Лекция 9. Тестирование пользовательского интерфейса

Лекция посвящена тестированию пользовательского интерфейса. Определяются задачи и цели данного вида тестирования, определяются методы функционального тестирования пользовательского интерфейса, вводятся основные понятия тестирования удобства использования (usability) интерфейсов. Цель данной лекции: дать представление о процессе тестирования пользовательского интерфейса, его задачах, целях и основных методах

9.1. Задачи и цели тестирования пользовательского интерфейса

Часть программной системы, обеспечивающая работу интерфейса с пользователем - один из наиболее нетривиальных объектов для верификации. Нетривиальность заключается в двояком восприятии термина "пользовательский интерфейс".

С одной стороны, пользовательский интерфейс - часть программной системы. Соответственно, на пользовательский интерфейс пишутся функциональные и низкоуровневые требования, по которым затем составляются тест-требования и тест-планы. При этом, как правило, требования определяют реакцию системы на каждый ввод пользователя (при помощи клавиатуры, мыши или иного устройства ввода) и вид информационных сообщений системы, выводимых на экран, печатающее устройство или иное устройство вывода. При верификации таких требований речь идет о проверке функциональной полноты пользовательского интерфейса - насколько реализованные функции соответствуют требованиям, корректно ли выводится информация на экран.

С другой стороны, пользовательский интерфейс - "лицо" системы, и от его продуманности зависит эффективность работы пользователя с системой. Факторы, влияющие на эффективность работы, слабо поддаются формализации в виде конкретных требований к отдельным элементам, однако должны быть учтены в виде общих рекомендаций и принципов построения пользовательского интерфейса программной системы. Проверка интерфейса на эффективность человеко-машинного взаимодействия получила название проверки удобства использования (usability verification ; в русскоязычной литературе в качестве перевода термина usability часто используют слово "практичность").

В данной лекции будут рассмотрены общие вопросы как функционального *тестирования пользовательских интерфейсов*, так и *тестирования удобства использования*.

9.2. Функциональное тестирование пользовательских интерфейсов

Функциональное *тестирование пользовательского интерфейса* состоит из пяти фаз:

- анализ требований к пользовательскому интерфейсу;
- разработка тест-требований и тест-планов для проверки пользовательского интерфейса;
- выполнение тестовых примеров и сбор информации о выполнении тестов;
- определение полноты покрытия пользовательского интерфейса требованиями;
- составление отчетов о проблемах в случае несовпадения поведения системы и требований либо в случае отсутствия требований на отдельные интерфейсные элементы.

Все эти фазы точно такие же, как и в случае тестирования любого другого компонента программной системы. Отличия заключаются в трактовке некоторых терминов в применении к пользовательскому интерфейсу и в особенностях автоматизированного сбора информации на каждой фазе.

Так, тест-планы для проверки пользовательского интерфейса, как правило, представляют собой сценарии, описывающие действия пользователя при работе с системой. Сценарии могут быть записаны либо на естественном языке, либо на формальном языке какой-либо системы автоматизации пользовательского интерфейса. Выполнение тестов при этом производится либо оператором в ручном режиме, либо системой, которая эмулирует поведение оператора.

При сборе информации о выполнении тестовых примеров обычно применяются технологии анализа выводимых на экран форм и их элементов (в случае графического интерфейса) или выводимого на экран текста (в случае текстового), а не проверка значений тех или иных переменных, устанавливаемых программной системой.

Под полнотой покрытия пользовательского интерфейса понимается то, что в результате выполнения всех тестовых примеров каждый интерфейсный элемент был использован хотя бы один раз во всех доступных режимах.

Отчеты о проблемах в пользовательском интерфейсе могут включать в себя как описания несоответствий требований и реального поведения системы, так и описания проблем в требованиях к пользовательскому интерфейсу. Основной источник проблем в этих требованиях - их тестонепригодность, вызванная расплывчатостью формулировок и неконкретностью.

9.2.1. Проверка требований к пользовательскому интерфейсу

9.2.1.1. Типы требований к пользовательскому интерфейсу

Требования к пользовательскому интерфейсу могут быть разбиты на две группы:

- требования к внешнему виду пользовательского интерфейса и формам взаимодействия с пользователем;
- требования по доступу к внутренней функциональности системы при помощи пользовательского интерфейса.

Другими словами, первая группа требований описывает взаимодействие подсистемы интерфейса с пользователем, а вторая - с внутренней логикой системы.

К первой группе можно отнести следующие типы требований.

- **Требования к размещению элементов управления на экранных формах**

  Данные требования могут определять общие принципы размещения элементов пользовательского интерфейса или требования к размещению конкретных элементов. Например, общие требования по размещению элементов на графической экранной форме могут выглядеть следующим образом:

  Каждое окно приложения должно быть разбито на три части: строка меню, рабочая область и статусная строка. Строка меню должна быть

горизонтальной и прижатой к верхней части окна, статусная строка должна быть горизонтальной и прижатой к нижней части окна, рабочая область должна находиться между строкой меню и статусной строкой и занимать всю оставшуюся площадь окна.

При тестировании данного требования достаточно определить, что в каждом окне системы действительно присутствуют три части, которые расположены и прижаты согласно требованиям даже при изменении размеров окна, его сворачивании/разворачивании, перемещении по экрану, при перекрытии его другими окнами.

Примером требований по размещению конкретного элемента может служить следующее:

Кнопка "Начать передачу" должна находиться непосредственно под строкой меню в левой части рабочей области окна.

При тестировании такого требования также необходимо определить, сохраняется ли расположение элемента при изменении размера окна, а также при использовании элемента (в данном случае - при нажатии).

- **Требования к содержанию и оформлению выводимых сообщений**

  Требования к содержанию и оформлению выводимых сообщений определяют текст сообщений, выводимых системой, его шрифтовое и цветовое оформление. Также часто в таких требованиях определяется, в каких случаях выводится то или иное сообщение.

  Так, например, для тестирования требования

  Сообщение "Невозможно открыть файл" должно выводиться в статусную строку прижатым к левому краю, красным цветом, полужирным шрифтом в случае недоступности открываемого файла по чтению.

  необходимо проверить, что при возникновении указанной ситуации сообщение действительно выводится согласно требованиям.

  Однако в случае тестирования требования вида

  Сообщения об ошибках должны выводиться в статусную строку прижатыми к левому краю красным цветом полужирным шрифтом.

  необходимо проверять форматы всех возможных сообщений об ошибках программы во всех возможных ошибочных ситуациях. Таким образом, очевидно, что при тестировании пользовательского интерфейса не всегда можно однозначно определить количество тестовых примеров, которые понадобятся для тестирования требования. Эта проблема вызвана тем, что требования к пользовательскому интерфейсу зачастую кажутся слишком очевидными для их точной формулировки. Эта неконкретность требований и вызывает большое количество тестов для каждого требования.

- **Требования к форматам ввода**

Данная группа требований определяет, в каком виде информация поступает от пользователя в систему. При этом кроме собственно требований, определяющих корректный формат, к этой группе относятся требования, определяющие реакцию системы на некорректный ввод. Для проверки таких требований необходимо проверять как корректный ввод, так и некорректный. Желательно при этом разбивать различные варианты ввода на классы эквивалентности (как минимум на два - корректные и некорректные).

Ко второй группе относятся следующие типы требований.

- **Требования к реакции системы на ввод пользователя**

  Данный тип требований определяет связь внутренней логики системы и интерфейсных элементов. Например,

  При нажатии кнопки "Сброс" значение таймера синхронизации передачи должно сбрасываться в 0.

  Для проверки такого требования в тестовом примере должно быть сымитировано нажатие на кнопку "Сброс", после чего должна проводиться проверка значения таймера. Однако некоторые требования определяют в качестве реакции системы не то, как меняется ее внутреннее состояние, а реакцию пользовательского интерфейса. Например, в требовании

  При нажатии кнопки "Отложенный сброс" должно выводиться окно "Ввод значения времени для отложенного сброса".

  в качестве реакции на использование одного интерфейсного элемента определяется появление другого интерфейсного элемента. Такие требования проверяются при помощи имитации ввода пользователя и анализа появляющихся интерфейсных элементов.

- **Требования к времени отклика на команды пользователя**

  В качестве отдельного типа требований можно выделить требования к времени отклика системы на различные пользовательские операции. Это связано с тем, что подсознательно пользователь воспринимает операции продолжительностью более 1 секунды как длительные. Если в этот момент система не сообщает пользователю о том, что она выполняет какую-либо операцию, пользователь начнет считать, что система зависла или работает в неверном режиме. В связи с этим либо каждое предельное время отклика должно быть указано в требованиях и пользовательской документации, либо во время длительных операций должны выводиться информационные сообщения (например, индикатор прогресса). Значения предельного времени и равномерность увеличения значений индикатора прогресса должны проверяться соответствующими тестами.

9.2.1.2. Тестопригодность требований к пользовательскому интерфейсу

Некоторые требования к пользовательскому интерфейсу могут оказаться тестонепригодными, либо их тестирование будет значительно затруднено. К таким требованиям в первую очередь относятся требования, описывающие субъективные характеристики интерфейса, которые не могут быть точно определены или измерены при

выполнении тестовых примеров. При анализе требований к пользовательскому интерфейсу необходимо четко представлять, какой элемент интерфейса и каким образом будет проверяться, какая его характеристика будет измеряться в ходе тестирования.

Примером тестонепригодного требования может служить классическое требование

Пользовательский интерфейс должен быть интуитивно понятным.

Без определения четких критериев интуитивной понятности проверка такого требования невозможна. При этом необходимо понимать, что критерий в данном случае может быть двух видов: детерминированным или вероятностным. Примером детерминированного критерия может быть дополнение к требованию вида

Под интуитивной понятностью интерфейса понимается доступность любой функции системы при помощи не более чем 5 щелчков мыши по интерфейсным элементам.

Требование с таким уточнением поддается как ручному, так и автоматизированному тестированию, более того, результат такого тестирования не будет зависеть от субъективного мнения тестировщика (понятия об интуитивной понятности у всех разные).

Примером вероятностного критерия может служить следующее дополнение:

Под интуитивной понятностью интерфейса понимается, что пользователь обращается к руководству пользователя не чаще, чем раз в пять минут на этапе обучения и не чаще, чем раз в 2 часа на этапе активного использования системы. Значения должны быть получены на репрезентативной выборке пользователей не менее 1000 человек.

Проверка требования с таким дополнением не является задачей классической верификации и относится уже скорее к проверке удобства применения пользовательского интерфейса. Однако здесь также вводится четкий критерий, при использовании которого результаты тестирования могут быть воспроизведены.

9.2.2. Полнота покрытия пользовательского интерфейса

При определении понятия покрытия пользовательского интерфейса можно ввести следующие его уровни:

- **функциональное покрытие** - покрытие требований к пользовательскому интерфейсу;
- **структурное покрытие** - для обеспечения полного структурного покрытия каждый интерфейсный элемент должен быть использован в тестовых примерах хотя бы один раз;
- **структурное покрытие с учетом состояния элементов интерфейса** - для обеспечения этого уровня покрытия необходимо не только использовать каждый элемент интерфейса, но и привести его во все возможные состояния (например, для чек-боксов - отмечен/не отмечен, для полей ввода - пустое/заполненное не целиком/заполненное полностью и т.п.)
- **структурное покрытие с учетом состояния элементов интерфейса и внутреннего состояния системы** - поведение некоторых интерфейсных элементов может изменяться в зависимости от внутреннего состояния системы. Каждое такое различимое поведение интерфейсного элемента должно быть проверено. Например, система может иметь два режима работы - нормальный и для

начинающего пользователя, в котором нажатие каждого элемента сопровождается появлением всплывающей подсказки. В этом случае нужно проверить оба режима и при этом проверить, что подсказки появляются только в режиме для начинающих.

При определении степени покрытия необходимо учитывать, что реакция на некоторые интерфейсные элементы определяется не программной системой, а на уровне операционной системы или среды выполнения. Так, например, реакция на использование многих интерфейсных элементов стандартного диалогового окна открытия файла определяется операционной системой и может не тестироваться.

Если уровень покрытия интерфейсных элементов тестами недостаточен, это является сигналом либо к уточнению требований к пользовательскому интерфейсу, либо к снижению степени подробности тестирования.

9.2.3. Методы проведения тестирования пользовательского интерфейса, повторяемость тестирования пользовательского интерфейса

Функциональное *тестирование пользовательского интерфейса* может проводиться различными методами - как вручную при непосредственном участии оператора, так и при помощи различного инструментария, автоматизирующего выполнение тестовых примеров. Рассмотрим эти методы более подробно.

9.2.3.1. Ручное тестирование

Ручное *тестирование пользовательского интерфейса* проводится тестировщиком-оператором, который руководствуется в своей работе описанием тестовых примеров в виде набора сценариев. Каждый сценарий включает в себя перечисление последовательности действий, которые должен выполнить оператор, и описание важных для анализа результатов тестирования ответных реакций системы, отражаемых в пользовательском интерфейсе. Типичная форма записи сценария для проведения ручного тестирования - таблица, в которой в одной колонке описаны действия (шаги сценария), в другой - ожидаемая реакция системы, а третья предназначена для записи того, совпала ли ожидаемая реакция системы с реальной и перечисления несовпадений (Таблица 9.1).

Таблица 9.1. Пример сценария для ручного *тестирования пользовательского интерфейса*

№ п/п	Действие	Реакция системы	Результат
1	Щелкните на пиктограмме System и выберите пункт меню 'System Management Applet'.	Появится окно ввода логина и пароля	Верно
2	Введите в появившееся окно ввода имя пользователя 'guest1' и пароль 'guest'. Затем нажмите кнопку 'Login'.	Появится окно 'System Management Applet'. В верхнем правом углу должно быть выведено имя вошедшего пользователя guest1 Все опции в окне должны быть отключены (выведены серым цветом)	Неверно Окно имеет название 'System *Management Application*'

3	Завершите сеанс работы с апплетом щелчком по пиктограмме 'Logout'.	Окно 'System Management Applet' должно быть закрыто	Верно

Ручное *тестирование пользовательского интерфейса* удобно тем, что контроль корректности интерфейса проводится человеком, т.е. основным "потребителем" данной части программной системы. К тому же при чисто косметических изменениях в интерфейсах системы, не отраженных в требованиях (например, при перемещении кнопок управления на 10 пикселей влево), анализ успешности прохождения теста будет выполняться не по формальным признакам, а согласно человеческому восприятию.

При этом *ручное тестирование* имеет и существенный недостаток - для его проведения требуются значительные человеческие и временные ресурсы. Особенно сильно этот недостаток проявляется при проведении регрессионного тестирования и вообще любого повторного тестирования - на каждой итерации повторного *тестирования пользовательского интерфейса* требуется участие тестировщика-оператора. В связи с этим в последнее десятилетие получили распространение средства автоматизации *тестирования пользовательского интерфейса*, снижающие нагрузку на тестировщика-оператора.

9.2.3.2. Сценарии на формальных языках

Естественный способ автоматизации *тестирования пользовательского интерфейса* - использование программных инструментов, эмулирующих поведение тестировщика-оператора при ручном тестировании пользовательского интерфейса.

Такие инструменты используют в качестве входной информации сценарии тестовых примеров, записанные на некотором формальном языке, операторы которого соответствуют действиям пользователя - вводу команд, перемещению курсора, активизации пунктов меню и других интерфейсных элементов.

При выполнении автоматизированного теста инструмент тестирования имитирует действия пользователя, описанные в сценарии, и анализирует интерфейсную реакцию системы. Для определения ожидаемого состояния пользовательского интерфейса здесь могут применяться различные методы - либо анализ снимков экрана и сравнение их с эталонными, либо доступ к данным интерфейсных элементов средствами операционной системы (например, доступ ко всем кнопкам окна по их дескрипторам и получение значений текста).

И при передаче информации в тестируемый интерфейс, и при получении информации для анализа могут использоваться два способа доступа к элементам интерфейса:

- позиционный, при котором доступ к элементу осуществляется при помощи задания его абсолютных (относительно экрана) или относительных (относительно окна) координат и размеров;
- по идентификатору, при котором доступ к элементу осуществляется при помощи получения интерфейсного элемента при помощи его уникального идентификатора в пределах окна.

При внесении изменений в пользовательский интерфейс при использовании первого метода в результате проведения регрессионного тестирования будет выявлено большое количество не прошедших тестов - достаточно изменения местоположения одного

ключевого интерфейсного элемента, как все сценарии начнут работать неверно. Соответственно при таком методе *автоматизации тестирования* необходимо менять значительную часть сценариев в системе тестов при каждом изменении интерфейса системы. Такой метод *автоматизации тестирования* подходит для систем с устоявшимся и редко изменяемым интерфейсом.

Второй метод *автоматизации тестирования* более устойчив к изменению расположения интерфейсных элементов, но изменения тестовых примеров могут потребоваться и здесь в случае изменения логики работы интерфейсных элементов. Например, пусть в первой версии системы при нажатии на кнопку "Передать данные" передача данных начиналась сразу и выводилось окно с индикатором прогресса. Сценарий тестового примера в этом случае включает в себя имитацию нажатия на кнопку и обращение к индикатору прогресса для получения значения прогресса в процентах.

Если во второй версии системы после нажатия на кнопку "Передать данные" вначале выводится окно "Вы уверены?" с кнопками "Да" и "Нет", то для проверки работы индикатора прогресса в тестовый сценарий необходимо добавить имитацию нажатия кнопки "Да".

Даже при обращении при помощи идентификаторов без модификации тестового примера тест не будет корректно выполняться. Тем не менее, этот способ обращения к интерфейсным элементам хорошо подходит для тестирования программных систем, в т.ч. с не устоявшимся пользовательским интерфейсом.

9.3. Тестирование удобства использования пользовательских интерфейсов

Удобство использования пользовательского интерфейса (usability) - показатель его качества, который определяет количество усилий, необходимых для изучения принципов работы с программной системой при помощи данного интерфейса, ее использования, подготовки входных данных и интерпретации выходных. Иначе говоря, удобство использования определяет степень простоты доступа пользователя к функциям системы, предоставляемым через человеко-машинный (пользовательский) интерфейс.

Тестирование удобства использования пользовательского интерфейса, вообще говоря, не относится к классическим методам тестирования программных систем. Специалист по тестированию пользовательского интерфейса должен сочетать в себе знания как в области программной инженерии, так и в физиологии, психологии и эргономике. Данный раздел курса не претендует на полноту изложения вопроса и дает самые общие представления о проблематике, связанной с тестированием удобства использования пользовательского интерфейса.

На удобство использования пользовательского интерфейса влияют следующие факторы:

- **легкость обучения** - быстро ли человек учится использовать систему;
- **эффективность обучения** - быстро ли человек работает после обучения;
- **запоминаемость обучения** - легко ли запоминается все, чему человек научился;
- **ошибки** - часто ли человек допускает ошибки в работе;
- **общая удовлетворенность** - является ли общее впечатление от работы с системой положительным.

Все эти факторы, несмотря на свою неформальность, могут быть измерены. Для таких измерений выбирается группа типичных пользователей системы, и в процессе их работы

измеряются показатели их работы с системой (например, количество допущенных ошибок), а также им предлагается высказать собственные впечатления от системы при помощи заполнения опросных листов.

Выделяют следующие этапы *тестирования удобства использования* пользовательского интерфейса [10].

1. **Исследовательское** - проводится после формулирования требований к системе и разработки прототипа интерфейса. Основная цель на этом этапе - провести высокоуровневое обследование интерфейса и выяснить, позволяет ли он с достаточной степенью эффективности решать задачи пользователя.
2. **Оценочное** - проводится после разработки низкоуровневых требований и детализированного прототипа пользовательского интерфейса. Оценочное тестирование углубляет исследовательское и имеет ту же цель. На данном этапе уже проводятся количественные измерения характеристик пользовательского интерфейса: измеряются количество обращений к системе помощи по отношению к количеству совершенных операций, количество ошибочных операций, время устранения последствий ошибочных операций и т.п.
3. **Валидационное** - проводится ближе к этапу завершения разработки. На этом этапе проводится анализ соответствия интерфейса программной системы стандартам, регламентирующим вопросы удобства интерфейса (например ISO 13407 [31], ISO 9126 [32]), проводится общее тестирование всех компонент пользовательского интерфейса с точки зрения конечного пользователя. Под компонентами интерфейса здесь понимается как его программная реализация, так и система помощи и руководство пользователя. Также на данном этапе проверяется отсутствие дефектов удобства использования интерфейса, выявленных на предыдущих этапах.
4. **Сравнительное** - данный вид тестирования может проводиться на любом этапе разработки интерфейса. В ходе сравнительного тестирования сравниваются два или более вариантов реализации пользовательского интерфейса.

Как правило, при тестировании удобства использования пользовательского интерфейса используются некоторые эвристические критерии и характеристики, которые заменяют точные оценки в классическом тестировании программных систем.

Например, Якоб Нильсен в своей работе [30] выделил 10 эвристических характеристик удобного пользовательского интерфейса, которые с его точки зрения должны проверяться при тестировании удобства использования интерфейса.

- **Наблюдаемость состояния системы**. Система всегда должна оповещать пользователя о том, что она в данный момент делает, причем через разумные промежутки времени.
- **Соотнесение с реальным миром**. Терминология, использованная в интерфейсе системы должна соотноситься с пользовательским миром, т.е. это должна быть терминология проблемной области пользователя, а не техническая терминология.
- **Пользовательское управление и свобода действий**. Пользователи часто выбирают отдельные интерфейсные элементы и используют функции системы по ошибке. В этом случае необходимо предоставлять четко определенный "аварийный выход", при помощи которого можно вернуться к предыдущему нормальному состоянию. К таким "аварийным выходам" относятся, например, функции отката и обратного отката.
- **Целостность и стандарты**. Для обозначения одних и тех же объектов, ситуаций и действий должны использоваться одинаковые слова во всех частях интерфейса.

Более того, терминология сообщений в пользовательском интерфейсе должна учитывать соглашения конкретной платформы.

- **Помощь пользователям в распознавании, диагностике и устранении ошибок**. Сообщения об ошибках должны быть написаны на естественном языке, а не заменяться кодами ошибок. Сообщения об ошибках должны четко определять суть возникшей проблемы и предлагать ее конструктивное решение.
- **Предотвращение ошибок**. Продуманный дизайн пользовательского интерфейса, предотвращающий появление ошибок пользователя, всегда лучше хорошо продуманных сообщений об ошибках. При проектировании интерфейса необходимо либо полностью устранить элементы, в которых могут возникать ошибки пользователя, либо проверять ввод пользователя в этих элементах и сообщать ему о потенциально возможном возникновении проблемы.
- **Распознавание, а не вспоминание**. При создании интерфейса необходимо минимизировать нагрузку на память пользователя, делая объекты, действия и опции ясными, доступными и явно видимыми. Пользователь не должен запоминать информацию при переходе от одного диалогового окна к другому. Во всех необходимых местах должны быть доступны контекстные инструкции по использованию интерфейса.
- **Гибкость и эффективность использования**. В интерфейсе должны быть предусмотрены горячие клавиши (не обязательные к использованию начинающим пользователем) - они часто значительно ускоряют работу опытного пользователя. Иными словами, система должна предоставлять два способа работы - для новичков и для опытных пользователей. Желательно при этом давать возможность пользователю автоматизировать часто повторяющиеся действия.
- **Эстетичный и минимально необходимый дизайн**. Окна не должны содержать не относящуюся к делу или редко используемую информацию. Каждый интерфейсный элемент, содержащий бесполезную информацию, играет роль информационного шума и отвлекает пользователя от действительно полезных интерфейсных элементов.
- **Помощь и документация**. Несмотря на то, что в идеальном случае лучше, когда системой можно пользоваться без документации, таковая все равно необходима - как в виде системы помощи, так и, возможно, в виде печатного руководства. Информация в документации должна быть структурирована таким образом, чтобы пользователь мог легко найти нужный раздел, посвященный решаемой им задаче. Каждый такой ориентированный на конкретную задачу раздел должен помимо общей информации содержать пошаговые руководства по выполнению задачи и не должен быть слишком длинным.

Все эти эвристики могут использоваться при тестировании удобства использования пользовательского интерфейса. Достаточно очевидно, что при тестировании удобства слабо применимы способы *автоматизации тестирования* при помощи сценариев и подобные методы. Один из наиболее эффективных методов проверки интерфейса на удобство - использование формальной инспекции. Вопросы в бланке инспекции могут быть как общего характера (так, например, можно использовать в качестве вопросов перечисленные выше 10 эвристик), так и вполне конкретными. Например, в работе [33] приводится список контрольных вопросов, которые желательно проверять при тестировании удобства использования web-сайтов. С некоторыми изменениями эти вопросы применимы и для обычных оконных интерфейсов.