

*Цель:* изучить классификацию видов тестирования, разработать проверки для различных видов тестирования, научиться планировать тестовые активности в зависимости от особенностей поставляемой на тестирование функциональности.

1. **Объект тестирования:** нож

2. Таблица 1.1

Вид тестирования	Краткое определение вида тестирования	Тестовые проверки
Functional Testing	тестирование, основанное на сравнительном анализе спецификации и функциональности компонента или системы.	<ul style="list-style-type: none"> <li>● Режет ли?</li> <li>● Можно заточить?</li> </ul>
Safety Testing	тестирование программного продукта с целью определить его способность при использовании оговоренным образом оставаться в рамках приемлемого риска причинения вреда здоровью, бизнесу, программам, собственности или окружающей среде.	<ul style="list-style-type: none"> <li>● Не порежусь ли?</li> <li>● Не умру?</li> </ul>
Security Testing	тестирование с целью оценить защищенность программного продукта от внешних воздействий (от проникновений). На практике зачастую под термином тестирование безопасности понимают в том числе и тестирование защищенности.	<ul style="list-style-type: none"> <li>● Не сломается при работе?</li> <li>● Не погнется ли?</li> <li>● Не затупится?</li> </ul>

Compatibility Testing	Проверка работоспособности приложения в различных средах .	<ul style="list-style-type: none"> <li>● Не заржавеет ли в воде?</li> <li>● Не испортится ли под воздействием низкой/высокой температуры?</li> </ul>
GUI Testing	тестирование, выполняемое путем взаимодействия с системой через графический интерфейс пользователя (правописание выводимой информации; расположение и выравнивание элементов GUI; соответствие названий форм/элементов GUI их назначению; унификация стиля, цвета, шрифта; окна сообщений; изменение размеров окна, поведение курсора и горячие клавиши).	<ul style="list-style-type: none"> <li>● Какого цвета?</li> <li>● Какой формы?</li> <li>● Есть ли рисунки/надписи?</li> </ul>
Usability Testing	тестирование с целью определения степени понятности, легкости в изучении и использовании, привлекательности программного продукта для пользователя при условии использования в заданных условиях эксплуатации (на этом уровне обращают внимание на визуальное оформление, навигацию, логичность, наличие обратной связи и др.).	<ul style="list-style-type: none"> <li>● Удобно ли им резать?</li> <li>● Удобно ли его держать?</li> </ul>
Accessibility Testing	тестирование, которое определяет степень	<ul style="list-style-type: none"> <li>● Возможно ли резать при</li> </ul>

	легкости, с которой пользователи с ограниченными способностями могут использовать систему или ее компоненты.	условии неполной работоспособности кисти?
Internationalization Testing	тестирование адаптации продукта к языковым и культурным особенностям целого ряда регионов, в которых потенциально может использоваться продукт.	<ul style="list-style-type: none"> <li>● Народу какой страны принадлежит нож данной формы?</li> <li>● Не противоречит ли нож культуре народа страны?</li> </ul>
Performance Testing	процесс тестирования с целью определения производительности программного продукта. В рамках тестирования производительности выделяют нагрузочное тестирование, объемное тестирование, тестирование стабильности и надежности, стрессовое тестирование.	<ul style="list-style-type: none"> <li>● Разрежет кость?</li> <li>● Не застрянет в вязкой пище?</li> <li>● Сколько прослужит?</li> </ul>
Stress Testing	вид тестирования производительности, оценивающий систему или компонент на граничных значениях рабочих нагрузок, или за их пределами, или же в состоянии ограниченных ресурсов, таких как память или доступ к серверу.	<ul style="list-style-type: none"> <li>● Кинуть в стену</li> <li>● Постучать по твердому предмету</li> <li>● Попытаться разрезать камень</li> </ul>
Negative Testing	тестирование на данных или сценариях, которые соответствуют нештатному поведению тестируемой	<ul style="list-style-type: none"> <li>● Разрежет ли камень?</li> <li>● Если нагреть лезвие и</li> </ul>

	системы - различные сообщения об ошибках, исключительные ситуации, "запредельные" состояния и т.д.	попытаться разрезать что-то твердое, не деформируется?
Black Box Testing	тестирование системы без знания внутренней структуры и компонентов системы (у тестировщика нет доступа к внутренней структуре и коду приложения либо в процессе тестирования он не обращается к ним).	<ul style="list-style-type: none"> <li>● Неизвестно, как работает</li> <li>● Неизвестно, как устроен</li> <li>● Проверка состава ножа под микроскопом</li> </ul>
Automated Testing	набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования. Тест-кейсы частично или полностью выполняет специальное инструментальное средство.	<ul style="list-style-type: none"> <li>● Создать робота, научить работать с ножом, дать ему материал для того, чтобы тестировал нож</li> </ul>
Unit/Component Testing	тестируются отдельные части (модули) системы.	<ul style="list-style-type: none"> <li>● Рукоять плотно обхватывается рукой?</li> <li>● Лезвие достаточной длины для разрезания пищи?</li> </ul>
Integration Testing	тестируется взаимодействие между отдельными модулями.	<ul style="list-style-type: none"> <li>● Собрали нож, прочно ли скреплены лезвие и рукоять?</li> <li>● Не отвалятся ли они друг от друга в процессе работы?</li> </ul>

**3. Разработать композицию тестов для первой поставки программного обеспечения (build 1), состоящей из трех модулей (модуль 1, модуль 2, модуль 3).**

Поверхностное тестирование (Smoke Test) выполняется для определения пригодности сборки для дальнейшего тестирования; необходимо полное тестирование системы как на корректных, так и на некорректных данных/сценариях (Acceptance Test, AT) позволяет обнаружить дефекты и внести запись о них в багтрекинг-систему, поэтому при первой поставке будет также использоваться New Feature Test.

$NFT_{AT(1,2,3)} + SMOKE$

**4. Разработать композицию тестов для второй поставки программного обеспечения (build 2): исправлены заведенные дефекты, доставлена новая функциональность – модуль 4.**

В данную поставку необходимо включить Defect Validation, так как были исправлены заведенные дефекты. При помощи  $NFT_{AT}$  проверим новую функциональность и при помощи Regressive Test(RT) проверим работу старой функциональности на позитивных сценариях(Minimal Acceptance Test, MAT)

$DV + NFT_{AT(4)} + RT_{MAT(1,2,3)} + SMOKE$

**5. Разработать композицию тестов для третьей поставки программного обеспечения (build 3): заказчик решил расширить рынки сбыта и просит осуществить поддержку программного обеспечения на английском языке.**

В данной поставке после включения в продукт английского языка необходимо проведение Internationalization Test. Кроме того перепроверим старые модули на корректную работу при помощи позитивных сценариев. Итого:

$SMOKE + IT + LT$

**6. Разработать композицию тестов для четвертой поставки программного обеспечения (build 4): заказчик хочет убедиться, что программное обеспечение выдержит нагрузку в 2000 пользователей.**

Для тестирования ПО устойчивости при большом потоке посетителей используем Нагрузочное тестирование(Performance and Load Testing) - вид тестирования производительности, проводимый с целью оценки поведения компонента или системы при возрастающей нагрузке, например количестве параллельных пользователей и/или операций, а также определения, какую нагрузку может выдержать компонент или система.

$SMOKE + PLT$

**Вывод:**

В данной лабораторной работе мы познакомились с процессом тестирования, выяснили, для чего тестирование необходимо и что является его конечной целью. Мы познакомились с видами тестирования и на примере применили их к выбранному объекту тестирования.