектных работ

Результаты анализа сводятся в *спецификацию требований* к программному продукту.

Как видно из типовой структуры, задачи по проектированию и планированию тестов могут быть распараллелены. Благодаря модульной природе ПО для каждого модуля можно предусмотреть параллельный путь для детального (процедурного) проектирования, кодирования и тестирования. После получения всех модулей ПО решается задача тестирования интеграции — объединения элементов в единое целое. Далее проводится тестирование правильности, которое обеспечивает проверку соответствия ПО требованиям заказчика.

Ромбиками на рис. 2.2 обозначены вехи — процедуры контроля промежуточных результатов. Очень важно, чтобы вехи были расставлены через регулярные интервалы (вдоль всего процесса разработки ПО). Это даст руководителю возможность регулярно получать информацию о текущем положении дел. Вехи распространяются и на документацию как на один из результатов успешного решения задачи.

Параллельность действий повышает требования к планированию. Так как параллельные задачи выполняются асинхронно, планировщик должен определить межзадачные зависимости. Это гарантирует «непрерывность движения к объединению». Кроме того, руководитель проекта должен знать задачи, лежащие на критическом пути. Для того чтобы весь проект был выполнен в срок, необходимо выполнять в срок все критические задачи.

Основной рычаг в планирующих методах — вычисление границ времени выполнения задачи.

Обычно используют следующие оценки:

1. Раннее время начала решения задачи $T_{\min}^{in}$ (при условии, что все предыдущие задачи решены в кратчайшее время).

2. Позднее время начала решения задачи $T_{\max}^{in}$ (еще не вызывает общую задержку проекта).

3. Раннее время конца решения задачи $T_{\min}^{out}$.

$$T_{\min}^{out} + T_{\min}^{in} + T_{реш}.$$

4. Позднее время конца решения задачи $T_{\max}^{out}$.

$$T_{\max}^{out} + T_{\max}^{in} + T_{реш}.$$

5. Общий резерв — количество избытков и потерь планирования задач во времени, *не* приводящих к увеличению длительности критического пути $T_{к.п}$.

Все эти значения позволяют руководителю (планировщику) количественно оценить успех

в планировании, выполнении задач.

Рекомендуемое правило распределения затрат проекта — 40-20-40:

- на анализ и проектирование приходится 40% затрат (из них на планирование и системный анализ — 5%);
- на кодирование — 20%;
- на тестирование и отладку — 40%.

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 (табл. 2.1).

Таблица 2.1. Исходные данные для расчета LOC-метрик

Проект	Затраты, чел.-мес	Стоимость, тыс. $	KLOC, тыс. LOC	Прогр. док-ты, страниц	Ошибки	Люд
aaa01	24	168	12,1	365	29	3
bbb02	62	440	27,2	1224	86	5
ccc03	43	314	20,2	1050	64	6

Таблица содержит данные о проектах за последние несколько лет. Например, запись о проекте aaa01 показывает: 12 100 строк программы были разработаны за 24 человеко-месяца и стоили $168 000. Кроме того, по проекту aaa01 было разработано 365 страниц документации, а в течение первого года эксплуатации было зарегистрировано 29 ошибок. Разрабатывали проект aaa01 три человека.

На основе таблицы вычисляются размерно-ориентированные метрики производительности и качества (для каждого проекта):

$$\text{Производительность} = \frac{\text{Длина}}{\text{Затраты}}\left[\frac{\text{тыс.LOC}}{\text{чел.-мес}}\right];$$

$$\text{Качество} = \frac{\text{Ошибки}}{\text{Длина}}\left[\frac{\text{Единиц}}{\text{тыс.LOC}}\right];$$

$$\text{Удельная стоимость} = \frac{\text{Стоимость}}{\text{Длина}}\left[\frac{\text{Тыс\$}}{\text{LOC}}\right];$$

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

Достоинства размерно-ориентированных метрик:

1) широко распространены;
2) просты и легко вычисляются.

Недостатки размерно-ориентированных метрик:

1) зависимы от языка программирования;
2) требуют исходных данных, которые трудно получить на начальной стадии проекта;
3) не приспособлены к непроцедурным языкам программирования.

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.

Тип элемента данных — уникальное не рекурсивное (неповторяемое) поле, распознаваемое пользователем. В качестве примера рассмотрим табл. 2.2.

В этой таблице 10 элементов данных: День, Хиты, % от Сумма хитов, Сеансы

пользователя, Сумма хитов (по рабочим дням), % от Сумма хитов (по рабочим дням), Сумма сеансов пользователя (по рабочим дням), Сумма хитов (по выходным дням), % от Сумма хитов (по выходным дням), Сумма сеансов пользователя (по выходным дням). Отметим, что поля День, Хиты, % от Сумма хитов, Сеансы пользователя имеют рекурсивные данные, которые в расчете не учитываются.

Таблица 2.2. Пример для расчета элементов данных

Уровень активности дня недели			
День	**Хиты**	**% от Сумма хитов**	**Сеансы пользоват**
Понедельник	1887	16,41	201
Вторник	1547	13,45	177
Среда	1975	17,17	195
Четверг	1591	13,83	191
Пятница	2209	19,21	200
Суббота	1286	11,18	121
Воскресенье	1004	8,73	111
Сумма по рабочим дням	9209	80,08	964
Сумма по выходным дням	2290	19,91	232

Примеры элементов данных для различных характеристик приведены в табл. 2.3, а табл. 2.4 содержит правила учета элементов данных из графического интерфейса пользователя (GUI).

Таблица 2.3. 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, загол столбцов, которые читаются из внутреннего файла
Внешние Запросы	Вводимые элементы: поле, используемое для поиска, щелчок мыши. Вывод элементы — отображаемые на экране поля

Таблица 2.4. Правила учета элементов данных из графического интерфейса пользователя

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

Обычно одному экрану GUI соответствует несколько транзакций. Типичный экран включает несколько внешних запросов, сопровождающих внешний ввод.

Обсудим порядок учета сообщений. В приложении с GUI генерируются 3 типа сообщений: сообщения об ошибке, сообщения подтверждения и сообщения уведомления. Сообщения об ошибке (например, Требуется пароль) и сообщения подтверждения

(например, Вы действительно хотите удалить клиента?) указывают, что произошла ошибка или что процесс может быть завершен. Эти сообщения не образуют самостоятельного процесса, они являются частью другого процесса, то есть считаются элементом данных соответствующей транзакции.

С другой стороны, уведомление является независимым элементарным процессом. Например, при попытке получить из банкомата сумму денег, превышающую их количество на счете, генерируется сообщение Не хватает средств для завершения транзакции. Оно является результатом чтения информации из файла счета и формирования заключения. Сообщение уведомления рассматривается как внешний вывод.

Данные для определения ранга и оценки сложности транзакций и файлов приведены в табл. 2.5-2.9 (числовая оценка указана в круглых скобках). Использовать их очень просто. Например, внешнему вводу, который ссылается на 2 файла и имеет 7 элементов данных, по табл. 2.5 назначается средний ранг и оценка сложности 4.

Таблица 2.5. 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 2.6. 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 2.7. 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 2.8. 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 2.9. 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)

Отметим, что если во внешнем запросе ссылка на файл используется как на этапе ввода, так и на этапе вывода, она учитывается только один раз. Такое же правило распространяется и на элемент данных (однократный учет).

После сбора всей необходимой информации приступают к расчету метрики — *количества функциональных указателей FP* (Function Points). Автором этой метрики является А. Албрехт (1979) [7].

Исходные данные для расчета сводятся в табл. 2.10.

Таблица 2.10. Исходные данные для расчета FP-метрик

Имя характеристики	**Ранг, сложность, количество**			
	Низкий	**Средний**	**Высокий**	**Итого**
Внешние вводы	0x3 = __	0x4 = __	0x6 = __	= 0
Внешние выводы	0x4 = __	0x5 = __	0x7 = __	= 0
Внешние запросы	0x3 = __	0x4 = __	0x6 = __	= 0
Внутренние логические файлы	0x7 = __	0x 10= __	0x15 = __	= 0
Внешние интерфейсные файлы	0x5 = __	0x7 = __	0x10 = __	= 0
	Общее количество			= 0

В таблицу заносится количественное значение характеристики каждого вида (по всем уровням сложности). Места подстановки значений отмечены прямоугольниками (прямоугольник играет роль метки-заполнителя). Количественные значения характеристик умножаются на числовые оценки сложности. Полученные в каждой строке значения суммируются, давая полное значение для данной характеристики. Эти полные значения затем суммируются по вертикали, формируя общее количество.

Количество функциональных указателей вычисляется по формуле

$$FP = \text{Общее количество} \times (0{,}65 + 0{,}01 \times \sum_{i=1}^{14} F_i),$$

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 (табл. 2.11).

Таблица 2.11. Определение системных параметров приложения

№	**Системный параметр**	**Описание**
1	Передачи данных	Сколько средств связи требуется для передачи или обмена информац 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 которс 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 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 организаци
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.:

$$\text{Производитель} = \frac{\text{ФункцУказатель}}{\text{Затраты}}\left[\frac{\text{FP}}{\text{чел.}-\text{мес}}\right];$$

$$\text{Качество} = \frac{\text{Ошибки}}{\text{ФункцУказатель}}\left[\frac{\text{Единиц}}{\text{FP}}\right];$$

$$\text{Удельная стоимость} = \frac{\text{Стоимость}}{\text{ФункцУказатель}}\left[\frac{\text{Тыс.\$}}{\text{FP}}\right];$$

$$\text{Документированность} = \frac{\text{СтраницДокумента}}{\text{ФункцУказатель}}\left[\frac{\text{Страниц}}{\text{FP}}\right].$$

Область применения метода функциональных указателей — коммерческие информационные системы. Для продуктов с высокой алгоритмической сложностью используются метрики *указателей свойств* (Features Points). Они применимы к системному и инженерному ПО, ПО реального времени и встроенному ПО.

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 табл. 2.12.

Таблица 2.12. Исходные данные для расчета указателя свойств

№	Характеристика	Количество	Сложность	Итого
1	Вводы	0	x4	= 0
2	Выводы	0	x5	= 0
3	Запросы	0	x4	= 0
4	Логические файлы	0	x7	= 0
5	Интерфейсные файлы	0	x7	= 0
6	Количество алгоритмов	0	x3	= 0
		Общее количество		= 0

После заполнения таблицы по формуле (2.1) вычисляется значение указателя свойств. Для сложных систем реального времени это значение на 25-30% больше значения, вычисляемого по таблице для количества функциональных указателей.

Достоинства функционально-ориентированных метрик:

1. Не зависят от языка программирования.
2. Легко вычисляются на любой стадии проекта.

Недостаток функционально-ориентированных метрик: результаты основаны на субъективных данных, используются не прямые, а косвенные измерения. FP-оценки легко пересчитать в LOC-оценки. Как показано в табл. 2.13, результаты пересчета зависят от языка программирования, используемого для реализации ПО.

Таблица 2.13. 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 в ходе руководства проектом

Процесс руководства программным проектом начинается с множества действий, объединяемых общим названием *планирование проекта.* Первое из этих действий — выполнение оценки. Оно закладывает фундамент для других действий по планированию проекта. При оценке проекта чрезвычайно высока цена ошибок. Очень важно провести оценку с минимальным риском.

Выполнение оценки проекта на основе LOC- и FP-метрик

Цель этой деятельности — сформировать предварительные оценки, которые позволят:

- предъявить заказчику корректные требования по стоимости и затратам на разработку программного продукта;
- составить план программного проекта.

При выполнении оценки возможны два варианта использования LOC- и FP-данных:

- в качестве оценочных переменных, определяющих размер каждого элемента продукта;
- в качестве метрик, собранных за прошлые проекты и входящих в метрический базис фирмы.

Обсудим шаги процесса оценки.

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

$$f_1, f_2, \ldots, f_n.$$

- *Шаг 2.* Для каждой функции f_i, планировщик формирует лучшую $LOC_{лучшi}$ ($FP_{лучшi}$), худшую $LOC_{худшi}$ ($FP_{худшi}$) и вероятную оценку $LOC_{вероятнi}$ ($FP_{вероятнi}$). 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:

$$LOC_{ожi} = (LOC_{лучшi} + LOC_{худшi} + 4x\ LOC_{вероятнi})/ 6.$$

- *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:

$$ПРОИЗВ_i = ПРОИЗВ_{ср} x (LOC_{ср} / LOC_{ожi}),$$

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:

$$ПРОИЗВ_i = ПРОИЗВ_{анi} x (LOC_{анi} / LOC_{ожi}).$$

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

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

для второго и третьего подходов

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

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

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

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

для третьего подхода

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

где $УД\_СТОИМОСТЬ_{анi}$ — метрика стоимости одной строки аналога, взятая из метрического базиса. Пример применения данного процесса оценки приведем ниже.