**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

РЯБОВА ЕЛИЗАВЕТА ПАВЛОНА

**ЮЗАБИЛИТИ ТЕСТИРОВАНИЕ. ОЦЕНКА ОТТЕСТИРОВАННОСТИ ПРОЕКТА: МЕТРИКИ И МЕТОДИКА ИНТЕГРАЛЬНОЙ ОЦЕНКИ. АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ.**

реферат студента

1 курса 10 группы

Минск 2020

СОДЕРЖАНИЕ

Аннотация……………………………………………..……………………….....3

Введение…………………………………………………………………………..4

1. Понятие юзабилити тестирования…………………………………………….5

2 Оценка оттестированности проекта: метрики и методика интегральной оценки……………………………………………………………………………...8

3.Автомотихация тестрирования……………………………………………….13

Заключение……………………………………………………………………….15

Список литературы ............................................................................................. 16

**Реферат**

**Ключевые слова: ТЕСТИРОВАНИЕ КАЧЕСТВА ПРОГРАММНОГО ПРОДУКТА, ПОЛЬЗОВАТЕЛЬ, ДИЗАЙН, ПРОГРАММА, АВТОМАТИЗАЦИЯ, МЕТОДИКИ ИНТЕГРАЛЬНОЙ ОЦЕНКИ**

***Объектом исследований*** является проект, который нуждается в оценке.

***Целью работы*** является изучение методов юзабилити-тестирования, методики интегральной оценки и автоматизации тестирования.

***Методами исследования*** являлось изучение информации, доступной в интернет ресурсах и информации из печатных изданий

***Область применения*** юзабилити-инженерия и смежные дисциплины

**Введение**

Существующие на сегодняшний день методы тестирования ПО не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого ПО.

Такой процесс формальной проверки или верификации может доказать, что дефекты отсутствуют с точки зрения используемого метода. (То есть нет никакой возможности точно установить или гарантировать отсутствие дефектов в программном продукте с учётом человеческого фактора, присутствующего на всех этапах жизненного цикла ПО).

Сейчас очень важно, чтобы любой программный объект был удобен для пользователя. Тестированием в этой области занимается юзабилити тестирование.

В данном реферате я ввожу понятие юзабилити тестирования. Цель работы - расширить знания студентов о возможностях тестирования ПО. Задачи: сформулировать суть понятия юзабилити тестирования, выделить основные цели и задачи этого метода.

**Юзабилити-тестирование**

**Юзабилити-тестирование (*Проверка эргономичности*) —**

исследование, выполняемое с целью определения, удобен ли некоторый искусственный объект (такой как веб-страница, пользовательский интерфейс или устройство) для его предполагаемого применения. Таким образом, проверка эргономичности измеряет эргономичность объекта или системы. Проверка эргономичности сосредоточено на определённом объекте или небольшом наборе объектов, в то время как исследования взаимодействия человек-компьютер в целом — формулируют универсальные принципы.

Проверка эргономичности — метод оценки удобства продукта в использовании, основанный на привлечении пользователей в качестве тестировщиков, испытателей и суммировании полученных от них выводов.

При испытании многих продуктов пользователю предлагают в «лабораторных» условиях решить основные задачи, для выполнения которых этот продукт разрабатывался, и просят высказывать во время выполнения этих тестов свои замечания.

Процесс тестирования фиксируется в протоколе (логе) и/или на аудио- и видеоустройства — с целью последующего более детального анализа.

Если проверка эргономичности выявляет какие-либо трудности (например, сложности в понимании инструкций, выполнении действий или интерпретации ответов системы), то разработчики должны доработать продукт и повторить тестирование.

Наблюдение за тем, *как*люди взаимодействуют с продуктом, нередко позволяет найти для него более оптимальные решения. Если при

тестировании используется модератор, то его задача — держать респондента сфокусированным на задачах (но при этом не „помогать“ ему решать эти задачи).

Основную трудность после проведения процедуры проверки эргономичности нередко представляют большие объёмы и беспорядочность полученных данных. Поэтому для последующего анализа важно зафиксировать:

1.Речь модератора и респондента;

2.Выражение лица респондента (снимается на видеокамеру);

3.Изображение экрана компьютера, с которым работает респондент;

4.Различные события, происходящие на компьютере, связанные с действиями пользователя:

**▪**Перемещение курсора и нажатия на клавиши мыши;

**▪**Использование клавиатуры;

**▪**Переходы между экранами (браузера или другой программы).

Все эти потоки данных должны быть синхронизированы по таймкодам, чтобы при анализе их можно было бы соотносить между собой.

Наряду с модератором в тестировании нередко участвуют наблюдатели. По мере обнаружения проблем они делают свои заметки о ходе тестирования так, чтобы после можно было синхронизировать их с основной записью. В итоге каждый значимый фрагмент записи теста оказывается прокомментирован в заметках наблюдателя. В идеале ведущий (т.е. модератор) представляет разработчика, наблюдатели — заказчика (например издателя, дистрибьютора), а испытатели — конечного пользователя (например покупателя).

Кроме вышеизложенного существует еще один подход к проверке эргономичности: для решения задачи предложенной пользователю разрабатывается "идеальный" сценарий решения этой задачи. Как правило, это сценарий, на который ориентировался разработчик. При выполнении задачи пользователями регистрируются их отклонения от задуманного сценария для последующего анализа. После нескольких итераций доработки программы и последующего тестирования можно получить удовлетворительный с точки зрения пользователя интерфейс.

## **Какие виды юзабилити-тестирования бывают?**

Используя разные критерии классификации, можно назвать несколько видов тестирований. В своей практике мы применяем все из них.

В зависимости от степени участия модератора (UX-аналитика):

* ***модерируемое*** — модератор дает задания, наблюдает за ходом их выполнения, задает уточняющие вопросы;
* ***немодерируемое***— специализированный сервис дает задания, собирает метрики и обратную связь в автоматическом режиме, без участия модератора.

В зависимости от места расположения респондента:

* ***очное*** — респондент и модератор находятся в одном помещении, как правило, в лаборатории, и общаются непосредственно;
* ***удаленное*** — респондент участвует в тестировании из дома или со своего рабочего места. Как правило, это немодерируемое тестирование. Если требуется участие модератора, аналитик общается с респондентом по видеосвязи.

В зависимости от целей:

* ***эксплораторное***— проводится на стадии разработки концепции интерфейса, чтобы проверить, насколько в целом концепция понятна пользователям, соответствует их потребностям и ожиданиям;
* ***проверочное***— проводится, чтобы найти и исправить юзабилити-проблемы или оценить показатели эффективности готового продукта или его макета (время, затраченное на выполнение задач, удовлетворенность пользователей и т.п.);
* ***сравнительное*** — проводится, чтобы сравнить эффективность новой и старой версий или двух конкурирующих продуктов; требует участия большого количества респондентов, чтобы можно было делать статистически значимые выводы.

**ОЦЕНКА ОТТЕСТИРОВАННОСТИ ПРОЕКТА: МЕТРИКИ И МЕТОДИКА ИНТЕГРАЛЬНОЙ ОЦЕНКИ.**

### Оценка Покрытия Программы и Проекта

*Тестирование программы* Р по некоторому критерию С означает покрытие *множества* компонентов программы P М = {m1...mk}по элементам или по связям

T = {t1...tn} - *кортеж* неизбыточных тестов ti.

Тест ti неизбыточен, если существует покрытый им *компонент* mi из M(P,C), не покрытый ни одним из предыдущих тестов t1...ti-1. Каждому ti соответствует неизбыточный *путь* pi - последовательность вершин от входа до выхода.

V(P,C) - *сложность тестирования* Р по критерию С - измеряется *max* числом неизбыточных тестов, покрывающих все элементы *множества* M(P,C)

*DV*(P,C,Т) - остаточная *сложность тестирования* Р по критерию С - измеряется *max* числом неизбыточных тестов, покрывающих элементы *множества* M(P,C), оставшиеся непокрытыми, после прогона набора тестов Т. Величина *DV* строго и монотонно убывает от V до 0.

TV(P,C,Т) = (V-*DV*)/V - оценка *степени тестированности* Р по критерию С.

Критерий окончания тестирования TV(P,C,Т) >= L, где (0 <= L <= 1). L - уровень оттестированности, заданный в требованиях к программному продукту.

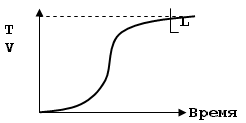


Рис. 4.1. Метрика оттестированности приложения

Рассмотрим две модели программного обеспечения, используемые при оценке оттестированности.

Для оценки *степени оттестированности* часто используется *УГП* - *управляющий граф программы*. *УГП* многокомпонентного объекта G(рис 4.2), содержит внутри себя два компонента G1 и G2, *УГП* которых раскрыты.

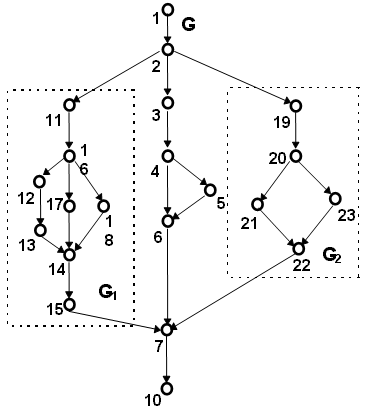


Рис. 4.2. Плоская модель УГП компонента G

В результате *УГП* компонента G имеет такой вид, как если бы компоненты G1 и G2 в его структуре специально не выделялись, а *УГП* компонентов G1 и G2 были вставлены в *УГП* G. Для тестирования компонента G в соответствии с критерием путей потребуется прогнать *тестовый набор*, покрывающий следующий набор трасс *графа* G(пример 4.1):

P1(G) = 1-2-3-4-5-6-7-10;

P2(G) = 1-2-3-4-6-7-10;

P3(G) = 1-2-11-16-18-14-15-7-10;

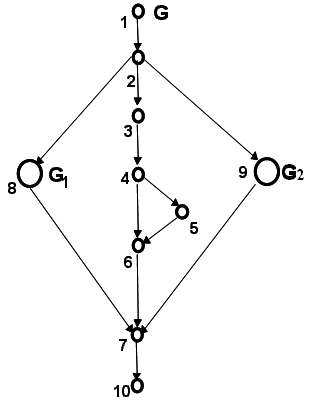
P4(G) = 1-2-11-16-17-14-15-7-10;

P5(G) = 1-2-11-16-12-13-14-15-7-10;

P6(G) = 1-2-19-20-23-22-7-10;

P7(G) = 1-2-19-20-21-22-7-10;

4.1. Набор трасс, необходимых для покрытия плоской модели УГП компонента G



P1(G) = 1-2-3-4-5-6-7-10;

P2(G) = 1-2-3-4-6-7-10;

P3(G) = 1-2-11-16-18-14-15-7-10;

P4(G) = 1-2-11-16-17-14-15-7-10;

P5(G) = 1-2-11-16-12-13-14-15-7-10;

P6(G) = 1-2-19-20-23-22-7-10;

P7(G) = 1-2-19-20-21-22-7-10;

4 .1. Набор трасс, необходимых для покрытия плоской модели УГП компонента G

Рис. 4.3. Иерархическая модель УГП компонента G

*УГП* компонента G, представленный в виде иерархической модели, приведен на В иерархическом *УГП* Gвходящие в его состав компоненты представлены ссылками на свои *УГП* G1 и G2

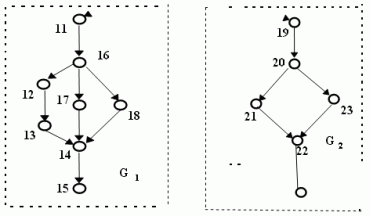


Рис. 4.4. Иерархическая модель: УГП компонент G1 и G2

Для исчерпывающего тестирования иерархической модели компонента G в соответствии с критерием путей требуется прогнать следующий набор трасс ([Пример 4.2](https://www.intuit.ru/studies/courses/48/48/lecture/1430?page=1#example.4.2)):

P1(G) = 1-2-3-4-5-6-7-10;

P2(G) = 1-2-3-4-6-7-10;

P3(G) = 1-2-8-7-10;

P4(G) = 1-2-9-7-10.

4.2. Набор трасс, необходимых для покрытия иерархической модели УГП компонента G

Приведенный набор трасс достаточен при условии, что компоненты G1 и G2 в свою *очередь* исчерпывающе протестированы. Чтобы обеспечить выполнение этого условия в соответствии с критерием путей, надо прогнать все трассы [Пример 4.3](https://www.intuit.ru/studies/courses/48/48/lecture/1430?page=1#example.4.3).

P11(G1)=11-16-12-13-14-15;

P12(G1)=11-16-17-14-15;

P13(G1)=19-20-23-22;

P21(G2)=19-20-21-22;

P22(G2)=11-16-18-14-15.

4.3. Набор трасс иерархической модели УГП, необходимых для покрытия УГП компонентов G1 и G2

Оценка *степени тестированности* *плоской модели* определяется долей прогнанных трасс из набора необходимых для покрытия в соответствии с критерием С.

(1) TV(G,С) = (V-DV)/V = \sum PT_{i}(G) / (\sum P_{i}(G)),

где PTi(G) - тестовый *путь* ( ti ) в *графе* G *плоской модели* равен 1, если он протестирован (прогнан), или 0, если нет.

Например, если в *УГП* ([Пример 4.1](https://www.intuit.ru/studies/courses/48/48/lecture/1430?page=1#example.4.1)) тесты t6 и t7, которым соответствуют трассы P6 и P8, не прогнаны, то в соответствии с соотношением (1) для TV(G,С) *степень тестированности* будет оценена в 0.71.

Оценка тестированности иерархической модели определяется на основе учета оценок тестированности компонентов. Если трасса некоторого теста tj*УГП* G включает узлы, представляющие компоненты Gj1,..Gjm, оценка TV *степени тестированности* которых известна, то оценка тестированности PTi(G) при реализации этой трассы определяется не 1, а минимальной из оценок TV для компонентов.

Интегральная оценка определяется соотношением (2):

(2) TV(G,C) = (V-DV)/V = (\sum PT_{i}(G) * \sum (TV(G_{ij},C))) / (\sum P_{i}(G))

где PTi(G) - тестовый *путь* ( ti ) в *графе* G равен 1, если протестирован, или 0, если нет. В *путь* PTi*графа* G может входить j узлов модулей Gij со своей *степенью тестированности* TV(Gij,С), из которых мы берем *min*, что дает худшую оценку степени тестированности пути.

### Методика интегральной оценки тестированности

1. Выбор критерия С и приемочной оценки тестированности программного проекта - L
2. Построение древа классов проекта и построение *УГП* для каждого модуля
3. Модульное тестирование и оценка TV на модульном уровне
4. Построение *УГП*, интегрирующего модули в единую иерархическую ( *классовую* ) модель проекта
5. Выбор тестовых путей для проведения интеграционного или системного тестирования
6. Генерация тестов, покрывающих тестовые пути шага 5
7. Интегральная оценка тестированности проекта с учетом оценок тестированности модулей-компонентов
8. Повторение шагов 5-7 до достижения заданного уровня тестированности L

**Автоматизация тестирования**

**Автоматизация тестирования** – это процесс проверки программного обеспечения, который включает проведение таких основных функций и шагов теста, как запуск, инициализация, выполнение, анализ и выдача результата, автоматически посредством специализированных инструментов.

Автоматизированное тестирование – аналог ручного тестирования , который выполняется программой-роботом, а не человеком.

Ключевые преимущества

1.Повышение качества тестирования, поскольку при использовании средств автоматизации «человеческий фактор» не оказывает влияния на качество тестирования.

2.Возможность выполнения таких видов тестирования, которые либо не могут быть выполнены вручную, либо требуют значительных затрат (дополнительное оборудование, персонал).

3.Ускорение процесса тестирования без потери качества. Проведение того же объема работ ручным методом занимает больше времени. Использование средств автоматизации для тестирования позволяет запускать уже написанные скрипты без дальнейших доработок.

4.В ходе тестирования отчеты о результатах работы программных приложений рассылаются и сохраняются автоматически.

**Автоматическое тестирование. Основные нюансы проверки качества программного обеспечения (ПО)**

Современное программное обеспечение является сложным многофункциональным объектом. Его ручная проверка требует значительных трудовых и временных затрат. На помощь приходят средства автоматизации тестирования, которые

* повышают качество,
* обеспечивают повторное использование тестов при корректировке ПО.

Процесс автоматизации тестирования – это интеллектуальное творчество ИТ-специалистов высокой квалификации, но для достижения поставленных целей его тоже необходимо вести планомерно.

Применение средств автоматизации для тестирования программного обеспечения актуально в следующих случаях:

* Запись в базу данных, логирование файлов, backend процессы, то есть места в системе, представляющие наибольшую труднодоступность.
* Достаточно часто используется функциональность, которая обладает высоким уровнем рисков на ошибки. При автоматизации критической функциональности посредством тестирования гарантированно обеспечивается быстрое нахождение ошибок.
* Автоматизированное тестирование эффективно для рутинных операций. Например, формы, в которых имеется большое количество полей для набора данных (перебор данных). Тестовый процесс позволяет автоматически выполнять заполнение полей, а также после сохранения осуществлять их проверку.
* Длинные сценарии (end-to-end).
* Тестирование данных, для которых необходимы точные математические расчеты.
* Для автоматизации правильности поиска данных.
* В зависимости от предъявляемых заказчиком требований и возможностей инструмента, автоматическому тестированию подлежат и другие программные продукты. Разработка отдельных тест-кейсов автоматизации делает процесс тестирования более эффективным. Создание начальных условий, минимально влияющих на другие тесты, позволяет использовать сценарии, способные вернуть систему к исходному состоянию. При тестировании с такой функциональностью от автоматизации обеспечивается максимальная отдача.

**Выводы**

В данном реферате:

\* Сформулировано понятие юзабилити-тестирования

\* Объяснены основные цели и задачи метода

\* Рассмотрена классификация методов тестирования

\* Раскрыто понятие и методы автоматизации тестирования

\* Раскрыт алгоритм работы юзабилити-тестирования

***СПИСОК ЛИТЕРАТУРЫ***

1. [Электрон. ресурс]: www.intuit.ru/studies/courses

2. [Электрон. ресурс]: Michael Felderer. Security Testing: A Survey/Matthias Bu¨chler, Martin Johns, Achim D. Brucker, Ruth Breu, Alexander Pretschner // researchgate. — 2016. — 12 март. — С. 22-26.

3. [Электрон. ресурс]: 33testers.blogspot.com/2015/06/blog-post\_17.html

4. [Электрон. ресурс]: testitquickly.com/2010/03/09/testing-basics-by-barancev

5. [Электрон. ресурс]: www.youtube.com/watch?v=LfL6YDiYsw4

6. [Электрон. Ресурс]: www.javenue.info/post/24