ГЛАВА 1

Лабиринты тестирования

Введение

Не все проекты тестирования программного обеспечения начинаются с мажорной ноты. Кому-то из участников тестирования, возможно, не хватает практики, или из опытного программиста наивный менеджер собирается сделать хорошего тестера (в особенности если эта работа не поручена никому конкретно). Как и другие технические дисциплины, тестирование программного обеспечения — это специальная область, требующая специфических навыков. Многие менеджеры ошибочно полагают, что программист может эффективно тестировать программное обеспечение без должной подготовки или наставлений.

Представьте себе человека, которому предстоит проделать большую работу по тестированию, и который не знает, с чего начать, более того, ему не к кому обратиться за помощью. Подобно путешественнику, который пробирается через лабиринт, начинающий тестер знает, что где-то должен быть конец пути, но не знает, как его достичь. К несчастью для тестера продукт выпускается в следующую пятницу, а документация проекта неполна, неточна, или, что еще хуже, ее вообще не существует.

Это, конечно, лишь преувеличение. Хорошая среда разработки программного обеспечения предполагает адекватное кадровое обеспечение и соответствующее время на такую деятельность, как тестирование программного обеспечения, а также руководство персоналом, который им занимается. Однако условия работы некоторых тестеров никак нельзя назвать идеальными — зачастую необходимо изучить и протестировать новое приложение в нереальные сроки. Некоторые организации решают выпустить продукт к определенной дате и хотят, чтобы тестеры подтвердили пригодность продукта к использованию. Это *далеко не самый лучший* способ тестирования, но у тестеров просто нет выбора, разве что стряхнуть пыль с резюме и попытаться подыскать работу получше.

Эта глава служит руководством по прохождению ряда этапов тестирования и представляет тесты для демонстрации правильной работы приложения. Однако сначала рассмотрим тестируемые приложения.

Пример приложения

В этой главе используется пример налогового калькулятора на 2000-й налоговый год, который был создан после того, как Налоговое управление США ввело федеральный подоходный налог №1040. Данный пример иллюстрирует цели тестирования, а вовсе не описывает попытку реализации всего налогового кодекса. Читатели за пределами США могут легко представить себя в роли тестеров, которые не знакомы с приложением.

Пользователь вводит финансовую информацию, а налоговый калькулятор возвращает значение, представляющее собой величину налога, взимаемого с налогоплательщика. Приложение содержит много окон для ввода данных, четыре из которых показаны ниже. В выходном окне перечисляются результаты. На следующих рисунках показаны примеры моментальных снимков экрана.

Рис. 1.1 – информация о налогоплательщике и его статусе.

Рис. 1.2 – информация о доходах.

Рис. 1.3 – информация о статьях отчислений.

Рис. 1.4 – другие налоги и кредиты, касающиеся налогоплательщика.

Рис. 1.5 – выходные данные налогового калькулятора.

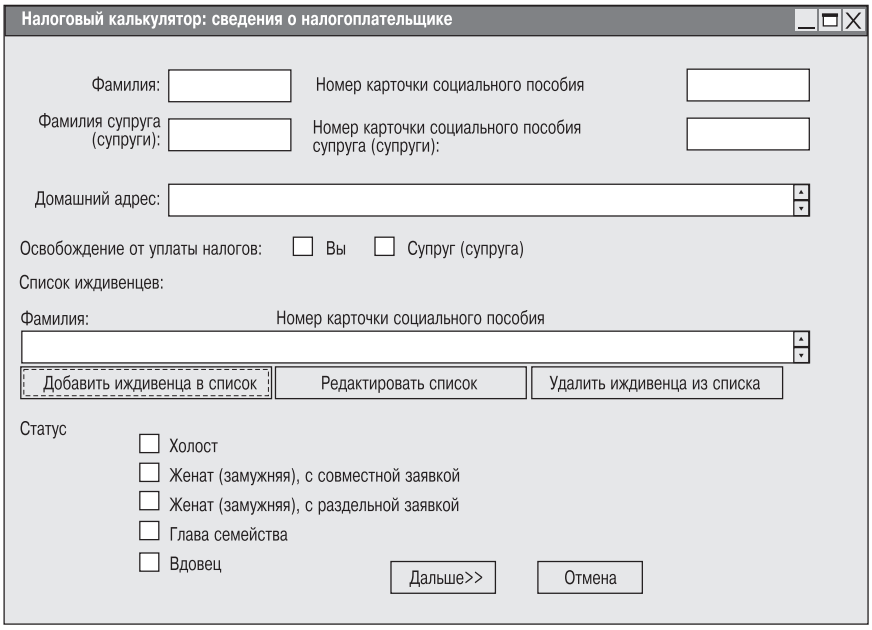

Рис. 1.1. Налоговый калькулятор. Информация о налогоплательщике

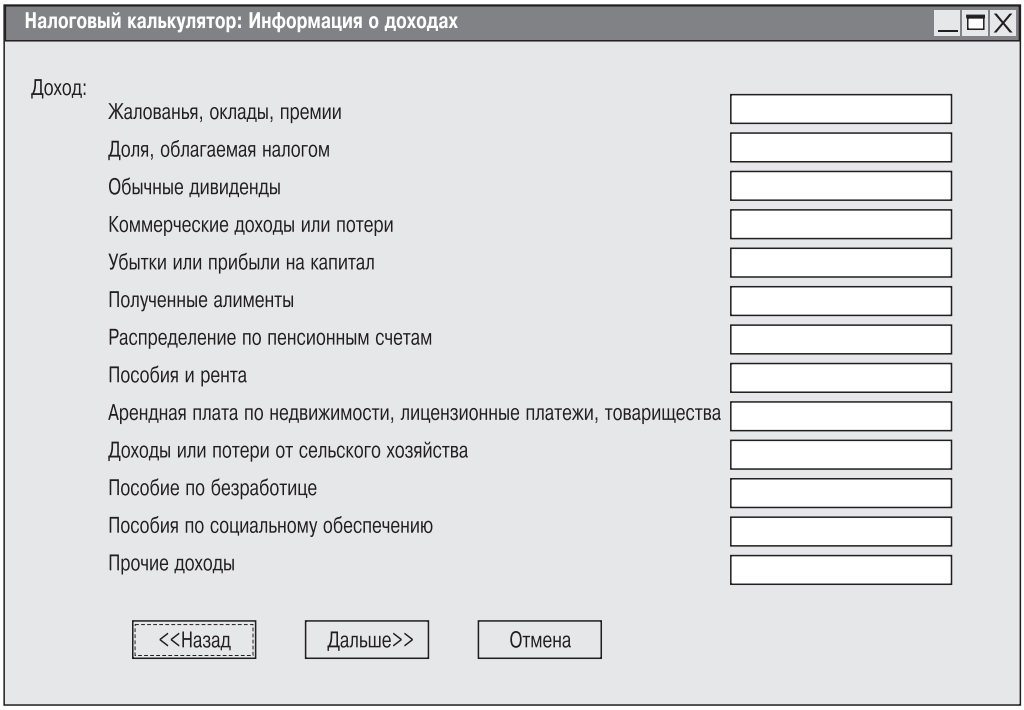

Рис. 1.2. Налоговый калькулятор. Информация о доходах

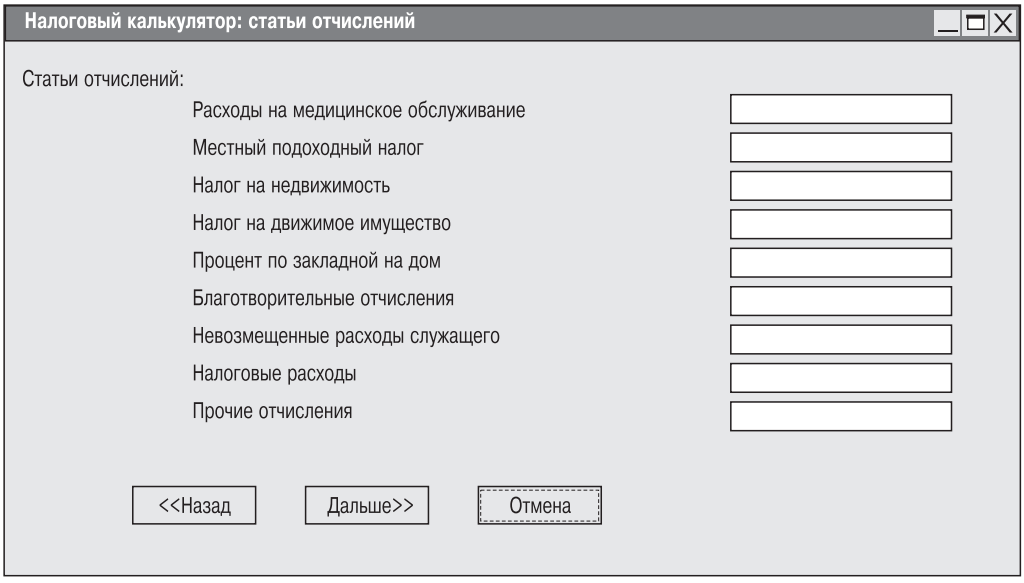

Рис. 1.3. Налоговый калькулятор. Информация о статьях отчислений

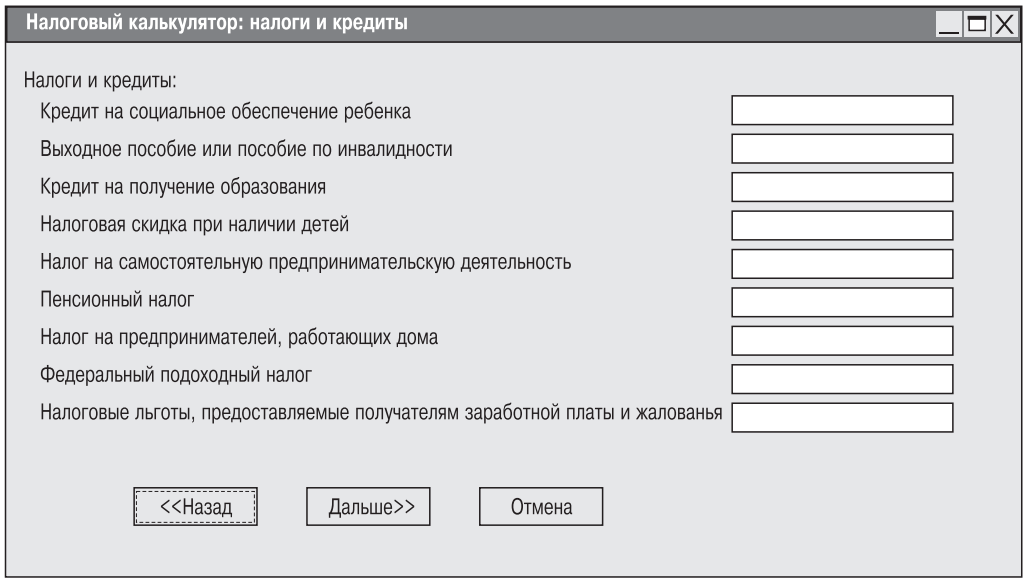

Рис. 1.4. Налоговый калькулятор. Информация о налогах и кредитах

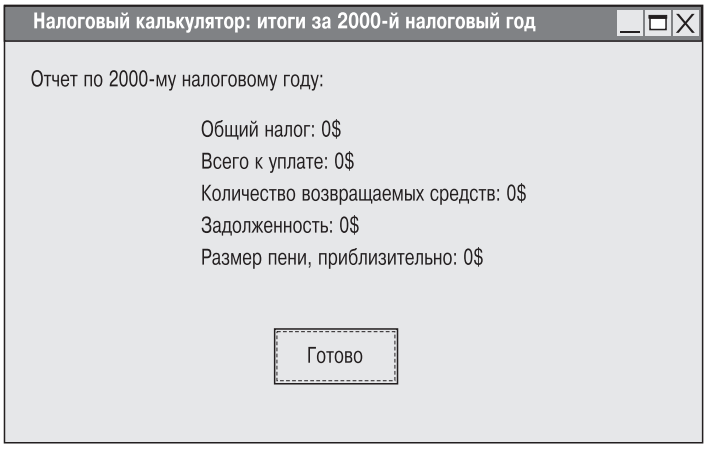

Рис. 1.5. Налоговый калькулятор. Вычисленные результаты

Задача тестеров заключается в том, чтобы убедиться, что приложение “Налоговый калькулятор” готово к выпуску. В этот момент они еще не знают, как пользоваться данным приложением. Что касается документации, то федеральное правительство выпускает огромный документ, описывающий налоговое законодательство, который объясняет, как вычислять результирующий налог, но отнюдь не помогает понять суть работы программного продукта. При тестировании иных приложений или в других подобных ситуациях адекватной информации, касающейся проекта, может оказаться

очень мало, а в некоторых случаях письменное изложение того, что должна делать система, вообще отсутствует.

Как правило, встречаются следующие трудности.

- Большое и сложное приложение.
- Ограниченная осведомленность о приложении или же ее полное отсутствие.
- Неадекватная пользовательская документация.
- Слабые знания о тестировании программного обеспечения.
- Нет доступа к справке или руководству по тестированию программного обеспечения.
- Нереальные сроки завершения тестирования.

Наращиваемый подход в тестировании

Для эффективной работы тестер программного обеспечения должен:

1. знать метод тестирования программного обеспечения;
2. знать тестируемое приложение.

В каждом новом тестировании тестер должен посвятить какое-то время изучению приложения. Неопытный тестер должен также изучить методы тестирования, включая общие концепции тестирования и способ определения тестовых примеров. Эта глава посвящена и тому, и другому. Основная задача заключается в составлении подходящего набора тестов, которые можно провести в сжатые сроки.

Чтобы уложиться в эти сроки, тестирование приложения "Налоговый калькулятор" будет разделено на восемь последовательных стадий. На первых стадиях путем оценки точности результатов определяется, работает ли приложение в нормальных условиях. На последних стадиях предпринята попытка вывести систему из строя. В остальной части этой главы данные стадии обсуждаются более детально.

Стадия 1:	Изучение
Цель:	Ознакомиться с приложением.
Стадия 2:	Базовый тест
Цель:	Разработать и реализовать простой тестовый пример.
Стадия 3:	Анализ тенденций
Цель:	Определить, работает ли приложение так, как было задумано, когда еще нельзя предварительно оценить реальные результаты.
Стадия 4:	Инвентаризация
Цель:	Определить различные категории данных и создать тесты для каждого элемента категории.

Стадия 5:	Комбинирование элементов инвентарных списков
Цель:	Скомбинировать различные входные данные.
Стадия 6:	Граничные оценки
Цель:	Оценить поведение приложения при граничных значениях данных.
Стадия 7:	Ошибочные данные
Цель:	Оценить отклик системы на ввод неправильных данных.
Стадия 8:	Создание напряжений
Цель:	Попытаться вывести систему из строя.

Сроки очень короткие, так что возможности пройти через все стадии не будет. Количество стадий тестирования, которое может провести один специалист, определяется затраченным на тестирование временем исходя из запланированной даты выпуска. После выполнения базового теста, если есть свободные тестеры, стадии можно проходить одновременно.

Опираясь на идеи, рассматриваемые в этой главе, разбиение на такие стадии, как инвентаризация, граничные оценки, ошибочные данные, создание напряжений в среде без труда можно применять к любым проектам тестирования и эти концепции будут снова возникать в последующих главах.

Для иллюстрации различных техник тестирования представлен образец тестового примера только для одного поля, или области ввода данных. При определении тестов для дополнительных полей нужно действовать аналогичным образом. При тестировании реальных приложений, для определения и реализации полного набора тестовых примеров может оказаться недостаточно времени или других ресурсов. В такой ситуации нужно определить, какие тесты наиболее важны и какие функции не будут тестироваться. Анализ степени риска поможет сконцентрировать усилия при тестировании на наиболее важных функциях приложения и выполнить наиболее уместные тесты. В главе 8 представлена техника анализа степени риска, которую можно использовать уже до начала составления тестовых примеров.

Тестеры, которым настолько “повезло”, что их пригласили в проект, в котором предоставляется достаточно времени для надлежащего тестирования, могут сделать следующее.

- Распланировать задачи тестирования.
- Спроектировать тесты, используя методы, описанные в последующих главах.
- Создать подходящую для проведения теста среду тестирования.
- Оценить, приобрести и установить инструменты автоматического тестирования.
- Написать соответствующую документацию, связанную с тестированием.

Подключение тестера к проекту на очень поздней стадии, в особенности, когда сроки завершения проекта уже близки, не позволяет спланировать тестирование надлежащим образом, и ограничивает возможности приобретения полезных инструментов.

В таких условиях нужно сконцентрироваться на разработке серии тестов, чтобы определить, правильно ли функционирует приложение.

Теперь можно начать тестирование приложения "Налоговый калькулятор".

Стадия 1. Изучение

Прежде чем тестер сможет выполнить какой-либо тест, он должен ознакомиться с приложением. Такое изучение может принимать различные формы:

- ознакомления с любыми доступными учебными пособиями и материалами;
- чтения любой доступной документации — технической или предназначенной для конечного пользователя;
- привлечения специалиста, который может продемонстрировать работу приложения;
- вписывания случайных данных и команд;
- проверки всех возможностей приложения;
- принятия скептической позиции ("Только посмотрите, что оно вытворяет!").

В ходе изучения тестеры узнают, как работает приложение, наблюдая за его поведением. Они также начинают чувствовать, что из себя представляют верные и неверные входные данные. Если приложение выдает неожиданный результат, это может свидетельствовать о потенциальных проблемах.

Изучение приложения и ознакомление с его функциями — это необходимый этап обучения. Тестеры создают тесты на ходу, часто находясь под влиянием предыдущих тестов. Этот подход, известный как *исследовательское тестирование*, — первый этап в процессе принятия решения о том, что будет тестироваться. Очевидно, что гораздо эффективнее работать в более формальной среде тестирования, в которой тесты определяются заблаговременно. Изучение, хотя и является преимуществом, все же имеет свои недостатки. Во-первых, результаты, выданные приложением, могут повлиять на решение тестера — он может поверить, что полученные результаты верны. Во-вторых, когда тестер сталкивается с ошибкой, может оказаться, что для определения верного пути в приложении или предыдущего состояния нет записи последовательности нажатых перед этим кнопок. Это может усложнить воспроизведение ошибки.

Задача изучения заключается в том, чтобы как можно больше узнать о приложении, используя его функции. В главе освещается лишь один аспект *исследовательского тестирования*. Всякий раз, когда тестер создает новый тест, основываясь на своих наблюдениях, полученных в ходе изучения текущего теста, он применяет метод *исследовательского тестирования* (чтобы лучше ознакомиться с этим методом, можно обратиться к работам [Bach] и [Kaner 99]).

Еще одна важная задача заключается в поиске квалифицированного специалиста, или *авторитетного лица проекта*, который бы хорошо знал приложение и мог разъяснить требования, ведь в ходе тестирования возникает множество вопросов. Любые решения, принятые авторитетным лицом проекта, или данные им разъяснения *необходимо* регистрировать и отмечать.

Стадия 2. Базовое тестирование

После изучения основных функций приложения тестер создает базовый тестовый пример. Обычно базовый тест несложен: он основывается на простейшем пути в приложении, в процессе которого используются установки по умолчанию или же другие непосредственные входные данные. При заданных входных данных тестер должен определить результирующие данные.

Результирующие данные состоят из любых наблюдаемых выходных данных (или отсутствия таковых), которые были получены в результате тестирования. Следует отметить, что для некоторых тестов ожидается, что они не будут вносить никаких изменений в приложение, тогда результат, если он совпадает с ожидаемым, оказывается верным. (Чтобы подробнее ознакомиться с различием между выходными и результирующими данными, см. [Beizer 90] и [Beizer 95].) В зависимости от приложения существует несколько источников определения ожидаемых результирующих данных.

1. Авторитетное лицо проекта или другой специалист по этому вопросу (ведущий программист, проектировщик или менеджер проекта) будет знать, как определить результирующие данные. В примере с налоговым калькулятором помочь может также бухгалтер или налоговый поверенный.
2. Пример пользовательского сценария может содержаться в пользовательской документации (если, конечно, она существует).
3. Необходимую информацию можно получить из документа, содержащего требования (если он существует).
4. Важные моменты могут содержаться и в другой, имеющей отношение к делу, документации. В примере с налоговым калькулятором инструкции для налогоплательщика содержат указания по вычислению результатов вручную.
5. Охарактеризовать ожидаемый результат может конечный пользователь.

В некоторых приложениях авторитетное лицо проекта определяет допустимый предел погрешности. Например, некоторые вычисления могут варьироваться за счет различных округлений. При введении данных Налоговое управление США позволяет налогоплательщику округлять текущие значения до ближайшего целого значения, вместо того чтобы сохранять центы в ходе вычислений и округлять конечный результат. В этом примере предполагается, что вследствие округления результаты могут отличаться на ±1$.

Определяя базовый тест путем перечисления входных данных и ожидаемых результатов, тестеры должны также подготовиться к *проведению* этого теста. Необходимо создать среду тестирования, чтобы запустить приложение с тестовыми данными. Создание среды тестирования состоит из одного или нескольких перечисленных ниже этапов.

- Установка приложения.
- Установка и программирование инструментов автоматизации тестирования, если в этом есть необходимость.

- Настройка специальных файлов, включая занесение в них данных, необходимых для тестирования.
- Создание утилит, которые поддерживают связь с приложением.
- Приобретение подходящего аппаратного обеспечения и необходимого оборудования.

Если время ограничено и у тестеров нет возможности выбрать, приобрести, установить, изучить и запрограммировать инструмент тестирования, они должны проводить тест вручную.

Во время проведения теста реальные результаты сравнивают с ожидаемыми, чтобы определить, пройден тест или нет. Проведение базового теста — это важное достижение, и на то есть несколько причин:

- тестеры исследовали приложение и теперь знают о нем достаточно, чтобы создать настоящий тест;
- для выполнения теста тестерам пришлось создать среду тестирования;
- этот базовый тест становится основой для создания последующих тестовых примеров;
- приложение не смогло адекватно пройти этот тест, так как тестеры обнаружили серьезную проблему.

Приложение, которое не смогло адекватно пройти базовый тест, не пригодно к использованию. Разработчики должны разобраться с проблемой и создать более надежное приложение, однако тестеры могут воспользоваться преимуществами сложившегося положения, чтобы определить и провести дополнительные тесты, перед тем как в их распоряжении появится новая версия приложения, пригодная для тестирования.

В примере с налоговым калькулятором базовый тест подразумевает задание самого простого типа налогоплательщиков с несложными финансовыми или деловыми операциями. Для данных в базовом тесте были подобраны следующие атрибуты.

- Одинокий человек, не имеющий материально зависимых лиц.
- Одна работа с оплатой $40 000 в год.
- Дополнительных доходов не имеет.
- Отчислений нет.

Исходя из приведенной выше информации следует ожидать, что налог с этого человека в 2000-м году составит $5 779 без учета налогового штрафа, который взимается при исключении налогового отказа.

Для отслеживания входных данных и ожидаемых результатов (или результирующих данных) используется табл. 1.1. Проводя тест, в эту простую таблицу можно записывать реальные результаты. В главе 4 показано множество форматов таблиц.

Таблица 1.1. Базовый тестовый пример

ID тестового примера	Статус	Заработок, ($)	Величина отчислений, ($)	Ожидаемые результаты: причитающийся налог, ($)	Реальный результат
налог 1	одинокий (-ая)	40 000	0	5 779	

Стадия 3. Анализ тенденций

Анализ тенденций — это необязательная стадия, которая полезна тогда, когда выполняется любое из следующих условий.

- Знакомство с приложением весьма поверхностно.
- Входные и выходные данные содержат численные значения.
- Неизвестны граничные значения данных.
- Сложно вычислить ожидаемые результаты.
- Имеется предположение относительно диапазона ожидаемых результатов.

В идеале, хотелось бы точно определить результирующие данные, минуя таким образом анализ тенденций и переходя к стадии 4. Поскольку известно, как вычислить ожидаемый налог при различных входных данных, эту стадию нужно было бы пропустить, однако для иллюстрации, к приложению "Налоговый калькулятор" все же будет применен анализ тенденций.

Характеризуя обычного налогоплательщика, базовый тестовый пример становится основой для анализа тенденций. На стадии анализа тенденций проводится отслеживание характера поведения путем изменения значения одной определенной численной переменной. Даже если результат реальных выходных данных точно не известен, можно оценить, изменяются ли значения выходных данных в ожидаемом направлении. Например, если постепенно увеличивать доход налогоплательщика, то, очевидно, что суммарный налог будет возрастать пропорционально. Если в выходных данных будут наблюдаться значительные скачки по сравнению с другими данными, то проблема нуждается в дальнейшем исследовании. Такая стратегия является частью *исследовательского тестирования*, поскольку перед началом тестирования не определялись ожидаемые результаты. Однако этот подход все же дает информацию о предполагаемом поведении приложения.

В табл. 1.2 отслеживаются тенденции при возрастании заработка. Прежде всего, определяются входные данные и ожидаемые тенденции, эта информация заносится в таблицу, после чего вводятся входные данные, записываются реальные выходные данные и оцениваются результаты. В этом примере заработок постепенно увеличивается на $3 000, ожидаемый результирующий налог возрастает монотонно. Похоже, что до сих пор налоговый калькулятор работал верно, но что будет, если одно выходное значение окажется нехарактерным в сравнении с остальными значениями в серии? Будет ли считаться, что найдена проблема, если тестовый пример "тенденция 5"

покажет в результате $10 500? Не обязательно. Ответ на этот вопрос даст авторитетное лицо проекта. Возможно, возросший заработок переместил налогоплательщика в более высокую налоговую категорию, увеличивая таким образом суммарный налог, или, может быть, приложение сгенерировало неверное значение.

Таблица 1.2. Базовый тестовый пример

ID тестового примера	Статус	Заработок, ($)	Величина отчислений, ($)	Ожидаемые результаты: причитающийся налог, ($)	Реальный результат
налог 1	одинокий (-ая)	40 000	0	5 779	5 779
тенденция 1	одинокий (-ая)	43 000	0	тенденция к возрастанию	6 619
тенденция 2	одинокий (-ая)	46 000	0		7 459
тенденция 3	одинокий (-ая)	49 000	0		8 299
тенденция 4	одинокий (-ая)	52 000	0		9 139
тенденция 5	одинокий (-ая)	55 000	0		9 979
тенденция 6	одинокий (-ая)	58 000	0		10 819
тенденция 7	одинокий (-ая)	61 000	0		11 659
тенденция 8	одинокий (-ая)	64 000	0		12 499

Можно также проанализировать и другие тенденции:

- увеличение числа материально зависимых лиц приводит к снижению налога;
- увеличение дополнительной прибыли приводит к увеличению налога;
- увеличение статей отчислений уменьшает налог (кроме случаев, когда существуют пороговые значения, ограничивающие допустимое число различных отчислений).

Последний случай может служить иллюстрацией законов о налогообложении, которые ограничивают число статей отчислений. Когда тестеры замечают сильные изменения в тенденциях, это не всегда означает наличие проблемы. То, является ли данный эффект результатом пересечения числовой границы или же это проблема в приложении, будет решать авторитетное лицо проекта.

Хотя анализ тенденций действительно показывает, функционирует ли приложение должным образом, эта форма тестирования неэффективна по сравнению с другими методами разработки тестовых примеров. В главе 4 введены методы разбиения на эквивалентные классы и методы анализа граничных значений, которые позволяют проводить эффективное тестирование с широким диапазоном данных.

Стадия 4. Инвентаризация

Стадия инвентаризации состоит из определения типов данных, доступных в приложении с последующей классификацией состояний, в которых может находиться каждый элемент данных. Тестирование позволяет убедиться, что каждое состояние используется по крайней мере один раз. Каждый набор состояний задает инвентарный список.

На данном этапе по-прежнему используется базовый тест и за один раз модифицируется одно значение, чтобы можно было убедиться, что каждый элемент корректно влияет на приложение. Рассмотрим два примера инвентарного списка: *Статус* и *Статьи отчислений*; для этого нужно знать, какие результаты ожидаются для каждого тестируемого состояния.

Инвентарный список *Статус* имеет пять возможных значений, перечисленных ниже. Это взаимоисключающие категории, поскольку за раз используется только одно значение. Каждое значение статуса влияет на вычисление суммарного налога. Применение инвентарного список *Статус* в базовом тесте приводит к созданию четырех дополнительных тестовых примеров, показанных в табл. 1.3. Различные величины, перечисленные в столбце *Величина отчислений*, представляют собой специфические пороговые значения, связанные со статусом. Данные пороговые значения объяснены в этом разделе позже.

Инвентарный список Статус:

- одинокий (-ая);
- женат (замужняя) с независимой декларацией доходов;
- женат (замужняя) с совместной декларацией доходов;
- глава семьи;
- вдовец (вдова).

Заметим, что каждый новый элемент в инвентарном списке влияет на результаты именно так, как было задумано (не принимая во внимание то, логично это состояние или нет). Кажется логичным, что три статуса требуют дополнительной информации. Статус “глава семьи” требует включения нескольких материально зависимых лиц, а оба статуса “женат (замужняя)” требуют информации о супруге. Кроме того, можно вычислить изменение в налоге. В других приложениях состояния, полученные в результате использования отдельных элементов в инвентарном списке, могут казаться необычными. Однако их все-таки нужно протестировать, поскольку пользователь, вероятно, введет те же нелогичные значения. Нужно убедиться в том, что приложение верно обрабатывает данные. Сюда входят любые логические схемы обработки ошибок и сообщения об ошибках.

Таблица 1.3. Тестовый пример, использующий инвентарный список *Статус*

ID тестового примера	Статус	Заработок, ($)	Величина отчислений, ($)	Ожидаемые результаты: причитающийся налог, ($)	Реальный результат
налог 1	одинокий (-ая)	40 000	44 000	5 779	
налог 2	женат (замужняя) с независимой декларацией доходов	40 000	3 675	6 537	
налог 3	женат (замужняя) с совместной декларацией доходов	40 000	7 350	4 061	
налог 4	глава семьи	40 000	6 450	4 616	
налог 5	вдовец (вдова)	40 000	7 350	4 481	

Другую категорию инвентарных списков представляют *статьи отчислений*, которые не подразумевают взаимоисключений, поскольку налогоплательщик может установить любую комбинацию статей отчислений. Несмотря на то, что можно использовать различное число статей отчислений, чтобы убедиться, что каждый элемент инвентарного списка корректно влияет на результирующие данные, поначалу за один раз в базовом тесте нужно использовать одну статью отчислений. В табл. 1.4 были внесены новые столбцы, чтобы отслеживать статьи отчислений и показать эти дополнительные тесты.

Инвентарный список статей отчислений:

- расходы на медицинское обслуживание и стоматологию;
- государственный и местный подоходный налог;
- налог на недвижимость;
- налог на движимое имущество;
- процент по закладной;
- пожертвования благотворительным организациям;
- невозмещенные затраты на содержание наемного работника;
- плата за подготовку налоговых документов;
- прочие отчисления.

Таблица 1.4. Тестовый пример, задействующий инвентарный список статей отчислений

ID тестового примера	Статус	Заработок, ($)	Тип статей отчислений	Величина отчислений, ($)	Ожидаемые результаты: причитающийся налог, ($)	Реальные результаты	Замечание: пределы статей расходов (налоговое законодательство)
налог 6	одинокий (-ая)	40 000	расходы на медицинское обслуживание и стоматологию	44 505	5 779		ограничивается до 7,5% от скорректированного валового дохода; вычету подлежит сумма 1 405$
налог 7	одинокий (-ая)	40 000	государственный и местный подоходный налог	44 505	5 765		применяется полностью
налог 8	одинокий (-ая)	40 000	налог на недвижимость	44 505	5 765		применяется полностью
налог 9	одинокий (-ая)	40000	налог на движимое имущество	44 505	5 765		применяется полностью
налог 10	одинокий (-ая)	40 000	процент по закладной	44 505	5 765		применяется полностью
налог 11	одинокий (-ая)	40 000	пожертвования благотворительным организациям	44 505	5 765		применяется полностью
налог 12	одинокий (-ая)	40 000	не возмещенные затраты на содержание наемного работника	44 505	5 779		ограничивается до 2% от скорректированного валового дохода; вычету подлежит сумма $3 605
налог 13	одинокий (-ая)	40 000	плата за подготовку налоговых документов	44 505	5 779		ограничивается до 2% от скорректированного валового дохода; вычету подлежит сумма $3 605
налог 14	одинокий (-ая)	40 000	прочие разнообразные отчисления	44 505	5 765		применяется полностью

Налоговое законодательство позволяет использовать либо статьи отчислений, либо стандартные отчисления — в зависимости от того, какая величина будет больше. Для одинокого налогоплательщика в рассматриваемом базовом примере стандартное отчисление составляет $4 400. Чтобы использовать статьи отчислений, их сумма должна превышать $4 400. Следовательно, в этом примере для каждого типа статей отчислений будет использоваться сумма, равная $4 405. Однако не все типы статей отчислений создаются равноценно; величину некоторых из них, как указано ниже, ограничивают пороговые значения, зависящие от прибыли.

- Затраты на медицинское обслуживание и стоматологию должны превышать 7,5% скорректированного валового дохода. (Скорректированный валовой доход получается путем суммирования всех заработков и прибылей с последующим вычитанием допустимых затрат.) В базовом примере с заработком $40 000 первые затраты на медицинское обслуживание в размере $3 000 не подлежат вычету. Следовательно, вычету подлежат только $1 405.
- Невозмещенные затраты на содержание наемного работника и на подготовку налога предполагают ограничение в 2% от скорректированного валового дохода. В базовом примере с заработком в $40 000 первые такие затраты в размере $800 не подлежат вычету. Допускается вычет из $3 605.

После применения всех статей отчислений оказалось, что ожидаемый суммарный налог составляет $5 765, принимая во внимание пороговые значения для отчислений (которые в данном примере не превышают предельного значения), приводящие в результате к налогу в размере $5 779. В табл. 1.4 показаны типы статей отчислений, используемые величины, а также введен дополнительный столбец, указывающий на то, влияет ли предел на сумму.

Результатом инвентаризации является таблица контрольных проверок элементов. В примере с налоговым калькулятором имеется значительно большее количество инвентарных списков, которые также требуют тестирования (например, расходы по сделке и источники доходов). Изолируя влияние каждого отдельного элемента инвентарного списка, можно определить, корректно ли обрабатывает его приложение.

На данный момент было определено много тестов путем задания входных значений и ожидаемых результатов. При выполнении каждого теста наблюдались и записывались реальные результаты. Если реальные результаты отличаются от ожидаемых, тест считается проваленным и возникает несоответствие, которое означает наличие проблемы. Единственное, что можно сделать в данной ситуации, — это выдвинуть предположение, что что-то неверно. Причин того, что тест не был пройден, может быть несколько.

1. Неверное описание ожидаемых результатов. Такое случается, если требования неясны, неправильно поняты или если ожидаемые результаты были неверно рассчитаны (этот тип ошибки можно обнаружить при пересмотре тестового примера. Описание пересмотров представлено в главе 9) .
2. Ошибка возникла в ходе выполнения теста. Возможно, тестер ввел неверные входные данные или же неисправно тестовое программное обеспечение (которое может иметь свои собственные дефекты).
3. В приложении действительно есть дефекты.

Что же делать? Необходимо убедиться, что ошибка произошла не из-за ввода тестером неверного значения и что проблема не в тестовом программном обеспечении. Тестеры не обязаны устранять дефекты, они должны составить список несоответствий, а возможность решать предоставляется авторитетному лицу проекта. Причины проблем могут быть следующими.

- Проблема возникла из-за требований: требование было пропущено; оно оказалось двусмысленным, не вполне понятным; о требовании не сообщили тестеру.
- Тест оказался ошибочным из-за того, что неверно были определены ожидаемые результаты.
- Проблема возникла в ходе проведения теста (сложности с инструментальными средствами тестирования или опечатка).
- В программном обеспечении действительно есть дефект.
- Авторитетное лицо не в состоянии повторно воспроизвести неполадку. В данном случае можно предположить, что сложности связаны с аппаратными конфигурациями — они могут оказаться различными у тестера и у того, кто пытается воспроизвести неполадку (возможно, этому специалисту были предоставлены неполные или неверные инструкции либо совсем другие тестовые данные. Возможно также, что он не совсем точно следовал инструкциям).

После того как разработчики устранят проблему и создадут новую версию приложения, тестеры должны повторно провести исходные тесты, запускаемые на предыдущих версиях, чтобы убедиться, что все функции в новом приложении работают так, как было задумано. Когда времени мало или же иные ресурсы ограничены, тестеру нужно решить, какие тесты будут использоваться повторно. В главе 8 предложено несколько методов оценки степени риска и определены наиболее важные тестовые примеры.

В примере с налоговым калькулятором использование определенных тестов зависит от того, что в версиях приложения остается неизменным, поскольку ожидаемые результаты будут изменяться из года в год согласно изменениям в налоговом законодательстве.

Стадия 5. Комбинирование элементов инвентарных списков

Следующий этап тестирования представляет собой комбинирование элементов инвентарных списков, чтобы можно было определить, предсказуема ли их совместная работа. Для некоторых приложений может оказаться выгодным применение наращиваемого подхода, в котором сначала сочетаются два элемента инвентарных списков, а затем три, четыре и т.д. Другой наращиваемый подход заключается в использовании таблиц тестовых примеров, в которых каждый последующий тестовый пример — это модификация предыдущего теста. Таким образом, последний из перечисленных выше тестов содержит все накопленные из предыдущих тестов состояния.

Не все инвентарные списки содержат простые значения данных или состояния. Рассмотрим различные приложения, которые обрабатывают данные из файлов и руководят управляемыми событиями прерываниями и для которых основной сложностью

является пропускная способность. Наращиваемый подход состоит из следующих типов тестов.

- Индивидуальная обработка каждого файла.
- Обработка одного файла и генерация одного прерывания.
- Обработка двух файлов.
- Обработка двух файлов и генерация одного прерывания.
- Обработка двух файлов и генерация двух прерываний.
- И т. д.

При определении всех комбинаций получается огромное число тестовых примеров — значительно больше, чем можно было бы выполнить в разумные сроки. Ключом к разрешению ситуации служит анализ степени риска, определяющий наиболее важные функции приложения с последующим заданием схемы комбинаций, которая наилучшим образом описывает отношения с высоким приоритетом.

В рассматриваемом примере налоговый калькулятор — это управляемое данными приложение, каждый элемент в его инвентарном списке ранее был протестирован изолированно, так что использование наращиваемого подхода для комбинирования значений из инвентарных списков не кажется таким уж сложным. Инвентарные списки применяются в качестве таблицы контрольных проверок. Необходимо выбрать типичные установки, которые используются большинством налогоплательщиков. Теперь сосредоточимся на нормальном использовании системы, чтобы показать, что в типичных условиях приложение работает как следует. Необычные комбинации данных будут рассмотрены позже.

В табл. 1.5 содержатся тесты с различными инвентарными списками. Столбцы с заголовками, помеченными звездочкой (*), содержат те статьи отчислений, для которых существуют ограничения, связанные с величиной заработка. Формат таблицы изменен, чтобы можно было лучше отследить все возможные комбинации.

Стадия 6. Граничные оценки

В тестах, выполняемых до настоящего момента, использовались типичные сценарии, которые применяются чаще всего. Следующий этап заключается в исследовании пределов возможностей приложения. Пределы возможностей приложения (или границы) определяются данными. Эти пределы принимают множество форм в зависимости от типа данных. Примерами могут служить:

- минимальные и максимальные значения диапазона данных;
- минимальный и максимальный размер поля (например, минимальное и максимальное число символов);
- минимальный и максимальный размер буфера.

Таблица 1.5. Тестовый пример с комбинацией инвентаризации

Входные данные							Статьи
ID тестового примера	**Статус**	**Заработок, ($)**	**Материально зависимые лица**	**Налог на медицину и стоматологию*, ($)**	**Государственный и местный налог, ($)**	**Налог на недвижимость, ($)**	**Налог на движимое имущество, ($)**
налог 15	одинокий (-ая)	40000	0	0	2200	1890	80
налог 16	женат (замужняя) с независимой декларацией доходов	60000	0	0	2200	1890	80

Общее правило тестирования границ — создать три тестовых примера, чтобы охватить следующие значения:

- граничное значение;
- граничное значение –1;
- граничное значение +1.

В налоговом калькуляторе предлагается использовать несколько границ, связанных, например, со следующими моментами:

- для каждой налоговой категории определяется диапазон с минимальным и максимальным значениями;
- как вычислять налог в зависимости от того, будет ли облагаемая налогом прибыль больше или меньше $100 000;
- некоторые статьи отчислений ограничиваются 2% от скорректированного валового дохода;
- отчисления с разбивкой по статьям должны превышать стандартное отчисление;
- по закону не существует верхнего предела на то, сколько человек может заработать; количество материально зависимых лиц также неограничено. Минимум — это, конечно, ноль.

Для большинства приложений типы данных имеют четко определенные верхние и нижние границы. Очевидно, что нужно перечислить границы всех налоговых категорий, а затем создать тест, использующий эти минимумы и максимумы. Однако налоговый кодекс США работает иначе. С помощью авторитетного лица проекта определяются те границы, которые влияют на вычисление налога для одинокого налогоплательщика. Инструкции, предоставленные Налоговым управлением США, содержат

отчислений						Выходные данные	
Процент по закладной, ($)	Пожертвования благотворительным организациям, ($)	Невозмещенные затраты на содержание наемного работника*, ($)	Плата за подготовку налога, ($)	Прочие отчисления, ($)	Сумма отчисления, ($)	Ожидаемые результаты: причитающийся налог, ($)	Реальные результаты
2800	500	100	50	0	7 470	4 911	
2800 0 0 0 0 0	500	100	50	0	7 470	11 073	

множество таблиц для вычисления налога. В табл. 1.7 показан список налоговых ставок, который используется тогда, когда облагаемый налогом доход одинокого налогоплательщика составляет $100 000 и больше. В основных инструкциях утверждается, что налогоплательщик с облагаемым налогом доходом меньше $100 000 не может использовать эту таблицу, однако данные инструкции составлены так, что налогоплательщик может увидеть налоговую ставку, которая применяется на каждом уровне. Такой список налоговых ставок определен для четырех диапазонов данных:

облагаемый налогом доход < $100 000	использовать иной способ для вычисления налога
$100 000 ≤ облагаемый налогом доход < $132 600	налогоплательщик находится в налоговой категории “31%”
$132 600 ≤ облагаемый налогом доход < $288 350	налогоплательщик находится в налоговой категории “36%”
$288 350 ≤ облагаемый налогом доход	налогоплательщик находится в налоговой категории “39,61%”

Диапазоны данных устанавливаются, исходя из суммы облагаемой прибыли. Приблизительно облагаемая прибыль состоит из суммы всех типов доходов, из которой вычтены все разрешенные корректировки и отчисления. Существует множество способов получения результирующей облагаемой налогом прибыли. Чтобы не усложнять объяснения, дальше будут вноситься изменения в базовый тестовый пример. Для одинокого налогоплательщика, у которого нет материально зависимых лиц или отчислений, облагаемая налогом прибыль получается путем вычитания из заработка стандартного отчисления $4 400 и одного освобождения от налога в размере $2 800. Это справедливо при условии, что скорректированный валовой доход меньше $96 700, в противном случае величина освобождения от налога, которую можно

вычесть, ограничивается. Ключевым моментом здесь является определение входящей финансовой информации, которая дает желаемую облагаемую налогом прибыль.

Таблица 1.7. Список налоговых ставок для налогоплательщика со статусом "Одинокий (–ая)"

Список X – использовать, если статус **"Одинокий (–ая)"**

Только если облагаемая налогом прибыль (из формы 1040, строка 39) составляет $100 000 или больше.

Если величина в форме №1040, строка 39 превышает, ($):	Но не превышает, ($):	Ввести в форму №1040, строка 40, ($):			Из величины, превышающей, ($):
0	26 250		+	15%	0
26 250	63 550	3937,50	+	28%	26 250
63 550	132 600	14381,50	+	31%	63 550
132 600	288 350	35787,00	+	36%	132 600
288 350	...	91857,00	+	39,6%	288 350

Источник: инструкция №1040 на 2000-й налоговый год, Налоговое управление, министерство финансов США

Когда определены диапазоны данных, следующим этапом становится разработка тестовых примеров для тестирования границ. Поскольку эти границы узкие, т.е. два значения находятся на противоположных концах диапазона, понадобится только два тестовых примера для каждой границы (пример *разбиения на классы эквивалентности* описан в главе 4). Следовательно, тесты, перечисленные в табл. 1.8, используют значения в граничных точках и по ту сторону границы. В результате округления некоторые из ожидаемых результатов оказываются идентичными. Эти результаты отличались, если бы при вычислениях сохранялись значения с точностью до цента. Два дополнительных столбца используются исключительно с информационной целью. Поскольку именно облагаемая налогом прибыль определяет тестируемые в данный момент границы (а также налоговую категорию), то эта информация будет включена вместе с данными тестового примера.

Когда облагаемая налогом прибыль меньше $100 000, для определения ставок используется справочная таблица. В табл.1.9 показана справочная таблица для облагаемой налогом прибыли в диапазоне от $26 000 до $27 000. Налоговое управление США предоставляет такие таблицы для каждого значения от $0 до $100 000. Разбиение справочной таблицы с шагом в $50, на каждом из которых определяется верхнее и нижнее значение, требует ввода 2000 значений. Поскольку справочная таблица создана с шагом $50, возникает вопрос: какое же значение использовать для тестирования? Чтобы гарантировать, что каждое значение в таблице верное, в действительности нужно создать тест для каждого диапазона значений. Но такой подход непрактичен, поскольку отнимает очень много времени.

Таблица 1.8. Тестовый пример, использующий список налоговых ставок

ID тестового примера	Статус	Заработок, ($)	Материально зависимые лица	Отчисления, ($)	Ожидаемые результаты: причитающийся налог, ($)	Реальные результаты	Примечание: величина облагаемой налогом прибыли, ($)	Примечание: метод вычисления налога
налог 17	одинокий (-ая)	107 200	0	0	25 681		100 000	налоговая категория “31%”
налог 18	одинокий (-ая)	139 519	0	0	35 787		132 599	налоговая категория “31%”
налог 19	одинокий (-ая)	139 520	0	0	35 787		132 600	налоговая категория “36%”
налог 20	одинокий (-ая)	292 749	0	0	91 857		288 349	налоговая категория “36%”
налог 21	одинокий (-ая)	292 750	0	0$	91 857		288 350	налоговая категория “39,6%”

Таблица 1.9. Таблица налогов для облагаемой налогом прибыли в диапазоне между $26 000 и $27 000

Если строка 39 (облагаемая налогом прибыль) составляет:		и Вы:			
по меньшей мере	но меньше чем	одинокий (-ая)	женат (замужняя) с независимой декларацией доходов*	женат (замужняя) с совместной декларацией доходов	глава семьи
			Ваш налог составляет, ($):		
26 000	26 050	3 904	3 904	4 437	3 904
26 050	26 100	3 911	3 911	4 451	3 911
26 100	26 150	3 919	3 919	4 465	3 919
26 150	26 200	3 926	3 926	4 479	3 926
26 200	26 250	3 934	3 934	4 493	3 934
26 250	26 300	3 945	3 941	4 507	3 941
26 300	26 350	3 959	3 949	4 521	3 949
26 350	26 400	3 973	3 956	4 535	3 956
26 400	26 450	3 987	3 964	4 549	3 964
26 450	26 500	4 001	3 971	4 563	3 971
26 500	26 550	4 015	3 979	4 577	3 979
26 550	26 600	4 029	3 986	4 591	3 986
26 600	26 650	4 043	3 994	4 605	3 994
26 650	26 700	4 057	4 001	4 619	4 001
26 700	26 750	4 071	4 009	4 633	4 339
26 750	26 800	4 085	4 016	4 647	4 316
26 800	26 850	4 099	4 024	4 661	4 024
26 850	26 900	4 113	4 031	4 675	4 031
26 900	26 950	4 127	4 039	4 689	4 039
26 950	27 000	4 141	4 046	4 703	4 046

*Этот столбец также должен использовать вдовец (вдова)

Источник: инструкция №1040 на 2000-й налоговый год, Налоговое управление, министерство финансов США

Во время тестирования справочной таблицы необходимо:

- убедиться, что индекс предоставляет доступ к нужной части таблицы;
- удостовериться, что значения в таблице верны.

Использование значения облагаемой налогом прибыли в качестве индекса в таблице кажется совершенно логичным. Разработчики должны быть уверены в том, что приложение предоставляет доступ к верной позиции в таблице, и, следовательно, поисковая часть работает правильно для всех индексов. Если это так, нужно внимательно изучить первый диапазон в табл. 1.9. Это даст возможность ввести два дополнительных теста, как показано в табл. 1.10.

Выполнение тестовых примеров "налог 22" и "налог 23" демонстрирует работоспособность приложения по крайней мере, для двух отдельных значений. Можно предположить, но не гарантировать, что приложение будет работать и для других диапазонов. А как же насчет сохранности данных в таблице? Единственный реальный способ проверить данные — создание тестов для каждого элемента справочной таблицы. Опять же, временны*е* ограничения делают это невозможным. Предполагая, что значения таблицы сохранены в файле какого-либо типа, самым эффективным путем проверки значений таблицы будет проинспектировать их, т.е. сделать копии таблиц и сравнить их с таблицами, предоставленными Налоговым управлением США. Нельзя сказать наверняка, какой подход нужно использовать. Какое бы ни было принято решение, следует убедиться, что авторитетное лицо проекта не возражает против такого подхода.

Граничным значением облагаемой налогом прибыли, составляющим $100 000, определяется то, будет ли для вычисления налога использоваться список налоговых ставок или таблица налогов. Хотелось бы иметь тест, который бы устанавливал облагаемую налогом прибыль в размере $99 999 и $100 000, как показано в табл. 1.11.

Если в тестовом примере "налог 24" приложение ошибочно использует при вычислениях для определения налога налоговую категорию 31%, вместо того, чтобы использовать таблицу налогов, оно возвратит неверное значение $25 681. Этот тест дает возможность получить обратную связь, чтобы проверить, использует ли приложение для вычисления налога подходящий метод.

Конечно, нужно также протестировать нулевое значение для различных полей, как это показано в табл. 1.12. Ожидается, что приложение "Налоговый калькулятор" будет отлавливать отрицательные значения и отправлять сообщение об ошибке. В тестовом примере "налог 30" доход составляет $7 200, что дает в результате значение облагаемой налогом прибыли, равное $0 и, следовательно, в столбце с причитающимся налогом окончательное значение равно $0.

В ходе определения тестов были сделаны предположения относительно того, какими будут граничные значения для тестирования и какие значения из него исключаются. Анализ степени риска помогает удостовериться, был ли выбран подходящий тест.

До сих пор рассматривалось только влияние данных на налогоплательщика со статусом "одинокий (-ая)". Аналогичные наблюдения применимы и к другим статусам, поскольку в списке налоговых ставок для различных статусов определяются различные граничные значения, а в таблицу налогов для различных статусов возвращаются различные значения. Понадобится также определить тесты, которые будут касаться границ других типов данных, аналогично тем, что касаются статей отчислений.

Таблица 1.10. Тестовый пример для поиска по облагаемой налогом прибыли

ID тестового примера	Статус	Заработок, ($)	Материально зависимые лица	Отчисления, ($)	Ожидаемые результаты причитающийся налог, ($)	Реальные результаты	Примечание: величина облагаемой налогом прибыли, ($)
налог 22	одинокий (-ая)	33 249	0	0	3 904		26 049
налог 23	одинокий (-ая)	33 250	0	0	3 911		26 050

Таблица 1.11. Тестирование граничного значения облагаемой налогом прибыли в размере $100 000

ID тестового примера	Статус	Заработок, ($)	Материально зависимые лица	Отчисления, ($)	Ожидаемые результаты: причитающийся налог, ($)	Реальные результаты	Примечание: величина облагаемой налогом прибыли, ($)	Примечание: метод вычисления налога
налог 24	одинокий (-ая)	107 199	0	0	25 673		99 999	Справочная таблица налогов
налог 25	одинокий (-ая)	107 200	0	0	25 681		100 000	налоговая категория “31%”

Таблица 1.12. Тест, проверяющий нулевую границу

ID тестового примера	Статус	Заработок, ($)	Материально зависимые лица	Величина отчислений, ($)	Ожидаемые результаты: причитающийся налог	Реальные результаты
налог 26	одинокий (-ая)	0	0	0	0	
налог 27	одинокий (-ая)	–1	0	0	Ошибка	
налог 28	одинокий (-ая)	0	0	–1	Ошибка	
налог 29	одинокий (-ая)	0	–1	0	Ошибка	
налог 30	одинокий (-ая)	7 200	0	0	0	

Стадия 7. Ошибочные данные

На следующей стадии тестирования делается попытка вывести приложение из строя, создавая условия, в которых обычный пользователь, вероятнее всего, работать не будет. Рассмотрим создание тестов следующих категорий.

- Данные не вводятся для того, чтобы проследить поведение приложения: обеспечит ли оно прерывание или сгенерирует сообщение об ошибке.
- Вводятся неверные числовые данные (отрицательные значения или буквенно-цифровые комбинации символов).
- Вводятся данные какого-либо формата, который для такого типа данных считается недопустимым.
- Используются необычные комбинации данных.
- Проверяется использование нулевого значения, если это еще не сделано в предыдущих тестах.

Необходимо дать текущим пользователям возможность поработать с новой системой. Нужно посмотреть, какие значения они вводят и как они интерпретируют подсказки системы.

Цель определения необычных комбинаций заключается в проверке приемлемого поведения приложения, даже в том случае, если тестовые данные не описывают нормального налогоплательщика. В пример с налоговым калькулятором можно включить следующие необычные ситуации.

1. Очень низкий заработок плюс очень большое число материально зависимых лиц.
2. Очень высокий заработок при отсутствии отчислений.
3. Отчисления, превышающие величину суммарной прибыли.

При создании тестового примера с ошибочными данными в результате обычно появляется сообщение об ошибке. Ожидается, что приложение отловит эти данные и проинформирует пользователя об ошибке. При таких условиях тест считается пройденным, поскольку система, как и ожидалось, возвратила сообщение об ошибке.

В некоторых случаях предыдущую стадию тестирования можно обойти и всецело сосредоточиться на том, чтобы вывести систему из строя и найти действительно досадные ошибки. Хотя у этого подхода есть свои преимущества, наилучшей тактикой будет проанализировать риск, чтобы определить, в каком направлении следует прилагать усилия по тестированию и за счет отказа от тестирования каких функций это нужно сделать. Когда на тестирование продукта назначено несколько человек, один из них может посвятить свое время исключительно нахождению серьезных ошибок. Фокусировка деятельности исключительно на нахождении серьезных дефектов создает некоторые сложности, так как многие проблемы не приводят к аварийным ситуациям. Тестеры должны достаточно хорошо знать приложение, чтобы определять, верны ли выходные данные. Попытки создать аварийную ситуацию в системе часто состоят в поверхностном наборе команд. При этом, однако, могут возникнуть

сложности, связанные с тем, что входная информация, которая привела к возникновению неполадки, не была документирована.

В главе 2 представлена таблица контрольных проверок верных и неверных данных, помогающая определить ошибочные данные.

Стадия 8. Создание напряжений

На следующей стадии тестеры отходят от функций приложения и оценивают условия их реализации в терминах рабочих характеристик и восстановления системы.

Оценка рабочих характеристик обычно включает измерение времени, которое уходит на выполнение отдельных операций. Характерные времена варьируются, если приложение запускается в специализированной или полностью загруженной системе. Еще один аспект, влияющий на рабочие характеристики системы, требует создания во время тестирования напряжений в среде, в которой она работает. Данный аспект включает следующие тесты.

- Уменьшение объема доступной памяти.
- Использование всего доступного дискового пространства.
- Параллельный запуск нескольких экземпляров приложения.
- Запуск приложения в момент, когда система выполняет резервирование.
- Генерация множества асинхронных, управляемых событиями процессов.

Иногда приложению приходится восстанавливаться после неполадок. Даже если тестер перезагружает систему, после перезапуска все должно функционировать корректно. Тесты, наносящие вред среде, должны включать:

- выбрасывание приложения;
- разъединение кабелей;
- отключение питания.

Доказательством того, что приложению был нанесен сильный ущерб, служат:

- недоступные файлы;
- открытие заблокированных файлов;
- разрушенные базы данных;
- невозможность перезапуска приложения.

В описании категорий тестов в главе 2 определены другие методы создания напряжений в среде.

Дальнейшие шаги

В образцах тестовых примеров представлено только использование значений данных в налоговом приложении. Естественно, для проверки функциональных возможностей других полей данных, которые не обсуждались в этой главе, нужны дополнительные тесты.

График выпусков достаточно сжат, поэтому в большинстве тестов в этой главе внимание было сосредоточено на демонстрации того, что приложение работает так, как было задумано. Хотя инвентаризация дает начало многим тестам, для полного изучения системы требуются другие тесты. Использование методов схем (см. главы 2 и 3) и таблиц (см. главы 4 и 5) поможет определить эти дополнительные тесты.

Подход на основе теории графов дает тестовые примеры, которые определяют все возможные пути в приложении. Эти методы включают:

- тестирование управляющей логики;
- тестирование потока данных;
- тестирование операционных потоков.

Данный метод объяснен и очень детально проиллюстрирован в работе [Beizer 95]. В отличие от предварительных тестов, использованных в этой главе, приближения теории графов приводят к исчерпывающему анализу программного обеспечения.

Совершенно разные категории тестов позволяют убедиться, что приложение *не* делает того, что оно не должно делать. На предыдущей стадии были созданы тесты, в которых была предпринята попытка вывести систему из строя (например, введение запредельных значений данных). Однако нужны еще тесты, в которых можно будет отыскать нежелательные побочные эффекты, такие как измененная память, разрушенные буфера или другие непредусмотренные результаты.

Стратегия, предложенная в данной главе, — это один из эффективных способов, с помощью которого можно разобраться с проблемой тестирования плохо документированных приложений в сжатые сроки. За другими идеями относительно того, как "тестировать в темноте", можно обратиться к работе [Rothman 99].

Успешное тестирование, — как и качество программного обеспечения, — зависит от многих других дисциплин разработки ПО, включая:

- *конфигурационный менеджмент*, чтобы иметь возможность следить, какая версия приложения в данный момент тестируется;
- *составление отчетов о неполадках*, чтобы работать со списком спорных вопросов и проблем, найденных тестером;
- *управление внесением изменений*, чтобы иметь возможность отслеживать, какие неполадки исправляются в текущем выпуске и чтобы сохранять список тех неполадок, которые не будут устранены в ближайшем будущем.

В главе 9 обсуждается важность этих тем для тестирования программного обеспечения.

Резюме

Даже когда близится выпуск продукта, тестеры могут оценить критичные компоненты приложения и подтвердить, что ключевые функции работают корректно. Естественно, тестеры могут обеспечить лучшую обратную связь, если в их распоряжении

достаточно времени. При тестировании приложения в сжатые сроки можно выделить следующие основные этапы.

Каждая стадия базируется на предыдущих. Основная тактика — определить наиболее типичный пользовательский сценарий и использовать эти данные в качестве базовых для создания дополнительных тестов. Изменяя за раз один параметр, тестер может убедиться, что каждое изменение влияет на приложение так, как было задумано.

Стадия 1:	Изучить приложение. Установить авторитетное лицо проекта.
Стадия 2:	Определить базовый тестовый пример, для выполнения которого необходимо создать среду тестирования.
Стадия 3:	Проанализировать тенденции, если сложно спрогнозировать результаты.
Стадия 4:	Определить инвентарные списки. Исходя из списков, сгенерировать тестовые примеры.
Стадия 5:	Скомбинировать данные из различных инвентарных списков.
Стадия 6:	Сделать граничные оценки.
Стадия 7:	Создать тесты с ошибочными данными, чтобы попытаться вывести систему из строя.
Стадия 8:	Создать напряжение в системных ресурсах. Определить, как система справляется с аварийными ситуациями.

Иногда тестерам приходится выбирать, какие тесты будут заданы и выполнены, поскольку для тестирования всех функций не хватает времени. Анализ степени риска (см. главу 8) помогает определить наиболее важные для тестирования области даже перед созданием какого-либо тестового примера.

Представленные здесь идеи помогут тестерам начать действовать. Для полного тестирования системы требуется дополнительная работа. В последующих главах описываются методы планирования и создания дополнительных тестовых примеров.

ГЛАВА 2

Схемы тестов

Введение

Часть проекта, которая касается тестирования, может оказаться слишком трудной для многих тестеров. Иногда проект кажется огромным, а данные беспорядочно разбросанными. В таком случае, очевидно, что многим тестерам сложно получить необходимую информацию и определить, какие дополнительные данные могут понадобиться для разработки тестовых примеров.

В этой главе будет рассмотрено типичное описание продукта вместе с далекими от идеала требованиями. Даже если работа организации в области тестирования все еще не налажена, можно продолжить разработку тестовых примеров на ранней стадии и определить проблемы.

Вместо того чтобы жаловаться на несовершенные требования тестерам следует поддерживать связь с командой разработчиков и попытаться усовершенствовать определение продукта. Во время рассмотрения требований авторитетное лицо проекта предоставляет дополнительную информацию (пользовательскую документацию и ответы на возникающие по ходу работы вопросы). В этой главе все действия тестеров представлены в хронологическом порядке, чтобы показать, что для начала не обязательно нужна вся информация о приложении. В лучшем случае работа должна проводиться параллельно с разработчиками, когда тестовые примеры и программное обеспечение разрабатываются одновременно. Со временем более детальная информация о приложении становится доступной, так как тестер использует итеративный подход к пониманию требований.

Работа тестера состоит в том, чтобы:

- узнать о новом приложении;
- перечислить возможности тестового примера;
- определить тестовый пример;
- выбрать, какие будут проводиться тесты;

- создать среду тестирования;
- провести тесты.

В примерах этой главы для работы на первых двух этапах используется схематический подход. Тестовые примеры определены в главе 3, а в главе 8 выбраны наилучшие из них. В главе 9 обсуждаются последующие этапы в цикле тестирования.

Пример приложения

Примером приложения служит контроллер духового шкафа, его функции кратко описаны на рис. 2.1. На рис. 2.2 представлена диаграмма контекста, на которой показаны входные и выходные данные для контроллера духового шкафа. Задача тестера — проанализировать требования с последующим определением набора тестов.

Краткое описание приложения

Спроектировать контроллер для электрического духового шкафа, который позволит пользователю выбирать нужную температуру приготовления и задавать время. Духовой шкаф будет разогреваться и выключаться согласно введенной пользователем информации.

Рис. 2.1. Описание приложения

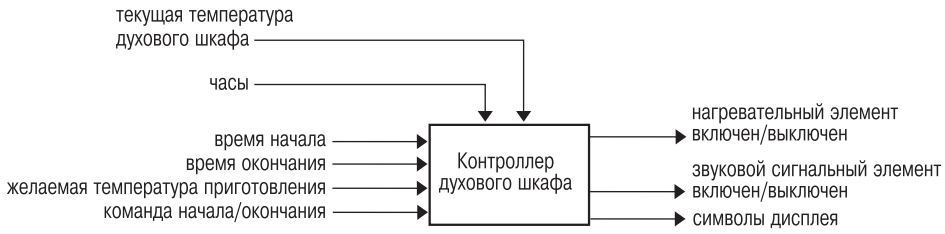

Рис. 2.2. Диаграмма контекста контроллера духового шкафа

В дополнение к перечисленной выше информации о продукте предоставлен набор так называемых “требований”, поскольку их содержимое не описывает реальных системных требований. В некоторых проектах приложения описываются более чем туманно.

Требований явно не хватает, однако для того, чтобы продолжить разработку теста, информации вполне достаточно. Для построения описания теста и его определения несложно будет сконцентрировать внимание на проблеме с данными требованиями. В идеале, кто-то должен провести анализ требований, чтобы убедиться, что они полные. Однако не каждая организация проходит через этапы, повышающие качество системы. Несмотря на это серьезное упущение, можно все-таки установить наличие проблем на ранней стадии цикла разработки, т.е. до того, как разработчики начнут проектировать и писать код.