



Верификация ПО

Тестирование интерфейса пользователя

К.А.Кулаков

Петрозаводск — 2017

Объект тестирования

- Графический интерфейс пользователя (Graphical user interface, GUI) - разновидность интерфейса обеспечивающее взаимодействие через графические элементы (меню, кнопки, значки, списки и т. п.).
- Варианты реализации GUI
 - интерфейс настольного приложения
 - мобильный интерфейс
 - веб интерфейс

Цели тестирования

- Обеспечение качественного взаимодействия с пользователем
 - Выявление ошибок функциональности
 - Выявление необработанных исключений при взаимодействии с интерфейсом
 - Выявление потери или искажения данных, передаваемых через элементы интерфейса
 - Выявление ошибки в интерфейсе (несоответствие проектной документации, отсутствие элементов интерфейса)

Особенности тестирования

- Тест планы в виде сценариев работы пользователя
- Сценарии на естественном языке или в виде скриптов
- Выполнение тестов в ручном режиме или с помощью эмулятора
- Анализ экранных форм и видимых элементов а не внутренних переменных
- Покрытие — участие интерфейсных элементов в тестах
- Найденный дефект: несоответствие реального поведения требованиям или проблемы в требованиях
- Проблема отчета об ошибке: расплывчатость формулировок

Типы требований к UI

- Требования к внешнему виду пользовательского интерфейса и формам взаимодействия с пользователем
 - Требования к размещению элементов управления на экранных формах
 - Требования к содержанию и оформлению выводимых сообщений
 - Требования к форматам ввода
- Требования по доступу к внутренней функциональности системы при помощи пользовательского интерфейса
 - Требования к реакции системы на ввод пользователя
 - Требования к времени отклика на команды пользователя

Функциональное тестирование UI

- 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

Оценка покрытия UI

- **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 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.

Ручное тестирование

- Плюсы:
 - Контроль корректности проводится человеком
 - Поиск «косметических» дефектов
 - Анализ успешности прохождения теста будет выполняться не по формальным признакам, а согласно человеческому восприятию
- Минусы:
 - Требуются значительные человеческие и временные ресурсы
 - При проведении регрессионного тестирования и вообще любого повторного тестирования - на каждой итерации повторного тестирования пользовательского интерфейса требуется участие тестировщика-оператора

Автоматическое тестирование

- Плюсы:
 - Снижение стоимости тестирования
 - Высокая скорость выполнения
 - Большой объем покрытия
 - Не требуется участие оператора-тестировщика при проведении регрессионного тестирования или любого другого повторного тестирования продукта
- Минусы:
 - Анализ успешности прохождения теста будет выполняться по формальным признакам
 - Невозможность поиска «косметических» дефектов
 - Высокая стоимость поддержки по сравнению с «обычными» функциональными тестами

Автоматизация тестирования UI

- Координатный метод

```
click(120,70)
```

```
type_keys("Hello world")
```

- Распознавание образов

```
find("OK_button.png").click()
```

- Подход, использующий механизмы реализации специальных возможностей (accessibility) и особенности реализации некоторых GUI фреймворков

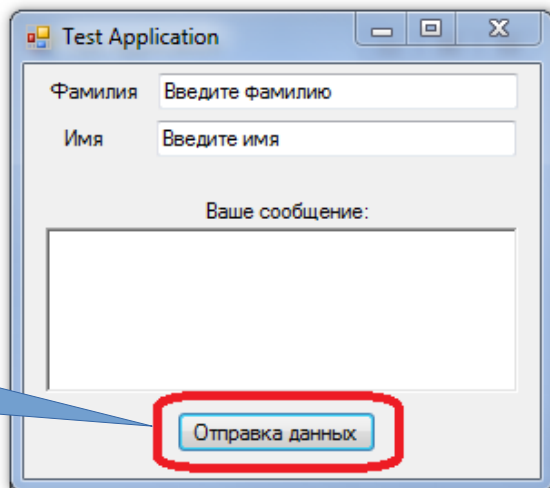
```
dlg.ColorComboBox.select("Green")
```

- Гибридный метод

Координатный метод

- Каждый элемент графического интерфейса пользователя ищется по координатам, заданным относительно окна или экрана
- Плюсы:
 - Быстрая разработка тестов
 - Простота поиска элементов при неизменности условий
- Минусы:
 - Крайне дорогая поддержка тестов
 - Зависимость тестов от настроек платформы (разрешение, шрифты, DPI, ...)
 - Невозможность отслеживания состояний объекта (активна кнопка или нет, установлен флажок у checkbox'а или не установлен, и т.д.)

Координатный метод

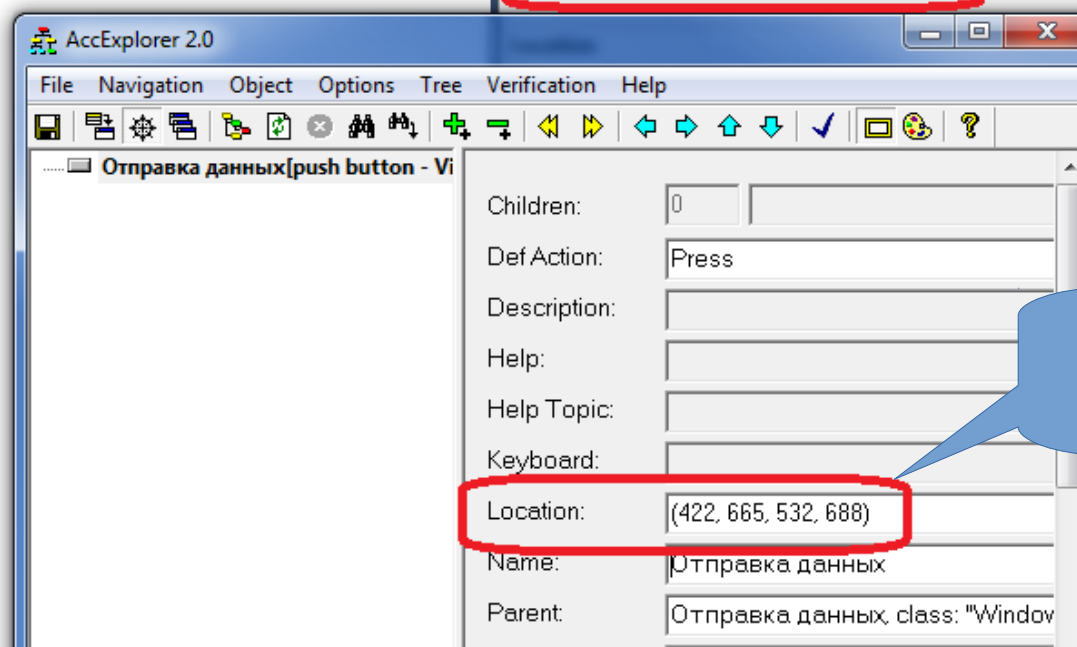


Кнопка

submitDataButton System.Windows.Forms.Button	
Anchor	Top, Left
AutoSize	False
AutoSizeMode	GrowOnly
Dock	None
Location	84, 191
X	84
Y	191
Margin	3, 3, 3, 3
MaximumSize	0, 0
MinimumSize	0, 0
Padding	0, 0, 0, 0
Size	110, 23

Относительные
координаты
кнопки

Размер
кнопки



Абсолютные
координаты

Координатный метод

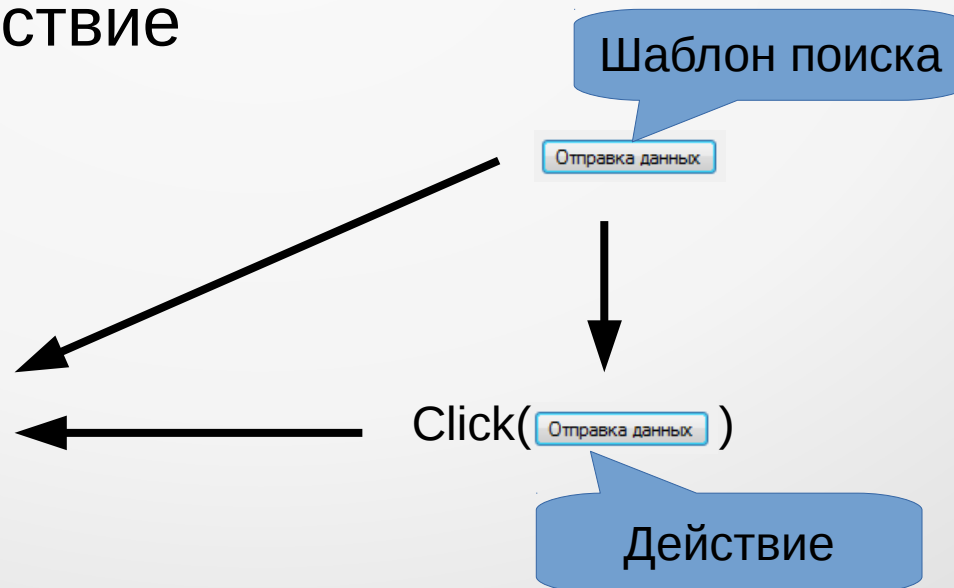
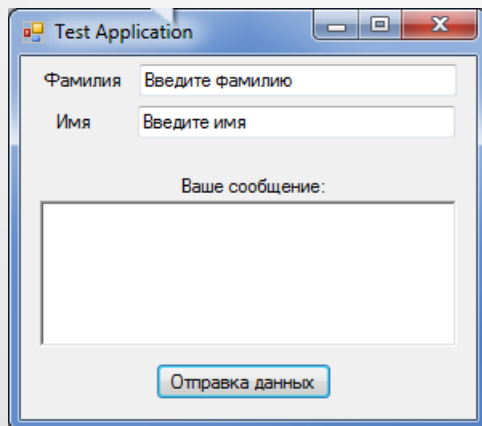
```
def main():
    waitForObject(":_QWidget")
    sendEvent("QResizeEvent", ":_QWidget", 22, 22, 769, 474)
    waitForObject(":_QGraphicsItem")
    mouseClick(":_QGraphicsItem", 221, 193, 1, Qt.LeftButton)
    waitForObject(":_QLineEdit")
    dragItemBy(":_QLineEdit", 153, -191, 26, 198, 1, Qt.LeftButton)
    waitForObject(":_QLineEdit")
    sendEvent("QMouseEvent", ":_QLineEdit", QEvent.MouseButtonRelease, 179, 7, Qt.LeftButton, 1)
    waitForObject(":MDC: Авторизация_QGraphicsView")
    type(":MDC: Авторизация_QGraphicsView", "<Ctrl+A>")
    waitForObject(":MDC: Авторизация_QGraphicsView")
    type(":MDC: Авторизация_QGraphicsView", "squish@mail.ru")
    waitForObject(":MDC: Авторизация_QGraphicsView")
    type(":MDC: Авторизация_QGraphicsView", "<Tab>")
    waitForObject(":MDC: Авторизация_QGraphicsView")
    type(":MDC: Авторизация_QGraphicsView", "1234")
    waitForObject(":MDC: Авторизация_CStartupWidget")
    sendEvent("QMoveEvent", ":MDC: Авторизация_CStartupWidget", 577, 70, 734, 62)
    mouseClick(":MDC: Авторизация_QWidget", 66, 372, 1, Qt.LeftButton)
    waitForObject(":MDC v1.0.3.1.nightly_CContactListWidget")
```

Распознавание образов

- Метод поиска элементов UI с использованием распознавания образов и (или) сравнение с образцом
- Плюсы:
 - Быстрая разработка тестов
 - Простота поиска элементов при неизменности условий
- Минусы:
 - Крайне дорогая поддержка тестов
 - Зависимость тестов от настроек платформы (разрешение, шрифты, DPI, ...)
 - Низкая «интеллектуальность» тестов

Распознавание образов

- Пример теста, построенного на системе распознавания образов:
 - Запустить приложение
 - Найти главное окно приложения
 - Найти требуемый элемент, сравнив с шаблоном
 - Произвести действие



Accessibility метод

- Метод управления UI через встроенные в ОС/SDK технологии воздействия на элементы UI
- Технологии
 - Microsoft Active Accessibility (MSAA)
 - Assistive Technologies Service Provider Interface (AT-SPI)
- Плюсы
 - более тесное взаимодействие с элементами UI
 - независимость от параметров отображения
 - Близкий к кодированию язык
- Минусы
 - не всегда есть поддержка
 - слабая кроссплатформенность

Accessibility метод



Accessibility технологии

Win32 API

MFC, VCL, частично WinForms
(Spy++)

MS UI Automation

WinForms, WPF, Qt, браузеры,
Store apps (Inspect.exe)

AT SPI (через Dbus)

Qt, GTK, wxWidgets, ...

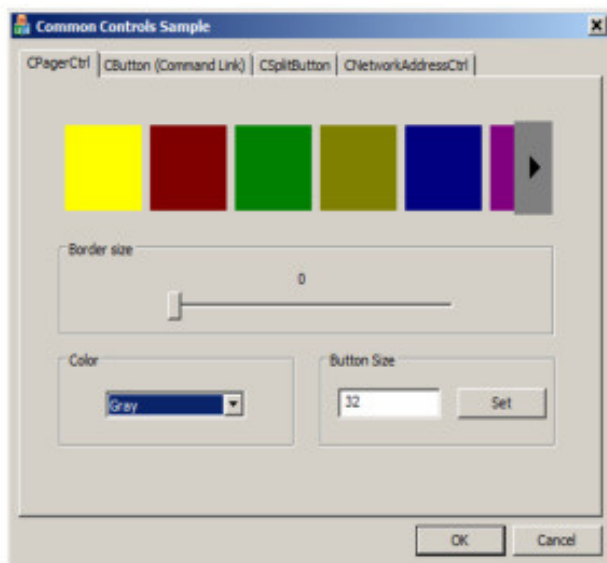
Apple Accessibility API

Cocoa, ...

Accessibility метод



Пример скрипта на pywinauto



```
app = Application().start("sample.exe")
```

```
dlg = app.CommonControlsSample
```

```
dlg.ColorComboBox.Select("Green")
```

```
dlg.ButtonSizeEdit.SetText("64")
```

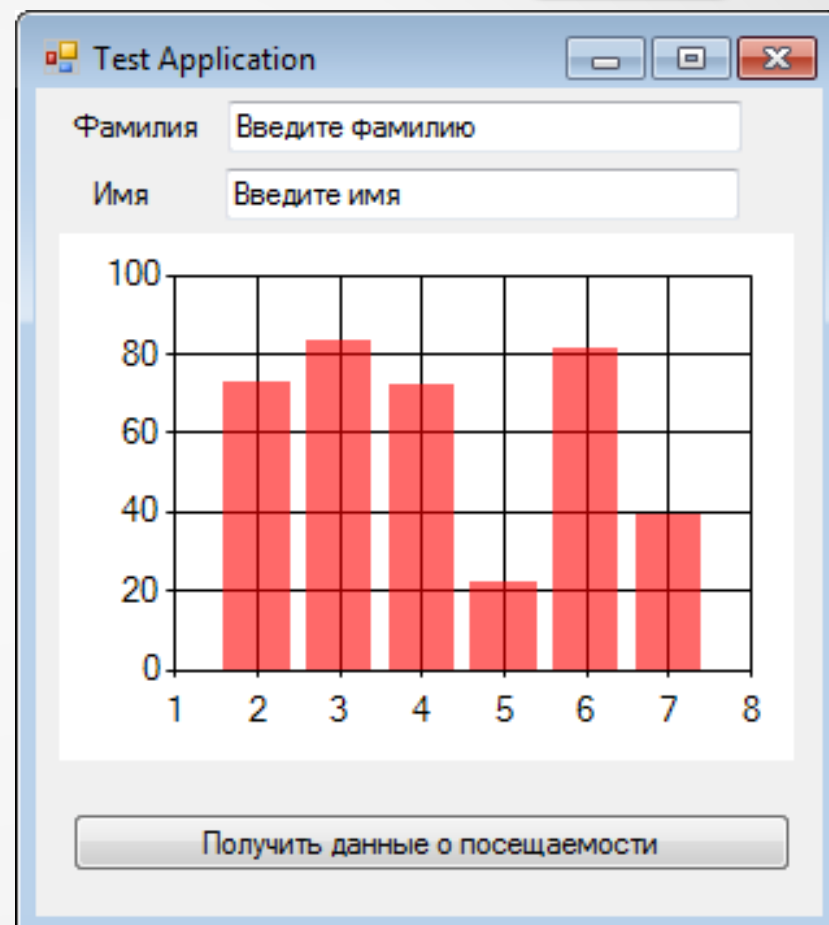
```
dlg.Set.Click()
```

```
dlg.OK.Click()
```

```
dlg.WaitNot("visible")
```

Гибридный подход

- Использование комбинации методов для улучшения результатов
- Пример: управление элементами формы через accessibility или координатный методы, анализ диаграммы через распознавание образов



Инструментарий

- Windows
 - Coded UI (Visual Studio)
 - Ranorex Automation Tools
 - Winium.Cruciatius (2GIS)
 - Microsoft UI Automation (MSAA)
 - Autolt
- Android
 - Espresso (Google)
 - Robotium
 - UI Automator
 - Monkey testing
- Web
 - Selenium
- Универсальные
 - Robot Framework

Selenium

The screenshot displays the Selenium IDE interface within a browser window. The browser title is "moz-extension://fddd816d-1c6d-485c-933f-54e69ba6d16b - Katalon Automatio...". The interface includes a toolbar with "New", "Record", "Play", and "Export" buttons. Below the toolbar is a table of test cases:

Test Cases	Command	Target	Value
Untitled Test Suite *	open	https://www.google.com.ar /?gws_rd=ssl	
Untitled Test Case *	click	id=lst-ib	
	type	id=lst-ib	test
	sendKeys	id=lst-ib	\${KEY_ENTER}
	click	link=test — Википедия	

Below the table are buttons for "Add", "Delete", "Delete All", "Execute", "Select", and "Find". A configuration section for the selected test case shows:

- Command: click
- Target: id=hplogo
- Value: (empty)

At the bottom left, the status shows "P: 0" and "F: 0". The bottom section is titled "Reference" and contains the following text:

click(locator)
Arguments:
• locator - an element locator
Clicks on a link, button, checkbox or radio button. If the click action causes a new page to load (like a link usually does), call `waitForPageToLoad`.

Selenium

[info] Executing: | open | https://www.google.com.ar/?gws_rd=ssl | |

[info] Wait for the new page to be fully loaded

[info] Executing: | click | id=lst-ib | |

[info] Executing: | type | id=lst-ib | test |

[info] Wait for all ajax requests to be done

[info] Executing: | sendKeys | id=lst-ib | \${KEY_ENTER} |

[info] Wait for the new page to be fully loaded

[info] Executing: | click | link=test — Википедия | |

[info] Test case passed

Тестирование удобства использования (Usability testing)

- Исследование, выполняемое с целью определения, удобен ли некоторый искусственный объект для его предполагаемого применения
- Цели:
 - Выявление сильных и слабых мест в интерфейсе для дальнейшего улучшения его в ходе итерационного процесса разработки
 - Оценка общего качества интерфейса – например, для выбора одного из двух возможных вариантов

Тестирование удобства использования (Usability testing)

- Этапы тестирования
 - Исследовательское – проводится после формулирования требований к системе и разработки прототипа интерфейса
 - Оценочное – проводится после разработки низкоуровневых требований и детализированного прототипа пользовательского интерфейса. Оценочное тестирование углубляет исследовательское и имеет ту же цель
 - Валидационное – проводится ближе к этапу завершения разработки. На этом этапе проводится анализ соответствия интерфейса программной системы стандартам, регламентирующим вопросы удобства интерфейса, проводится общее тестирование всех компонентов пользовательского интерфейса с точки зрения конечного пользователя
 - Сравнительное – данный вид тестирования может проводиться на любом этапе разработки интерфейса.

Тестирование удобства использования (Usability testing)

- Механизмы оценки
 - Производительность, эффективность (efficiency) – сколько времени и шагов понадобится пользователю для завершения основных задач приложения, например, размещение новости, регистрации, покупка и т.д
 - Правильность (accuracy) – сколько ошибок сделал пользователь во время работы с приложением
 - Активизация в памяти (recall) – как много пользователь помнит о работе приложения после приостановки работы с ним на длительный период времени? (повторное выполнение операций после перерыва должно проходить быстрее, чем у нового пользователя)
 - Эмоциональная реакция (emotional response) – как пользователь себя чувствует после завершения задачи – растерян, испытал стресс. Посоветует ли пользователь систему своим друзьям

Тестирование удобства использования (Usability testing)

- Эвристики Якоба Нильсена
http://www.useit.com/papers/heuristic/heuristic_list.html
 - Наблюдаемость состояния системы
 - Соотнесение с реальным миром
 - Пользовательское управление и свобода действий
 - Целостность и стандарты
 - Помощь пользователям в распознавании, диагностике и устранении ошибок
 - Предотвращение ошибок
 - Распознавание, а не вспоминание
 - Гибкость и эффективность использования
 - Эстетичный и минимально необходимый дизайн
 - Помощь и документация

Тестирование удобства использования (Usability testing)

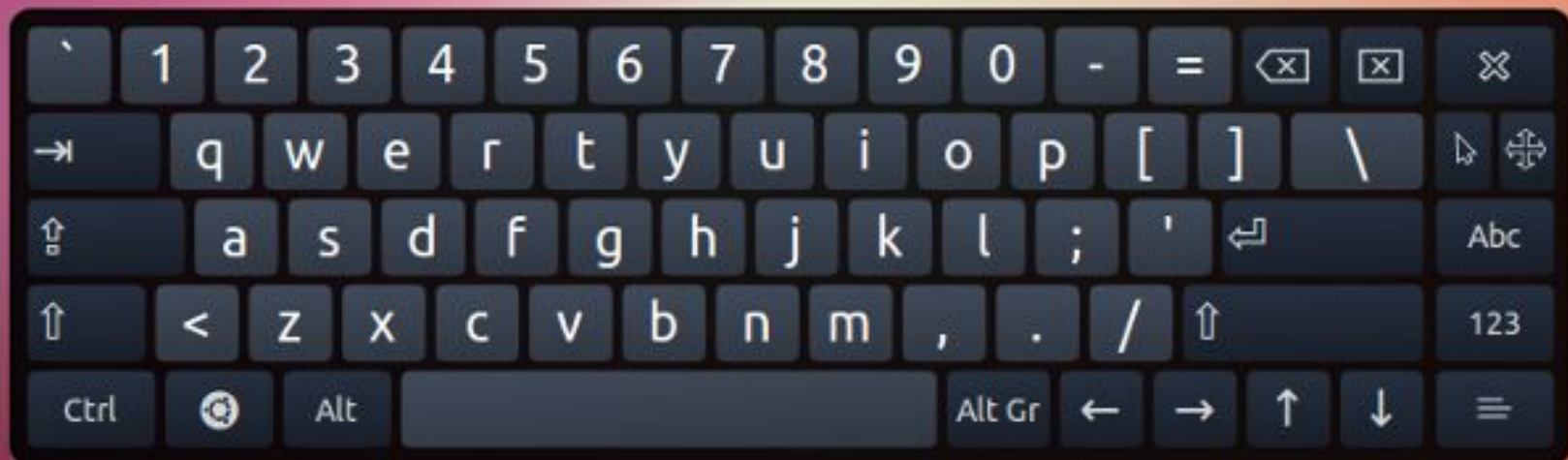
- Стадии теста
 - Подготовка – подготовка и проверка оборудования и места тестирования, начального состояния системы, наличия всех необходимых материалов – задания, мануалов и т.п.
 - Введение – приветствие участника, пояснение цели теста и процесса, предупреждение о записи действий и слов участника и т.д.
 - Непосредственно тестирование – роль проводящего тестирование минимальна, говорить («думать вслух») и делать должен участник
 - «Разбор полетов» – вопросы участнику на тему субъективного удовлетворения от работы, замечаний и предложений, пояснения каких-либо действий участника в ходе теста

Тестирование специальных возможностей (Accessibility testing)

- Проверка возможности использования программного обеспечения людьми с ограниченными возможностями, такими как:
 - Нарушения зрения
 - Нарушения подвижности
 - Нарушения слуха
 - Нарушения когнитивной функции

Тестирование специальных возможностей (Accessibility testing)

- Инструменты
 - Экранный диктор
 - Экранная лупа
 - Распознавание речи
 - Экранная клавиатура



Тестирование специальных возможностей (Accessibility testing)

- Примеры тестов
 - Проверка правильной работы пользовательского интерфейса на дисплее с высоким DPI
 - Проверка правильной работы пользовательского интерфейса в высококонтрастном режиме
 - Проверка правильной работы пользовательского интерфейса при использовании экранной лупы
 - Проверка доступности с клавиатуры элементов управления пользовательского интерфейса
 - Проверка пользовательского интерфейса с использованием специализированного ПО