Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра Информатики
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.С. Кузнецов
«13» 06 2017 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Алгоритм определения оптимального набора инструментов тестирования программного обеспечения

09.04.04 «Программная инженерия»
код и наименование направления
09.04.04.01 «Программное обеспечение вычислительной техники и автоматизированных систем»
код и наименование магистерской программы

Научный руководитель _____ 9.06.17 доц. кафедры Информатики СФУ, к.т.н О.А.Антамошкин
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 9.06.17 А.С. Постовалова
подпись, дата инициалы, фамилия

Рецензент _____ 9.06.17 зав. кафедры ЭиИТМ СФУ, д.т.н А.А. Ступина
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2017

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра Информатики
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.С. Кузнецов
«____» _____ 2017 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Алгоритм определения оптимального набора инструментов тестирования программного обеспечения

09.04.04 «Программная инженерия»
код и наименование направления
09.04.04.01 «Программное обеспечение вычислительной техники и автоматизированных систем»
код и наименование магистерской программы

Научный руководитель _____ доц. кафедры Инорматики СФУ, к.т.н О.А.Антамошкин
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ А.С. Постовалова
подпись, дата инициалы, фамилия

Рецензент _____ зав. кафедры ЭиИТМ СФУ, д.т.н А.А. Ступина
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2017

РЕФЕРАТ

Магистерская диссертация по теме «Алгоритм определения оптимального набора инструментов тестирования программного обеспечения» содержит 62 страницы текстового документа, 4 иллюстрации, 1 таблицу, 9 формул, 31 использованный источник.

ТЕСТИРОВАНИЕ, ТЕСТ-КОМПЛЕКТ, ТЕСТ-АНАЛИЗ, АЛГОРИТМ, ОЦЕНКА ТЕСТИРОВАНИЯ, ТЕСТ-КЕЙС, ЧЕК-ЛИСТ, IT-ПРОЕКТ.

Цель работы: разработать алгоритм определения оптимального набора тест-комплекта IT-компании.

Задачи:

1. Исследование особенностей и классификация видов IT-проектов.

2. Анализ современных инструментов тестирования, позволяющих ускорить создание тестовой документации.

3. Разработка алгоритмов построения тест-комплекта, обеспечивающих минимизацию использования ресурсов разработчика.

4. Формальная постановка задачи формирования тест-комплекта.

5. Применение предлагаемых инструментов на реальных проектах.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Качество программного продукта определяется по нескольким критериям. Качественный программный продукт должен отвечать функциональным и нефункциональным требованиям, в соответствии с которыми он создавался, иметь ценность для бизнеса, отвечать ожиданиям пользователей [1].

В жизненном цикле управления приложениями качество должно отслеживаться на всех этапах жизненного цикла ПО. Оно начинает формироваться с определения необходимых требований. При задании требований необходимо указывать желаемую функциональность и способы проверки её достижения.

Качественный программный продукт должен обладать высоким потребительским качеством, независимо от области применения: внутреннее использование разработчиком, бизнес, наука и образование, медицина, коммерческие продажи, социальная сфера, развлечения, веб и др. Для пользователя программный продукт должен удовлетворять определенному уровню его потребностей.

Тестирование программного продукта позволяет на протяжении всего жизненного цикла ПО гарантировать, что программные проекты отвечают заданным параметрам качества. Главная цель тестирования - определить отклонения в реализации функциональных требований, обнаружить ошибки в выполнении программ и исправить их как можно раньше в процессе выполнения проекта.

На протяжении всего жизненного цикла разработки ПО применяются различные типы тестирования для гарантии того, что промежуточные версии отвечают заданным показателям качества. При этом применяются автоматические и ручные тесты.

Каждый специалист по качеству сталкивается с проблемой становления процесса тестирования. Особенно это наблюдается в молодых компаниях, стартапах.

Актуальность задачи формирования тест-комплекта объясняется тем, что становится существенной проблема оформления тестовых случаем. Данная проблема влияет на оценку тестирования, и в свою очередь на качество продукта.

Учитывая вышесказанное, новизна данной работы заключается в использовании алгоритма с пошаговым оцениванием времени на использование тех или иных инструментов для тестирования ПО.

Существенной проблемой становится оформление тест-кейсов и оценка времени тестирования приложения.

Объектом диссертационного исследования является процесс обеспечения качества продуктов IT-компании.

Предмет исследования – инструменты тестирования ПО.

Цель диссертационного исследования состоит в создании инструментария для определения оптимального набора тест-комплекта IT-проекта.

Поставленная цель достигается путем решения следующих задач:

1. Исследование особенностей и классификация видов IT-проектов.

2. Анализ современных инструментов тестирования, позволяющих ускорить создание тестовой документации.

3. Разработка алгоритмов построения тест-комплекта, обеспечивающих минимизацию использования ресурсов разработчика.

4. Формальная постановка задачи формирования тест-комплекта.

5. Применение предлагаемых инструментов на реальных проектах.

ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Исследование видов IT-проекта

Создание качественного IT-продукта – это, прежде всего, создание сложных систем взаимодействия между элементами продукта на всех этапах жизненного цикла проекта. Задача проектного подхода в таком случае – обеспечение такого взаимодействия на качественно высоком уровне с максимальным эффектом. Этот процесс значительно осложняется широким спектром направлений IT-проектов по своей основной целевой ориентации. Такая особенность подобных проектов наглядно демонстрирует необходимость подробного анализа и выделения классификационных особенностей.

В первую очередь, необходимо дать определение термину «IT-проект», основываясь на стандартном определении термина «проект». Тогда IT-проект – это временное предприятие, направленное на разработку уникального продукта, имеющее чётко определённый срок выполнения, ограничения по ресурсам, свои критерии качества и понятия об успешном завершения [3].

Согласно целевой направленности, IT-проект - это проект, в рамки которого входят работы, связанные с информационными технологиями, которые в свою очередь направлены на создание, развитие и поддержку информационных систем. Основываясь на том, что под информационными системами понимают комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства и информационные ресурсы, а также системный персонал и обеспечивающий поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей, становится очевидным, что IT-проекты являются комплексными, отличаются такими характеристиками как сложность, масштабность и разнообразие.

В связи с этим, можно выделить ряд особенностей IT-проектов, влияющих на формирование эффективной системы управления [16]:

- нестандартный жизненный цикл, который может включать в себя также тестовый, гарантийный и пост-гарантийный этапы разработки;

- необходимость четкого определения, уже на этапе инициации, требований к IT-проектам несмотря на подвижность и неоднозначность некоторых направлений в IT-сфере;

- необходимость оперативного внесения изменений на этапе тестирования, что создает сложности, с которыми сталкиваются практически все руководители IT-проектов, вследствие чего происходит отставание от запланированных сроков.

По статистическим данным 70% IT-проектов не укладываются в отведённые сроки, что приводит к превышению бюджета проекта, и, как следствие, невыполнению основных заявленных требований к проекту [15]. Для удобства анализа и синтеза проектов, а также систем управления ими множество разнообразных IT-проектов классифицируется по разным основаниям. В научной литературе встречаются разные подходы к классификации проектов. Наиболее часто применяемой классификацией является:

1. Класс. По составу и структуре проектов и его предметной области проекты разделяются на монопроекты, мультипроекты, мегапроекты.

2. Тип. По основным сферам деятельности, в которых осуществляется проект, выделяют технические, организационные, экономические, социальные и смешанные проекты.

3. Вид. По характеру предметной области проекты подразделяются на: инвестиционные, научно-исследовательские, учебно-образовательные, смешанные.

4. Масштаб. По размерам самого проекта, количеству участников и степени влияния на окружающий мир проекты делят на: мелкие, средние, крупные и очень крупные проекты.

5. Длительность. По продолжительности периода осуществления проекты подразделяются на: краткосрочные, среднесрочные и длительные проекты.

6. Сложность. По степени сложности: простые, сложные и очень сложные.

Использование указанной классификации не в полной мере даёт представление об отличиях IT-проектов от других видов проектов. Как следствие, возникает необходимость индивидуализации представленной классификационной системы [17].

Если рассматривать разработку продуктов для онлайн использования, то проекты можно разделить на:

1. Разработка интернет-представительства (сайт-визитка, интернет-витрины, промо-сайты).

2. Разработка информационных ресурсов (тематические сайты, блоги).

3. Разработка веб-сервисов (автоматизация процессов, социальные сети, интернет-магазины).

Все три вида имеют свою специфику. Основываясь на такое разделение, можно выделить следующие классификационные признаки IT-проектов:

1. По характеру изменений:

- простые (проект, осуществление изменений в структуре и содержании которых, не приводит к изменению его стоимостных и временных параметров);

- средние (проект, осуществление изменений в структуре и содержании которых, приводит к несущественному изменению его стоимостных и временных параметров);

- сложные (проект, осуществление изменений в структуре и содержании которых, приводит к существенному изменению его стоимостных и временных параметров).

2. По масштабу:

- малые (стоимостью до 250 тыс. руб);

- средние (стоимостью до 1 мил. руб);

- большие (стоимостью до 2 млн. руб);

- сверхбольшие (стоимостью свыше 2 млн. руб).

3. По длительности:

- Краткосрочные – длительностью до 6 мес;

- Среднесрочные – длительностью от 6 мес до 2-х лет;

- Долгосрочные – длительностью свыше 2-х лет.

4. По стадиям жизненного цикла системы:

- оформление замысла и концепции;

- формулирование требований к системе;

- разработка системы;

- введение системы в эксплуатацию;

- поддержка существующей системы.

5. По виду продукта:

- система;

- программный продукт;

- технические средства;

- программно-технические комплексы;

- материалы, работы и услуги.

6. По функциональному назначению (ориентированность продуктов IT-проектов на направления деятельности заказчика):

- производственные;

- технологические;

- финансовые;

- исследовательские;

- маркетинговые;

- по управлению персоналом;

- по управлению проектами;

- игровые;

- комбинированные.

7. По глубине взаимного проникновения

- бизнеса заказчика и подрядчика;
- аутсорсинг;
- решения «под ключ»;
- совместные проекты;
- сервисная модель;
- аудит и консалтинг.

8. По виду заказчика:

- Стартап;
- Гос.учереждение;
- Частный сектор.

9. По виду автоматизируемых процессов:

- основные и вспомогательные;
- технологические и офисные;
- управленческие;
- аналитические;
- транзакционные;
- реального времени;
- с тем или иным акцентом на вычислительную обработку;
- передача данных;
- организация хранения;
- обработка медиа-контента;
- обеспечение безопасности и т.п.

10. По степени сложности:

- монопроекты - отдельные проекты определенного вида;
- мультипроекты - комплексные проекты, состоящие из ряда монопроектов и требующие многогранного проектного управления;
- мегапроекты - целевые программы развития регионов, отраслей, включающие ряд моно- и мультипроектов.

11. По территориальному распространению:

- мононациональные – продукт проекта ориентирован на территориальные регионы со схожей ментальностью;

- полинациональные – продукт проекта ориентирован на территориальные регионы с разной ментальностью.

12. По уровню влияния разработки интерфейса на проект:

- низкий – результат разработки интерфейса имеет незначительное влияние на оценку проекта в целом, занимает небольшой отрезок времени в жизненном цикле проекта;

- средний – результат разработки интерфейса имеет значительное влияние на оценку проекта в целом, занимает значительный отрезок времени в жизненном цикле проекта;

- высокий − результат разработки интерфейса имеет критически важное влияние на оценку проекта в целом, занимает значительный отрезок времени в жизненном цикле проекта, может повлиять на принятие решения о закрытии проекта.

1.2 Методы проектирования и создания тест-кейсов

Тест дизайн – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования [30]. Test design [ISTQB Glossary of terms]: The process of transforming general testing objectives into tangible test conditions and test cases.

Тест-дизайн преследует такие цели как:

1. Разработать тесты, которые обнаружат наиболее серьезные ошибки продукта.

2. Минимизировать количество тестов, необходимых для нахождения большинства серьезных ошибок.

Проектирование тестов включает в себя различные техники - это рекомендации, советы и правила по которым стоит разрабатывать тест для проведения тестирования приложения [2]. Но тем не менее следование этим техникам приведет к более качественному тестирование ПО. Далее приведены базовые техники тест-дизайна:

1. Доменный анализ:
    a. Классы эквивалентности;
    b. Граничные значения.
2. Таблицы принятия решений.
3. Диаграмма состояний и переходов.
4. Тестирование парных значений.

1.2.1 Доменный анализ

Доменные анализ - является тестированием областей определения. Область определения — математический термин — это совокупность всех возможных значений переменной [5]. В доменном анализе программа рассматривается как функция многих переменных, каждая из которых принимает конечное множество значений. Каждое такое множество можно разбить как минимум на два класса эквивалентности — валидные и невалидные значения [6, 8].

Тестирование областей определения предполагает три шага:

- выделение подобластей для каждого параметра, все элементы которых предположительно приводят к одинаковому поведению программы (для сокращения количества тестов);

- выбор конкретных значений для тестирования внутри каждого класса (в т.ч. для выявления ошибок, связанных с тем, что область определения задана неверно);

- сочетание этих значений (для увеличения тестового покрытия и выявления ошибок, зависящих от взаимодействия нескольких параметров).

Классы эквивалентности

Теория разбиения на классы эквивалентности, предложенная Гленфордом Майерсом (Glenford Myers) [22], направлена на сокращение общего числа необходимых тестовых сценариев путем разбиения входных условий на конечное число классов эквивалентности. Создается два типа классов: допустимые входные данные программы рассматриваются как допустимый класс эквивалентности, а все остальные входные данные заносятся в недопустимый класс эквивалентности.

Несколько простых правил:

Если область определения параметра — диапазон, то имеет смысл выделение трех классов эквивалентности: слева от диапазона (невалидные

значения), сам диапазон (валидные значения) и справа от диапазона (снова невалидные). При выделении классов нужно использовать включающие границы с целью однозначности и точности: одно и то же значение не может относиться к двум классам одновременно.

Если область определения — набор неупорядоченных данных, то всегда можно выделить как минимум два класса — валидные и невалидные значения [9]. Полученное разбиение можно «дробить» дальше. Например, множество латинских букв можно разбить на два подмножества: латиница в верхнем и нижнем регистре соответственно.

В примере выше используется очевидный способ дробления на подклассы, но он не единственный. Такое разбиение не всегда целесообразно в том смысле, что с его помощью не так уж часто находятся ошибки. Вот некоторые другие приемы:

- по частоте использования конечными пользователями (например, для параметров типа логина и пароля может иметь смысл выделение в отдельный класс символов «qwertyQWERTY1234567980»);

- случайные равные по размеру подклассы (обеспечение условного тестового покрытия, если нет никаких других логических способов выполнить разбиение на подклассы).

Различают линейные (упорядоченные) и нелинейные (неупорядоченные) классы эквивалентности. Очевидно, к последним невозможно применить анализ граничных значений, т.е. нет логического способа выделить элементы, с большей вероятностью приводящие к ошибке. Примером такого класса может быть множество специальных символов, которые можно ввести с клавиатуры. Для дробления такого класса может пригодиться второй прием. Входным параметром для его применения будет количество подклассов, которые планируется использовать в тестировании: из каждого будет взято по одному значению.

Типичные ошибки этого этапа тестирования областей определения: слишком много или слишком мало классов, классы выделены неправильно (по отношению к функциональности программы).

Признаки эквивалентности тестов:

- направлены на поиск одной и той же ошибки;

- если один из тестов обнаруживает ошибку, другие скорее всего, тоже её обнаружат;

- если один из тестов не обнаруживает ошибку, другие, скорее всего, тоже её не обнаружат;

- тесты используют схожие наборы входных данных;

- для выполнения тестов мы совершаем одни и те же операции;

- тесты генерируют одинаковые выходные данные или приводят приложение в одно и то же состояние;

- все тесты приводят к срабатыванию одного и того же блока обработки ошибок;

- ни один из тестов не приводит к срабатыванию блока обработки ошибок.

Польза раскладывания значений ввода на эквивалентные классы состоит в том, что мы отсеиваем огромное количество значений ввода, использовать которые для тестирования просто бессмысленно.

Выбор значений

После того, как завершено разбиение на классы эквивалентности, необходимо выбрать значения из каждого класса, которые будут использоваться в тестах. Анализ граничных значений — только один из способов, причем подходящий только для линейных классов. Что же предпринять в других случаях?

Случайный выбор. Крайне желательно, чтобы при каждом следующем выполнении теста наугад выбиралось какое-то другое значение из класса. В этом случае можно применять случайный выбор с возвратом (значение,

выбранное ранее, может быть выбрано повторно с той же вероятностью) или без возврата (значение, выбранное ранее, не может быть больше выбрано, тем самым вероятность выбора оставшихся в классе значений увеличивается).

Пропорциональное разбиение. Есть разные алгоритмы, основная цель которых — уменьшить риск неправильного разбиения на классы эквивалентности. Для этого можно брать несколько значений из каждого класса (число увеличивается с увеличением полезности той или иной функции программы). Другой способ: взять из каждого класса не фиксированное количество значений, а фиксированную часть класса. Таким образом, для классов, содержащих больше элементов, получится больше тестов.

Анализ граничных значений. Следует помнить, что основная идея этой техники — выделение значений, приводящих к ошибкам с большей вероятностью, чем другие. Эта техника не ограничивается непосредственно элементами управления на экране программы. Кроме числовых границ диапазонов стоит помнить о временных границах (например, срок бесплатного пользования программой), границах циклов (количество неправильных вводов пароля), границах типов (даже если согласно спецификации, можно в некоторое поле можно ввести ничем не ограниченное сверху целое число, это число так или иначе будет ограничено — максимальным значением целочисленного типа данных, который выбрал программист в реализации этой функции). Есть и другие границы, связанные с нефункциональными видами тестирования — производительности, конфигураций.

Эмпирическое знание. Некоторые значения могут выбираться чаще других или предполагать особое использование с точки зрения бизнес-логики приложения. Идеи таких тестов может подсказать человек, хорошо ориентирующийся в предметной области программы.

При анализе граничных значений и выделении классов эквивалентности для числовых параметров следует уделить особенное внимание результату вычислений. Такое разбиение и выбор значений для тестирования поможет найти важные ошибки. Какие ограничения накладываются на область

значений, т.е. результат вычислений, и какие значения при этом должны принимать входные параметры? Какие нужно задать значения на вход, чтобы выйти за границы этой области? Например, если результат вычисления должен быть положительным, стоит выделить три класса эквивалентности и соответствующие им граничные значения:

- входные данные, при которых результат строго положительный (валидный класс);

- входные данные, при которых результат равен нулю (невалидный класс);

- входные данные, при которых результат отрицателен (невалидный класс).

Может оказаться, что значения, оказавшиеся при таком разбиении в невалидных классах, разрешены для ввода согласно спецификации. Это может быть ошибкой составления требований, на которую стоит указать бизнес-аналитикам.

1.2.2 Тестирование парных значений

Дефекты, зависящие от входных данных, можно поделить на те, которые возникают при конкретном значении одного параметра, и те, для возникновения которых нужно сочетание конкретных значений более чем одного параметра. Для обнаружения последних применяют комбинаторные техники тестирования, одной из которых является попарное тестирование (pairwise).

Комбинаторные техники можно применить, когда выделены классы эквивалентности для каждого параметра, и выбраны значения, на которых будут проводиться тесты для каждого параметра по отдельности [25]. Для составления комбинаций можно воспользоваться несколькими стратегиями:

- «слабое» vs. «сильное» комбинирование — слабое позволит составить минимум тестов для обнаружения всех дефектов, возникающих на конкретном

значении одного параметра, сильное предназначено для обнаружения дефектов, возникающих на «стыке» значений параметров;

- «нормальное» vs. «надежное» комбинирование — нормальное использует только валидные значения, выбранные для тестирования, надежное — все.

Это независимые взаимодополняющие характеристики, т.е., например, можно применить слабое нормальное комбинирование. Такая стратегия предполагает составление тестов, в которых хотя бы один раз встречаются все валидные значения каждого параметра, выбранного для тестирования.

Пример:

- параметр A может принимать валидные значения a1, a2, a3 и невалидные a4, a5;

- параметр B может принимать валидное значение b1 и невалидные b2, b3, b4;

- параметр C может принимать валидные значения c2, c3 и c4 и невалидное c1.

Слабое нормальное комбинирование выдаст такие тесты:

- a1, b1, c1;

- a2, b1, c2;

- a3, b1, c3;

- a1, b1, c4.

Сильное нормальное комбинирование — все возможные комбинации валидных значений каждого из параметров:

- a1, b1, c1;

- a1, b1, c2;

- a1, b1, c3;

- a1, b1, c4;

- a2, b1, c1;

- a2, b1, c2;

- a2, b1, c3;

- a2, b1, c4;
- a3, b1, c1;
- a3, b1, c2;
- a3, b1, c3;
- a3, b1, c4.

Аналогично можно составить наборы тестов, используя стратегии слабого надежного и сильного надежного комбинирования (попробуйте сделать это для примера выше).

Очевидно, что при использовании сильного и/или надежного комбинирования количество тестов будет резко возрастать при увеличении количества значений какого-либо из параметров и, конечно, при увеличении количества самих параметров. Техника попарного перебора (pairwise) — один из способов уменьшить количество тестов, при этом попытавшись сохранить качество тестирования, т.е. свести к минимуму количество необнаруженных ошибок. Но применяя эту технику, важно понимать, что ошибки на стыке более чем двух значений параметров останутся ненайденными.

Другой способ уменьшить количество тестов — узнать, есть ли зависимости между входными параметрами, и учесть это в тестах [22]. Это возможно далеко не всегда: часто тестирование черного ящика не позволяет «заглянуть внутрь». В этом случае можно попытаться выявить зависимости эмпирически и учесть их в комбинаторных тестах. Однако, есть риск ошибиться при выделении таких закономерностей.

Комбинаторные тесты можно и нужно составлять с помощью соответствующих инструментов, чтобы избежать человеческого фактора.

1.2.3 Таблицы принятия решений

Таблица решений — способ компактного представления модели со сложной логикой; инструмент для упорядочения сложных бизнес требований, которые должны быть реализованы в продукте. Это взаимосвязь между множеством условий и действий.

В таблицах решений представлен набор условий, одновременное выполнение которых должно привести к определённому действию.

Таблицы принятия решений (decision tables/ cause-effect tables) применяются: в случае зависимости исходящих данных или поведения программы от комбинаций значений входящих данных при проверке “бизнес-правил” [9].

Decision Table описывают логику приложения основываясь сущностях (свойства/условия) состояния системы. Каждая decision table должна описывать 1 состояние системы. Шаблон таблицы решений следующий:

Таблица 1 - Шаблон таблицы решений

	Правило 1	Правило 2	...	Правило N
Сущность				
Сущность 1				
...				
Сущность M				
Действие				
Действие 1				
...				
Действие K				

Сущность(conditions) от 1 до m - это разные свойства системы, они представляют в таблице входные данные, которые можно ввести в систему.

Действия(actions) от 1 до n - это действия которые могут произойти с указанной комбинацией сущностей, в зависимости от комбинации всех входных данных сущностей, действия принимают нужные значения. Каждое правило определяет уникальный набор входных данных всех свойств, которые приводят к исполнению конкретных действий.

1.2.4 Диаграмма состояний и переходов

В большинстве случаев требований к состояниям сущности в явном виде нет, и тестировщику приходится формулировать их самому, выделяя нужную информацию из сценариев использования / user stories.

State-Transition testing, как и decision tables testing, отличный инструмент для фиксирования требований и описания дизайна приложения. В отличии от Decision tables testing, которые описывают конкретное состояние приложения, State-Transition testing описывают как эти состояния приложения могут меняться [19]. Диаграммы определяют все события, которые возникают во время работы приложения, и как приложение реагирует на эти события.

Диаграмма состоит из определенного количества элементов:

- состояние (state) (представленное в виде круга на диаграмме) - это состояние приложения, в котором оно ожидает 1 или более событий. Состояние помнит входные данные полученные до этого и показывает, как приложение будет реагировать на полученные события. События могут вызывать смену состояния и/или инициировать действия;

- переход (transition) (представленное в виде стрелки на диаграмме) - представляет переход одного состояния в другое, происходящий по событию;

- событие (event) (представленное ярлыком над стрелкой) - событие это что то, что заставляет приложение поменять свое состояние. События могут поступать извне приложения, поступающие через интерфейс приложения. Так

же само приложение может генерировать события, например, как событие "истек таймер". Когда происходит событие приложение может изменить состояние или остаться в том же состоянии и/или выполнить действие. События могут иметь параметры, например, событие "payMoney" может иметь параметры "Cash", "Check", "Debit Card", или "Credit Card";

- действие (action) (представлено после "/" в ярлыке над преходом) - это действие, инициированное сменой состояния. Это может быть "напечатать билет", "показать на экране" и др. Обычно действия создают что то, что является выходными/возвращаемыми данными системы. Действия возникают при переходах, сами по себе состояния пассивны;

- точка входа показывается на диаграмме как черная точка;

- точка выхода показывается на диаграмме как мишень.

Проектировать тесты по диаграммам состояний

State-Transition Diagrams могут быть легко использованы для создания тест кейсов.

Простые позитивные тесты по диаграммам состояний представляют собой проверку валидных переходов между состояниями, т.е. возможность выполнить действия, обозначенные стрелками. Негативные тесты – проверка невозможности выполнения никаких других действий. Таким образом, легко посчитать количество необходимых позитивных тестов – количество стрелок на диаграмме. Количество негативных тестов тоже несложно подсчитать:

$$T^- = N - T^+, \quad (1)$$

где T^- – количество негативных тестов, N – количество состояний, T^+ – количество позитивных тестов.

Формулу выше легко получить из матричного представления диаграммы состояний. Такое представление строится следующим образом:

- матрица квадратная, количество строк и столбцов равно количеству состояний на диаграмме;

- строки будут обозначать исходящие стрелки, столбцы – входящие;

- если стрелка выходит из состояния 1 и входит в состояние 2, на пересечении строки 1 и столбца 2 следует поставить +.

Таким образом, количество плюсов в матрице соответствует позитивным тестам, а пустых ячеек – негативным. Их количество рассчитывается по формуле выше.

Сложные тесты по диаграмме состояний

Простые позитивные тесты представляют собой, как описано выше, проверку одного корректного перехода между двумя состояниями. Таким образом, можно сформулировать простейший критерий покрытия диаграммы тестами: протестированы все существующие переходы между состояниями, и объект побывал в каждом состоянии хотя бы один раз. Но можно проектировать и более сложные сценарные тесты, состоящие из более чем одного перехода. Такие тесты могут быть в дальнейшем использованы и для нагрузочного тестирования [20]. Аналогично, можно задать более сильные критерии покрытия тестами: проверены все возможные пары переходов между состояниями, тройки и т.д. (до N-1, где N – количество состояний объекта). Такое покрытие называется покрытием переходом Чау или покрытием N-1 переходов. Соответственно, самое слабое покрытие будет покрытием 0 переходов.

1.3 Инструменты для создания тест-кейсов

Существует несколько видов артефактов для описания проверок реализации тестируемой функции или её части.

Тестовый сценарий (test case)

Тестовый сценарий (test case) или тестовый случай — набор входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанный для определенной цели или тестового условия, таких как выполнение определенного пути программы или же для проверки соответствия определенному требованию. [ISTQB]

Тестовые сценарии, как минимум, состоят из действий (шагов) и ожидаемых результатов. В некоторых случаях, описываются предусловия и постусловия. Также рекомендуется создавать краткое содержание тестового случая [27]. В содержании обычно описывается цель теста и выбор методики тестирования. Некоторые тесты сопровождаются визуальной информацией — карты, схемы и т.д.

1. Предусловие (Pre Conditions) — список действий или критерии, которые приводят систему к состоянию пригодному для проведения основной проверки.

2. Шаги / Ожидаемые результаты (Actions / Results) — список действий, переводящих систему из одного состояния в другое, для получения результата, на основании которого можно сделать вывод о удовлетворении реализации, поставленным требованиям.

3. Постусловие (Post Conditions) -список действий, переводящих систему в первоначальное состояние.

В качестве предусловия часто выступает задача — создать тестовые данные.

Тестовые данные (test data) — данные, которые существуют (например, в базе данных) на начало выполнения теста и влияют на работу, или же испытывают влияние со стороны тестируемой системы или компонента [21].

Группа тестовых сценариев называется — тестовым набором (test suite). Группировать тесты можно по тестовым областям, объекту тестирования, типу тестов, цели тестов и т.д.

Чек-лист (check list)

Чек-лист (check list) — это документ, описывающий что должно быть протестировано [10]. При этом чек-лист может быть абсолютно разного уровня детализации [7]. На сколько детальным будет чек-лист зависит от требований к отчетности, уровня знания продукта сотрудниками и сложности продукта.

Как правило, чек-лист содержит только действия (шаги), без ожидаемого результата. Чек-лист менее формализован чем тестовый сценарий. Его уместно использовать тогда, когда тестовые сценарии будут избыточны. Также чек-лист ассоциируются с гибкими подходами в тестировании.

Зачем нужен чек-лист?

- не забыть что-то протестировать;
- помогает осуществлять контроль за тестированием.

Какие преимущества чек-листов по сравнению с тест-кейсами:

- нивелирование эффекта пестицида в регрессионном тестировании;
- расширение тестового покрытия за счёт отличий при прохождении;
- сокращение затрат на содержание и поддержку тестов;
- отсутствие рутины, которую так не любят квалифицированные тестировщики;
- возможность проходить и комбинировать тесты по-разному, в зависимости от предпочтений сотрудников;

При этом, чек-листы сохраняют множество плюсов, за которые так популярны детальные тест-кейсы:

- статистика: кто, когда, что проходил (с детализацией по сборке продукта и окружению, на котором проводилось тестирование);

- памятка, которая помогает не забыть важные тесты;

- возможность оценить состояние продукта, его готовность к выпуску.

Конечно, было бы нечестно рассказать про плюсы и умолчать о минусах чек-листов:

- начинающие тестировщики не всегда эффективно проводят тесты без достаточно подробной документации;

- чек-листы невозможно использовать для обучения начинающих сотрудников, так как в них недостаточно подробной информации;

- заказчику или руководству может быть недостаточно того уровня детализации, который предлагают чек-листы.

Чит-лист (cheat sheets)

Чит-лист (cheat sheets) — список проверок, которые можно использоваться в разных условиях. Другими словами — набор стандартных проверок, который пригодится на все (или почти все) случаи жизни. Например, элементы управления и формы отлично покрываются чит-листами. Во многих компаниях их еще называют guideline [7].

Чит-листы являются одной разновидность чек-листов. Главным отличием является то, что чек-листы, как правило, имеют узкую направленность на конкретный объект тестирования.

Интеллект-карты (mind map)

Интеллект-карты (mind map) - это особый вид записи материалов в виде радиантной структуры, то есть структуры, исходящей от центра к краям, постепенно разветвляющейся на более мелкие части. Интеллект-карты могут заменить традиционный текст, таблицы, графики и схемы [13].

Оптимальным инструментом, сочетающим в себе плюсы тест-кейсов и чек-листов, на мой взгляд, является MindMap (или альтернативное русское название интеллект-карта).

Основные преимущества использования интеллект-карт:

- удобная статистика: с MindMap можно сказать со уверенностью, какие именно компоненты продукта уже проверялись, а какие все еще нуждаются в проверке;

- надежность тестирования: можно очень долго тестировать приложение, но так и не убедиться в том, что проверено действительно все. Постоянно актуализированная MindMap, охватывающая весь функционал, позволит минимизировать риски по пропуску ошибок;

- полный охват картины тестирования: из постановок неясно какой процент требований реализован, какая информация неактуальна, что будет реализовано в будущем. Тестировщику, который знаком с проектом, проще войти в курс дела, не перечитывая постановки и задачи;

- иерархическая система интеллект-карты удобна своей наглядностью. Правка кода, соответствующего одной ветки наглядно показывает каких дочерних веток коснутся изменения – их и нужно будет проверять в первую очередь [23].

1.4 Исследовательское тестирование

Исследовательское тестирование (exploratory testing): Неформальный метод проектирования тестов, при котором тестировщик активно контролирует проектирование тестов в то время, как эти тесты выполняются, и использует полученную во время тестирования информацию для проектирования новых и улучшенных тестов [4, 31].

Для управления исследовательским тестированием применяется методика тестовых сессий - ограниченных промежутков времени, в рамках которых происходит тестирование. При этом каждая сессия имеет тему [5].

Чтобы систематизировать исследовательское тестирование можно использовать идею туров [11].

Тур – это в своем роде план тестирования. Который отражает цели и задачи, на которых будет сконцентрировано внимание тестировщика во время

сессии. В данном случае проводится аналогия между тестировщиком как туристом, а тестируемое приложение – это город. Обычно у туриста (тестировщика) мало времени, поэтому он выполняет конкретную задачу в рамках выбранного тура, ни на что другое не отвлекаясь.

Вывод

В первой главе рассматриваются различные типы проектов, а также методы проектирования и написания тестовых случаев.

Рассмотренные методы проектирования являются базовыми в тест-анализе. Так же для них необходим низкий порог входа, что позволяет тестировщикам любой квалификации использовать их. Одним из результатов было участие в II научно-практической конференции аспирантов с темой «Чек-листы и чит-листы как инструмент тестирования»

Касательно классификации it-проектов, то для оценки тестирования необходимо знать о времени и бюджете проектов, чему удовлетворяет деление по масштабу и длительности.

ГЛАВА 2 ОБОБЩЕННЫЙ АЛГОРИТМ ОПРЕДЕЛЕНИЯ НАБОРА ИНТРУМЕНТОВ ТЕСТИРОВАНИЯ ПО

2.1 Описание алгоритма

При выборе техник тест-дизайна нужно учитывать:

1. Время и бюджет проекта.

2. Доступная документация.

3. Навыки команды.

Для формирования тест-комплекта необходимо учитывать такие параметры проекта как длительность и стоимость проекта.

1. Собираем и изучаем документацию на проект/функционал. Для формирования тест-комплекта необходимо учитывать такие-виды документации как:

- требование к программному обеспечению и функциональные спецификации;

- user story, описанные в системе управления проектами;

- другие виды документации, описанные в п 2.3 Документация в проекте.

2. Оценить время на проектирование и создание тест-кейсов

Время на проектирования тест-кейсов оцениваем методом “Грубая экспертная оценка”, данная оценка учитывает создание позитивных и негативных сценариев (2.2.1 Оценка времени на проектирование и создание тестовых случаев)

3. Сравнить со временем, выделенным на тестирование в проекте и отбросить техники проектирования, занимающие более 10% данного времени.

- если отбрасываются все техники, то тестовая документация на проекте не создается, используется метод исследовательского тестирования.

4. Умножить на коэффициент инструментов тестирования время проектирования тест-кейсов:

4.1 Тест-кейсы - 3,14;

4.2 Чек-листы - 1,57;

4.3 Интеллект-карты - 1,2.

5. Сравнить со временем, выделенным на тестирование в проекте.

6. Отбросить техники, которые занимают более 30% процентов.

- если отбрасываются все техники, то тестовая документация на проекте не создается, используется метод исследовательского тестирования.

7. Оценить время на проверку тест-кейсов.

Время на проверку созданных тест-кейсов оцениваем методом “Индуктивной оценки” (п. 2.2.2 Оценка времени на проверку тестовых случаев).

8. Отбросить техники превышающие выделенное время на тестирование.

- если отбрасываются все техники, то тестовая документация на проекте не создается, используется метод исследовательского тестирования;

9. Выбираем технику и инструмент с меньшим временем.

Ниже представлена блок-схема алгоритма.

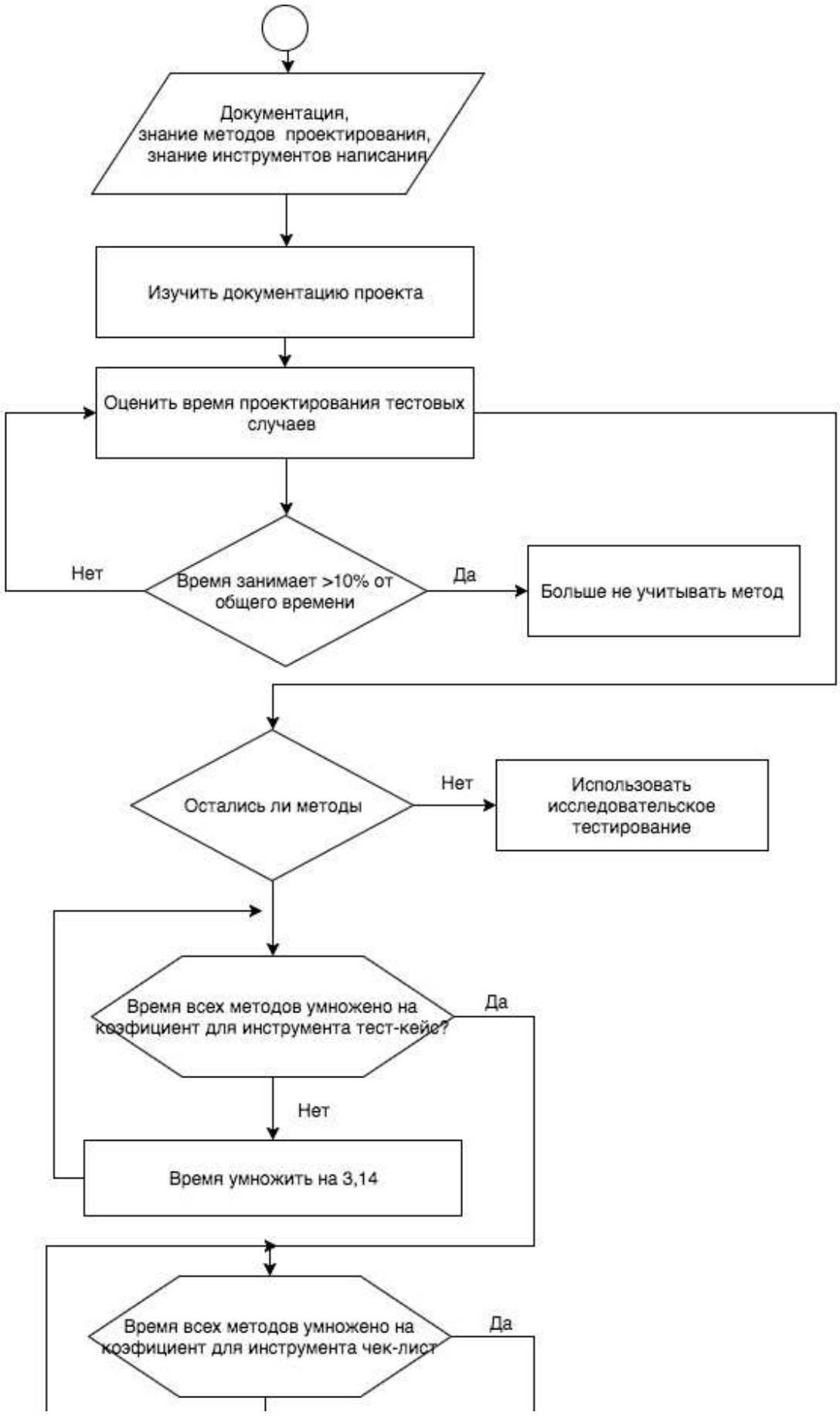

Рисунок 1 - Блок-схема алгоритма определения оптимального набора инструментов тестирования ПО. Часть 1.

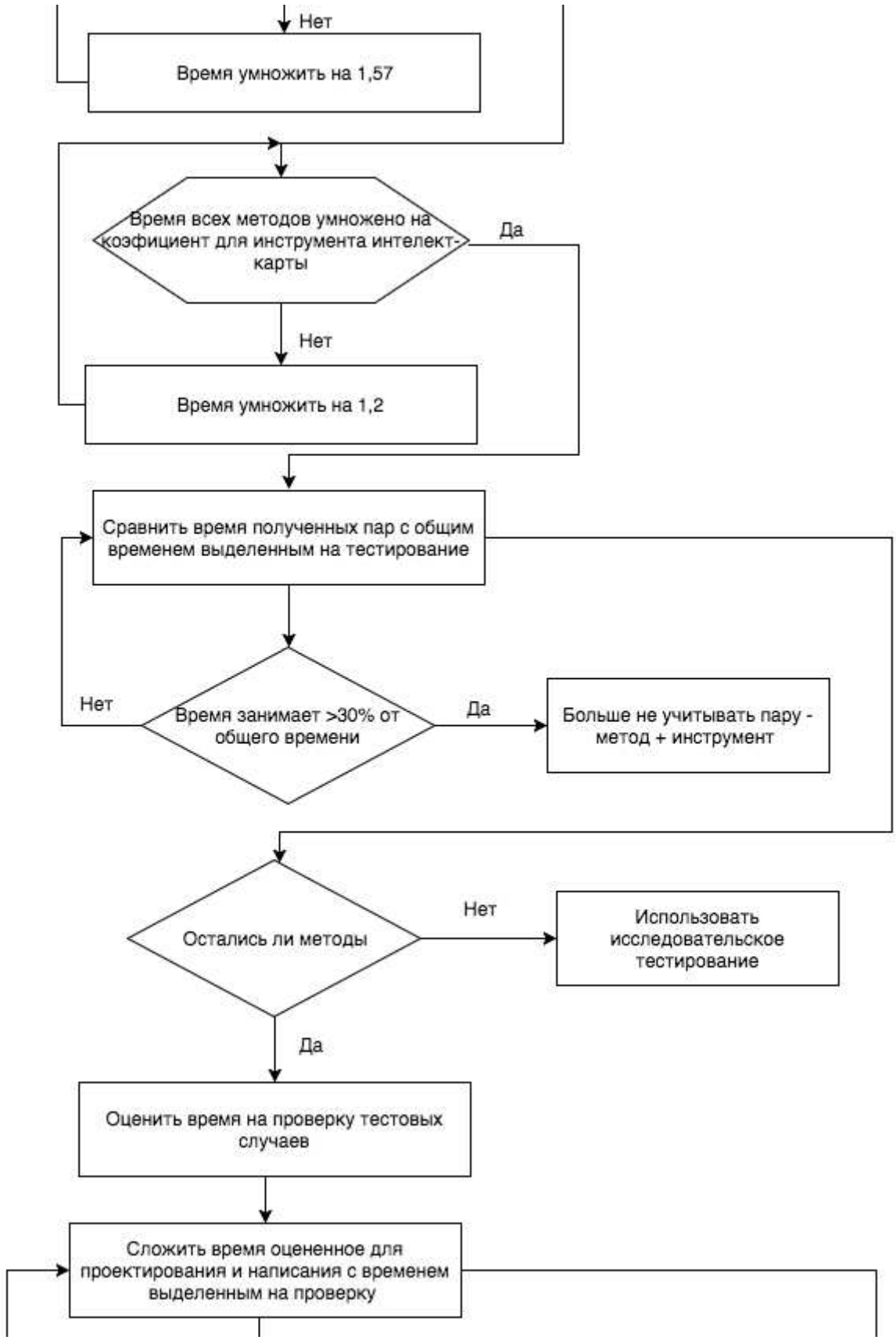

Рисунок 2 - Блок-схема алгоритма определения оптимального набора инструментов тестирования ПО. Часть 2

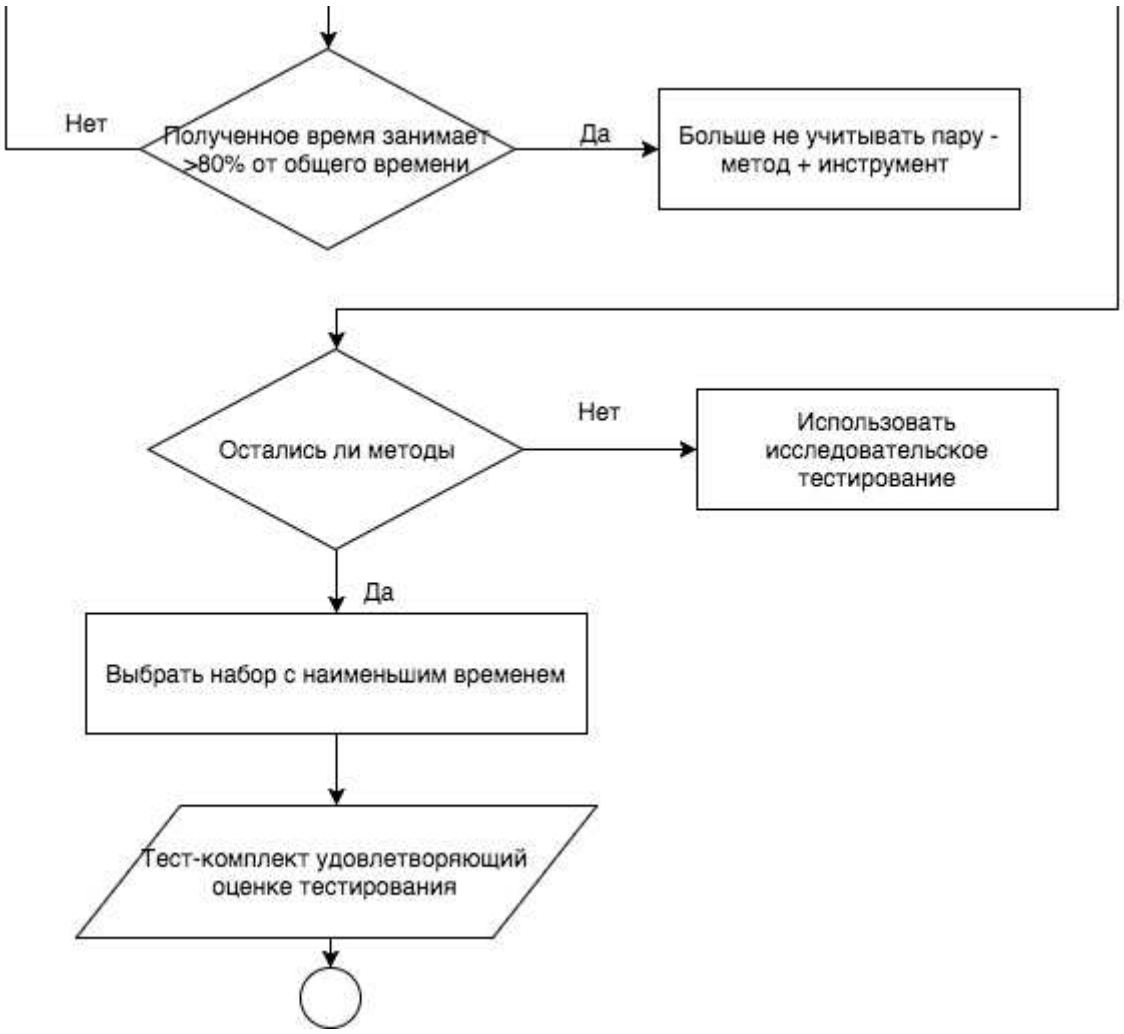

Рисунок 3 - Блок-схема алгоритма определения оптимального набора инструментов тестирования ПО. Часть 3

2.2 Формальная постановка задачи

2.2.1 Классическая задача о рюкзаке

Классическая задача о рюкзаке представляет собой рюкзак грузоподъемность V и n предметов с весом и стоимостью каждый. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины V (вместимость рюкзака), а суммарная стоимость была максимальна [24].

В рамках данной работы будет будут использованы соответствия:

- рюкзак с грузоподъёмностью V – проект с временем, выделенным на тестирования приложения;

- предметы – различные наборы техник проектирования и инструментов создания тестовых случаев.

Наборы будут формироваться в результате комбинирования методов проектирования и инструментов создания тестовых случаев, а именно:

1. Проектирование методом доменного анализа с использованием инструмента описания тестовый случаев – тест кейсы.

2. Проектирование методом доменного анализа с использованием инструмента описания тестовых случаем – чек-листы.

3. Проектирование методом доменного анализа с использованием инструмента описания тестовых случаем – интеллект-карты.

4. Проектирование методом парных значений с использованием инструмента описания тестовых случаем – тест-кейсы.

5. Проектирование методом парных значений с использованием инструмента описания тестовых случаем – чек-листы.

6. Проектирование методом парных значений с использованием инструмента описания тестовых случаем – интеллект-карты.

7. Проектирование методом таблиц принятия решений с использованием инструмента описания тестовых случаем – тест-кейсы.

8. Проектирование методом таблиц принятия решений с использованием инструмента описания тестовых случаем – чек-листы.

9. Проектирование методом таблиц принятия решений с использованием инструмента описания тестовых случаем – интеллект-карты.

10. Проектирование методом диаграмм состояний и переходов с использованием инструмента описания тестовых случаем – тест-кейсы.

11. Проектирование методом диаграмм состояний и переходов с использованием инструмента описания тестовых случаем – чек-листы.

12. Проектирование методом диаграмм состояний и переходов с использованием инструмента описания тестовых случаем – интеллект-карты.

Весом предмета будет является время необходимое на проектирование, написание и выполнение тестовых случаев. А стоимостью предмета коэффициент сложности применения тестировщиком.

Формальное описание задачи:

Пусть дан проект с выделенным временем на тестирование V и наборы инструментов тестирования ПО с весом (итоговым временем выполнения) $a_j > 0$ и стоимостью $c_j > 0,\ j = 1, \dots, n.$

Введем логические переменные x_j – признак наличия предмета в наборе:

$$x_j = \begin{cases} 1, & \text{если набор техник с номером } j \text{ помещены в проект} \\ 0, & \text{иначе.} \end{cases}$$

Общий вес заполненного проекта ограничен временем, выделенным на тестирование:

$$\sum_{j=1}^{n} a_j x_j \ \leq V. \qquad (2)$$

Ограничение на переменный:

$x \epsilon \{0,1\}, j = 1, \dots, n.$

Максимизации суммарной стоимости наборов инструментов соответствует целевая функция:

$$\sum_{j=1}^{n} c_j x_j \rightarrow max. \quad (3)$$

2.3 Оценка тестирования

Процесс создания программной компоненты начинается с этапа планирования. При этом оценивается не только сама разработка продукта, но и его тестирование. В основу оценки берется опыт предыдущей работы. Разрабатывается архитектура системы, создаются подробные требования к системе. Требование может состоять не только в создании нового свойства программной компоненты, но и в изменении существующих свойств. Каждое требование к системе оценивается, получая временное и стоимостное выражение оценки. В результате, полная стоимость всех требований не должна превышать бюджет, выделенный на этап создания программной компоненты.

После начала работы и выполнения некоторого количества требований, становятся известны реальные временные сроки выполнения запланированных работ. Реальное время выполнения работ может отличаться от запланированного времени вследствие ошибок в объеме работ или мощности используемых ресурсов (другие причины изменения времени здесь не учитываем) [28].

В процессе работы требуется регулярно получать прогноз времени выполнения и стоимости работ, который позволил бы принимать своевременные меры по снижению рисков проекта.

Таким образом, задача оценки трудозатрат состоит в нахождении достоверных оценок длительности и стоимости разработки, удовлетворяющих следующим требованиям:

1. Оценка трудозатрат по созданию программной компоненты должна опираться на предшествующий опыт оценки аналогичных компонент.

2. Оценка должна оценивать, как новую работу, так и работу по внесению изменений и переработку алгоритмов работы компоненты.

3. Оценка должна учитывать квалификацию сотрудников.

Все эти требования также применимы к оценке тестирования программного продукта.

2.3.1 Оценка времени на проектирования и создание тестовых случаев

Существует множество решений, по оценке трудозатрат тестирования программного обеспечения [29]. Наиболее распространенные:

Грубая экспертная. Например, известно, что обычно на одну страницу технического задания пишется 5 тестовых случаев. Обычно один тестовый случай описывается 20 мин. В техническом задании 300 страниц. 5 х 20 х 300 = 30000 — итоговое количество часов, требующееся на проектирование и написание тестовых случаев.

$$\sum T = (T_1 + T_{-1}) * X * Y, \quad (4)$$

где X – количество тестовых случаев на страницу технического задания, T_1 – количество минут, требуемых на написание позитивного тестового случая, Y – количество страниц в техническом задании, T_{-1} - количество минут, требуемых на написание негативного тестового случая.

Грубая дедуктивная. Допустим, есть план проекта, есть сроки, есть команда. И есть четко представление, какое соотношение разработчиков к тестировщикам должно быть в проекте. Например, проект на полгода. Разработчиков — пятеро (значит, нужны два тестировщика на эти полгода). Видно, что писать тестовые случаи тестировщики будут примерно четверть времени от всего времени, отведенного на тестирование. Это грубый, но быстрый способ подсчета.

Тестирование занимает X времени от времени проекта. Написание тестовых случаев Y времени от времени на тестирование

$$\sum T = T_{project} * X * Y, \quad (5)$$

Где $X, Y \epsilon [0; 1]$, X,Y – зависят от типа проектов.

Индуктивно-опытная. Это медленный способ оценки, но он хорошо работает. В данном способе необходимо попробовать написать тестовые случаи для каждой части технического задания. Делать это лучше с той степенью подробности, с которой в дальнейшем они будут описываться. Впрочем, можно сделать хотя бы высокоуровневые тестовые случаи. Это может занять день, два, неделю, но зато поможет понять, сколько времени уйдет вообще.

Разбиваем техническое задание на однородные части (например, по функционалу) и пробуем писать тестовые случаи для каждой части.

Смотрим, сколько времени ушло на написание тестовых случаев для такого-то количества страниц каждой части, исходя из этого вычисляем, сколько всего времени должно уйти на написание тестовых случаев для каждой части.

Складываем посчитанное для каждой части время:

$$\sum T = \sum_{a=1}^{n} T_{part\ of\ a} * Q_{parts\ in\ areas}. \quad (6)$$

2.3.2 Оценка времени на проверку тестовых случаев

При оценке тестирования важно знать не только время необходимое для проектирования и написания тестовых случаев, также время необходимое на проверку функционала приложения по созданным тестовым случаям [26]. Для оценки выполнения тестов также есть несколько способов:

1. $T_1 \approx T_2$. (7)

Можно смотреть, сколько времени обычно уходит на тестирование *похожего функционала*.

Можно смотреть на *тот же функционал*, если ранее уже тестировался подобный функционал, но в таком случае необходимо собирать статистику по времени тестирования $T_1 \approx T_1^{`}$.

2. $T_1 = T\frac{Q_{tc}}{Q_{tc1}}$. (8)

Дедуктивная оценка — необходимо знать, сколько тестируемый функционал занимает места в общем функционале (если, например, уже известно, сколько времени занимает полная регрессия), тогда видно, что тестирование функционала займет, например, пятую часть от всего времени.

3. $T_1 = \sum T_{tc} + T_{env} + T_{precond}$. (9)

Индуктивная оценка — известно, сколько времени тратиться на каждый тестовый случай. Также известно количество тестовых случаем для тестирования данного функционала.

Также важно помнить, что новый функционал может снова затребовать время на подготовку среды. Важно помнить о существовании коэффициента на знакомство с системой: больше времени на тесты уходит, если прогонять их в первый раз, чем если тесты прогоняются второй или третий раз. Но если второй раз наступает через месяц, то лучше учитывать коэффициент на знакомство с системой.

2.4 Документация в проекте

В общем случае документацию можно разделить на два больших вида в зависимости от времени и места её использования [12].

Продуктная документация используется проектной командой во время разработки и поддержки продукта. Она включает:

- план проекта и в том числе тестовый план;

- требования к программному продукту и функциональные спецификации;

- архитектуру и дизайн;

- тест-кейсы и наборы тест-кейсов;

- технические спецификации, такие как схемы баз данных, описания алгоритмов, интерфейсов и т.д.

Проектная документация включает в себя как продуктную документацию, так и некоторые дополнительные виды документации и используется не только на стадии разработки, но и на более ранних и поздних стадиях (например, на стадии внедрения и эксплуатации). Она включает:

- пользовательскую и сопроводительную документацию, такую как встроенная помощь, руководство по установке и использованию, лицензионные соглашения и т.д.

- маркетинговую документацию, которую представители разработчика или заказчика используют как на начальных этапах (для уточнения сути и концепции проекта), так и на финальных этапах развития проекта (для продвижения продукта на рынке).

В некоторых классификациях часть документов из продуктной документации может быть перечислена в проектной документации — это совершенно нормально, т.к. понятие проектной документации по определению является более широким [18]. На рисунке 1 представлена графическая форма классификации документации по признаку того, где (для чего) она является наиболее востребованной.

Рисунок 4 - классификация документации проекта

Степень важности и глубина тестирования того или иного вида документации и даже отдельного документа определяется большим количеством факторов, но неизменным остаётся общий принцип: всё, что мы создаем в процессе разработки проекта (даже рисунки маркером на доске, даже письма, даже переписку в скайпе), можно считать документацией и так или иначе подвергать тестированию или использовать во время тестирования.

Вывод

Описанный алгоритм позволяет получить полную оценку тестирования, разбитую на этапы и набор инструментов для формирования тестового комплекта. Формализация задачи позволяет автоматизировать процесс оценки тестирования.

Также из описанных методов оценки проектирования был выбран – метод грубой экспертной оценки. Данный метод не привязывается к количеству тестировщиков на проекте и является быстрым в применении.

Для оценки времени выполнения тестов был выбран метод индуктивной оценки. Данный метод учитывает не только время на проверку тестовых случаев, но время на подготовку окружения и ознакомления с системой (если вы тестируете продукт в первый раз).

ГЛАВА 3 ПРИМЕНЕНИЕ АЛГОРИТМА ПОСТРОЕНИЯ ТЕСТ-КОМПЛЕКТА НА РЕАЛЬНЫХ ЗАДАЧАХ

Разработанный алгоритм был опробован на нескольких проектах компании.

3.1 Проект А. Веб-приложения для государственного учреждения

Стоимость проекта: 435 400 рублей.

Проект был оценен на 622 часов.

Время, выделенное на тестирование – 160 час.

Оценка времени на проектирование и написание тест-кейсов.

На проект был назначен тестировщик с опытом работы в тестировании от года.

Перед стартом проекта было создано и описано 15 пользовательских сценариев.

При оценке проектирования тестов используется метод Грубой экспертной оценки.

В формуле 4, в данном случае, X – количество тестовых случаев на один пользовательский сценарий, Y – количество пользовательских сценариев, T_1 – время, требуемое для написания одного позитивного тестового случая, T_{-1} - время, требуемое для написания одного негативного тестового случая.

Результаты оценки для каждого метода проектирования.

Доменный анализ:

1. На один пользовательский сценарий пишется, в среднем, 4 позитивных и 7 негативных тестов. X = 11.

2. Всего в проекте 17 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки положительно тестового случая 2 минут.

4. Для разработки негативного теста 3 минут.

$\sum T = (2 + 3) * 17 * 11 = 935$ минут $= 15{,}58$ часа.

Тестирование парных значений:

1. На один пользовательский сценарий пишется, в среднем, 12 позитивных и 6 негативных тестов. X = 18

2. Всего в проекте 17 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки положительно тестового случая 2 минут.

4. Для разработки негативного теста 4 минут.

$\sum T = (2 + 4) * 22 * 8 = 1836$ минут $= 30{,}6$ часа.

Таблица принятия решений:

1. На один пользовательский сценарий пишется, в среднем, 13 тестов.

2. Всего в проекте 17 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки тестового случая 4 минут.

$\sum T = 13 * 17 * 4 = 884$ минут $= 14{,}73$ часа.

Диаграмма состояний и переходов:

1. На один пользовательский сценарий пишется, в среднем, 13 тестов.

2. Всего в проекте 17 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки тестового случая 5 минут.

$\sum T = 13 * 5 * 17 = 1105$ минут $= 18{,}42$ часа.

После оценки времени на проектирования тестов и выделенным временем на тестирования для дальнейшего анализа подходит методы таблица принятия решений и доменный анализ, потому что только у данных методов время оказалось меньшим и равным 10% от общего времени тестирования.

Далее время проектирования умножается на коэффициент инструментов тестирования:

1. Доменный анализ:

a. Использование тест-кейсов – 15,58 ч * 3,14 = 48,92.

b. Использование чек-листов – 15,58 ч * 1,57= 24,46.

c. Использование интеллект карт – 15,58 ч * 1,2= 18,7.

2. Таблица принятия решений:

a. Использование тест-кейсов – 14,73 ч * 3,14 = 46,25.

b. Использование чек-листов – 14,73 ч * 1,57= 23,12.

c. Использование интеллект карт – 14,73 ч * 1,2= 17,68.

Далее отбрасываются инструменты, время использование которых занимает более 30% от общего времени тестирования – 48 часов. Удовлетворяю же критерию -чек-листы, интеллект карты для обоих методов, а также тест-кейсы для таблиц принятия решений.

Финальным этапом происходит оценка времени исполнения тестовых случаем, используя метод индуктивной оценки с помощью формулы 9.

Для обоих комбинаций (метод проектирования + инструмент написания) закладывается одинаковое время на подготовку среды и риски, 2 часа и 1 час соответственно.

Общее количество тестовых сценариев при доменном анализе – 187

Общее количество тестовых сценариев при использовании таблицы принятия решений – 221.

Время выполнения тестов по тест-кейсам – 35 минут.

$T_2 = 35 * 221 + 60 + 120 = 7915$ минут $= 131{,}92$ часа.

Итоговое время необходимое для тестирования:

$T_{общ2} = 46{,}25 + 131{,}92 = 178{,}17$ часа.

Время выполнения тестов по чек-листам – 20 минут.

$T_1 = 20 * 187 + 60 + 120 = 3920$ минут $= 65{,}33$ часа.

$T_2 = 20 * 221 + 60 + 120 = 4600$ минут $= 76{,}67$ часа.

Итоговое время необходимое для тестирования:

$T_{общ1} = 24{,}46 + 65{,}33 = 89{,}79$ часа.

$T_{общ2} = 23{,}12 + 76{,}67 = 99{,}79$ часа.

Время выполнения тестов по интеллект-карте – 25 минут.

$T_1 = 25 * 187 + 60 + 120 = 4855$ минут $= 80{,}92$ часа.

$T_1 = 25 * 221 + 60 + 120 = 5705$ минут $= 95{,}08$ часа.

Итоговое время необходимое для тестирования:

$T_{общ1} = 18{,}7 + 80{,}92 = 99{,}62$ часа.

$T_{общ2} = 17{,}68 + 95{,}08 = 112{,}76$ часа.

Итоговое время при использовании тест-кейсов превышает 80% от общего времени тестирования. В остальных случаях можно выбрать минимальную связку.

Вывод

Тест-комплект с минимальным временем – связка доменный анализ и чек-листы. Но так как все тест-комплекты не превышают ограничения, то тестировщик может выбрать удобный для него метод. При этом будет использоваться только один метод, так как комбинация превышает выделенное время на тестирование.

Реальное время тестирование – 100 часов, что превышает оцененное время 10%.

3.2 Проект Б. Мобильное приложение

Стоимость проекта: 190 000 рублей.

Проект был оценен на 523 часов.

Время, выделенное на тестирование – 142 час.

Оценка времени на проектирование и написание тест-кейсов.

На проект был назначен тестировщик с опытом работы в тестировании в общем до года, в тестировании мобильных приложений – отсутствует.

Перед стартом проекта было создано и описано 8 пользовательских сценариев.

При оценке проектирования тестов используется метод Грубой экспертной оценки.

В формуле 4, в данном случае, X – количество тестовых случаев на один пользовательский сценарий, Y – количество пользовательских сценариев, T_1 – время, требуемое для написания одного позитивного тестового случая, T_{-1} - время, требуемое для написания одного негативного тестового случая.

Результаты оценки для каждого метода проектирования.

Доменный анализ:

1. На один пользовательский сценарий пишется, в среднем, 6 позитивных и 9 негативных тестов. X = 15.

2. Всего в проекте 8 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки положительно тестового случая 4 минут.

4. Для разработки негативного теста 3 минут.

$\sum T = (4 + 3) * 15 * 8 = 840$ минут $= 14$ часа.

Тестирование парных значений:

1. На один пользовательский сценарий пишется, в среднем, 15 позитивных и 7 негативных тестов. X = 22.

2. Всего в проекте 8 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки положительно тестового случая 2 минут.

4. Для разработки негативного теста 4 минут.

$$\sum T = (2 + 4) * 22 * 8 = 1056 \text{ минут} = 17{,}6 \text{ часа}.$$

Таблица принятия решений:

1. На один пользовательский сценарий пишется, в среднем, 20 тестов.

2. Всего в проекте 8 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки тестового случая 7 минут.

$$\sum T = 20 * 7 * 8 = 1120 \text{ минут} = 18{,}67 \text{ часа}.$$

Диаграмма состояний и переходов:

1. На один пользовательский сценарий пишется, в среднем, 20 тестов.

2. Всего в проекте 8 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки тестового случая 6 минут.

$$\sum T = 20 * 6 * 8 = 960 \text{ минут} = 16 \text{ часа}.$$

После оценки времени на проектирования тестов и выделенным временем на тестирования для дальнейшего анализа подходит методы диаграмм состояний и переходов и доменный анализ, потому что только у данного метода время оказалось меньшим и равным 10% от общего времени тестирования.

Далее время проектирования умножается на коэффициент инструментов тестирования:

1. Доменный анализ:

a. использование тест-кейсов – 14 ч * 3,14 = 43,96;

b. использование чек-листов – 14 ч * 1,57= 21,98;

c. использование интеллект карт – 14 ч * 1,2= 16,8.

2. Диаграмма состояний и переходов:

a. использование тест-кейсов – 16 ч * 3,14 = 50,24;

b. использование чек-листов – 16 ч * 1,57= 25,12;

c. использование интеллект карт - 16 ч * 1,2= 19,2;

Далее отбрасываются инструменты, время использование которых занимает более 30% от общего времени тестирования – 42,6 часов. Удовлетворяю же критерию -чек-листы и интеллект карты.

Финальным этапом происходит оценка времени исполнения тестовых случаем, используя метод индуктивной оценки с помощью формулы 9.

Для обоих комбинаций (метод проектирования + инструмент написания) закладывается одинаковое время на подготовку среды и риски, 1 час для каждого параметра.

Общее количество тестовых сценариев при доменном анализе – 120

Общее количество тестовых сценариев при использовании диаграммы состояний и переходов - 160

Время выполнения тестов по чек-листам – 20 минут.

$T_1 = 20 * 120 + 60 + 60 = 2520$ минут $= 42$ часа.

$T_2 = 20 * 160 + 60 + 60 = 3320$ минут $= 55{,}33$ часа.

Итоговое время необходимое для тестирования:

$T_{общ1} = 42 + 21{,}98 = 63{,}98$ часа.

$T_{общ2} = 55{,}33 + 25{,}12 = 80{,}45$ часа.

Время выполнения тестов по интеллект-карте – 25 минут.

$T_1 = 25 * 120 + 60 + 60 = 3120$ минут $= 52$ часа.

$T_1 = 25 * 160 + 60 + 60 = 4120$ минут $= 68{,}67$ часа.

Итоговое время необходимое для тестирования:

$T_{общ1} = 52 + 16{,}8 = 68{,}8$ часа.

$T_{общ2} = 68{,}67 + 19{,}2 = 87{,}87$ часа.

Так как итоговое время всех инструментов не превышает 80% от общего времени тестирования, то необходимо выбрать тест-комплект с минимальным временем.

Вывод

Тест-комплект с минимальным временем – связка доменный анализ и чек-листы. Но так как все тест-комплекты не превышают ограничения, то тестировщик может выбрать удобный для него метод. При этом будет использоваться только один метод, так как комбинация превышает выделенное время на тестирование.

Реальное время тестирования – 65 часов, что превышает оцененное время на 1%.

3.3 Проект B. Чат-бот в Telegram

Стоимость проекта: 43 400 рублей.

Проект был оценен на 54 часов.

Время, выделенное на тестирование – 16 час.

Оценка времени на проектирование и написание тест-кейсов.

На проект был назначен тестировщик с опытом работы в тестировании в общем до года.

Перед стартом проекта было создано и описано 3 пользовательских сценариев.

При оценке проектирования тестов используется метод Грубой экспертной оценки.

В формуле 4, в данном случае, X – количество тестовых случаев на один пользовательский сценарий, Y – количество пользовательских сценариев, T_1 – время, требуемое для написания одного позитивного тестового случая, T_{-1} - время, требуемое для написания одного негативного тестового случая.

Результаты оценки для каждого метода проектирования.

Доменный анализ:

1. На один пользовательский сценарий пишется, в среднем, 3 позитивных и 5 негативных тестов. X = 8.

2. Всего в проекте 3 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки положительно тестового случая 2 минут.

Для разработки негативного теста 3 минут.

$$\sum T = (2 + 3) * 8 * 3 = 120 \text{ минут} = 2 \text{ часа}$$

Тестирование парных значений:

1. На один пользовательский сценарий пишется, в среднем, 15 тестов.

2. Всего в проекте 3 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки положительно тестового случая 2 минут.

4. Для разработки негативного теста 2 минут.

$$\sum T = (2 + 2) * 15 * 3 = 180 \text{ минут} = 3 \text{ часа}$$

Таблица принятия решений:

1. На один пользовательский сценарий пишется, в среднем, 12 тестов.

2. Всего в проекте 3 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки тестового случая 5 минут.

$$\sum T = 12 * 5 * 3 = 180 \text{ минут} = 3 \text{ часа}$$

Диаграмма состояний и переходов:

1. На один пользовательский сценарий пишется, в среднем, 6 тестов.

2. Всего в проекте 3 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки тестового случая 5 минут.

$$\sum T = 6 * 5 * 3 = 90 \text{ минут} = 1{,}5 \text{ часа}$$

После оценки времени на проектирования тестов и выделенным временем на тестирования для дальнейшего анализа подходит метод диаграмм состояний и переходов, потому что только у данного метода время оказалось меньшим и равным 10% от общего времени тестирования.

Далее время проектирования умножается на коэффициент инструментов тестирования:

Диаграмма состояний и переходов:

a. использование тест-кейсов – 1,5 ч * 3,14 = 4,71;

b. использование чек-листов – 1,5 ч * 1,57= 2,36;

c. использование интеллект карт – 1,5 ч * 1,2= 1,8.

Далее отбрасываются инструменты, время использование которых занимает более 30% от общего времени тестирования – 4,8 часов. Удовлетворяю критерию все инструменты.

Финальным этапом происходит оценка времени исполнения тестовых случаем, используя метод индуктивной оценки с помощью формулы 9.

Для обоих комбинаций (метод проектирования + инструмент написания) закладывается одинаковое время на подготовку среды и риски, 1 час для каждого параметра.

Общее количество тестовых сценариев – 8

Время выполнения тестов по тест-кейсам – 50 минут

$$T_1 = 8 * 50 + 60 + 60 = 520 \text{ минут } = 8{,}67 \text{ часа}$$

Итоговое время необходимое для тестирования:

$$T_{\text{общ1}} = 4{,}71 + 8{,}67 = 13{,}38 \text{ часа}$$

Время выполнения тестов по чек-листам – 40 минут.

$$T_1 = 8 * 40 + 60 + 60 = 440 \text{ минут } = 7{,}33 \text{ часа}$$

Итоговое время необходимое для тестирования:

$$T_{\text{общ2}} = 2{,}36 + +7{,}33 = 9{,}66 \text{ часа}$$

Время выполнения тестов по интеллект-карте – 40 минут.

$$T_1 = 40 * 8 + 60 + 60 = 440 \text{ минут } = 7{,}33 \text{ часа}$$

Итоговое время необходимое для тестирования:

$$T_{\text{общ3}} = 1{,}8 + 7{,}3 = 9{,}1 \text{ часа}$$

При использовании тест-кейсов итоговое время превышает 80% выделенного времени на тестирование. В данном случае выбирается минимальная по времени связка – диаграмма состояний и переходов и интеллект-карты.

Реальное время тестирования – 10,2 часов, что превышает на 11% оцененное время.

3.4 Проект Г. «Сайт агрегатор для бронирования и покупки»

Стоимость проекта: 473 538 рублей.

Проект был оценен на 592 часов.

Время, выделенное на тестирование – 234 час.

Оценка времени на проектирование и написание тест-кейсов.

На проект был назначен тестировщик с опытом работы в тестировании в общем от 2-х лет.

Перед стартом проекта было создано и описано 20 пользовательских сценариев.

При оценке проектирования тестов используется метод Грубой экспертной оценки.

В формуле 4, в данном случае, X – количество тестовых случаев на один пользовательский сценарий, Y – количество пользовательских сценариев, T_1 – время, требуемое для написания одного позитивного тестового случая, T_{-1} - время, требуемое для написания одного негативного тестового случая.

Результаты оценки для каждого метода проектирования.

Доменный анализ:

1. На один пользовательский сценарий пишется, в среднем, 5 позитивных и 9 негативных тестов. X = 14

2. Всего в проекте 20 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки положительно тестового случая 2 минут.

4. Для разработки негативного теста 3 минут.

$$\sum T = (2 + 3) * 14 * 20 = 1400 \text{ минут} = 23{,}33 \text{ часа}$$

Тестирование парных значений:

1. На один пользовательский сценарий пишется, в среднем, 25 тестов.

2. Всего в проекте 20 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки положительно тестового случая 1 минут.

Для разработки негативного теста 2 минут.

$$\sum T = (2 + 1) * 25 * 20 = 1500 \text{ минут} = 25 \text{ часа}$$

Таблица принятия решений:

1. На один пользовательский сценарий пишется, в среднем, 17 тестов.

2. Всего в проекте 20 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки тестового случая 4 минут.

$$\sum T = 17 * 4 * 20 = 1360 \text{ минут} = 22{,}67 \text{ часа}$$

Диаграмма состояний и переходов:

1. На один пользовательский сценарий пишется, в среднем, 20 тестов.

2. Всего в проекте 20 пользовательских сценариев.

3. Тестировщику с описанным выше опытом работы необходимо для разработки тестового случая 3 минут.

$$\sum T = 20 * 3 * 20 = 1200 \text{ минут} = 20 \text{ часа}$$

После оценки времени на проектирования тестов и выделенным временем на тестирования для дальнейшего анализа подходит метод диаграмм состояний и переходов, потому что только у данного метода время оказалось меньшим и равным 10% от общего времени тестирования.

Далее время проектирования умножается на коэффициент инструментов тестирования:

Доменный анализ:

a. Использование тест-кейсов: 23,33 * 3,14 = 73,26;

b. Использование чек-листов: 23,33 * 1,57 = 36,63;

c. Использование интеллект-карт: 23,33 * 1,2 = 28.

Таблица принятия решений:

a. Использование тест-кейсов: 22,67 * 3,14 = 71,2;

b. Использование чек-листов: 22,67 * 1,57 = 35,59;

c. Использование интеллект-кат: 22,67 * 1,2 = 27,2

Диаграмма состояний и переходов:

a. Использование тест-кейсов – 20 ч * 3,14 = 62,8;

b. Использование чек-листов – 20 ч * 1,57= 31,4;

c. Использование интеллект карт – 20 ч * 1,2= 24.

Далее отбрасываются инструменты, время использование которых занимает более 30% от общего времени тестирования – 70,2 часов. Не подходит использование тест-кейсов для доменного анализа и таблиц принятия решений.

Финальным этапом происходит оценка времени исполнения тестовых случаем, используя метод индуктивной оценки с помощью формулы 9.

Для обоих комбинаций (метод проектирования + инструмент написания) закладывается одинаковое время на подготовку среды и риски, 1 час для каждого параметра.

Доменный анализ:

Общее количество тестовых сценариев – 280

Время выполнения тестов по чек-листам – 35 минут.

$$T_1 = 280 * 35 + 60 + 60 = 9920 \text{ минут } = 165{,}33 \text{ часа}$$

Итоговое время необходимое для тестирования:

$$T_{\text{общ1}} = 36{,}33 + 165{,}33 = 201{,}66 \text{ часа}$$

Время выполнения тестов по интеллект-карте – 30 минут.

$$T_1 = 280 * 35 + 60 + 60 = 9920 \text{ минут } = 165{,}33 \text{ часа}$$

Итоговое время необходимое для тестирования:

$$T_{\text{общ2}} = 28 + 165{,}33 = 198{,}33 \text{ часа}$$

Таблицы принятия решений:

Общее количество тестовых сценариев – 340

Время выполнения тестов по чек-листам – 25 минут.

$$T_1 = 340 * 30 + 60 + 60 = 8620 \text{ минут } = 143{,}67 \text{ часа}$$

Итоговое время необходимое для тестирования:

$$T_{\text{общ1}} = 35{,}59 + +143{,}67 = 179{,}26 \text{ часа}$$

Время выполнения тестов по интеллект-карте – 30 минут.

$$T_1 = 340 * 30 + 60 + 60 = 10320 \text{ минут } = 172 \text{ часа}$$

Итоговое время необходимое для тестирования:

$$T_{\text{общ2}} = 27{,}2 + 172 = 199{,}2 \text{ часа}$$

Диаграмма состояний и переходов:

Общее количество тестовых сценариев – 400

Время выполнения тестов по тест-кейсам – 30 минут.

$$T_1 = 400 * 30 + 60 + 60 = 12120 \text{ минут } = 202 \text{ часа}$$

Итоговое время необходимое для тестирования:

$$T_{\text{общ1}} = 62{,}8 + 202 = 264{,}8 \text{ часа}$$

Время выполнения тестов по чек-листам – 20 минут.

$$T_1 = 400 * 20 + 60 + 60 = 8120 \text{ минут } = 135{,}33 \text{ часа}$$

Итоговое время необходимое для тестирования:

$$T_{общ2} = 31{,}4 + 135{,}33 = 166{,}73 \text{ часа}$$

Время выполнения тестов по интеллект-карте – 20 минут.

$$T_1 = 400 * 20 + 60 + 60 = 8120 \text{ минут } = 135{,}33 \text{ часа}$$

Итоговое время необходимое для тестирования:

$$T_{общ1} = 24 + 135{,}33 = 159{,}33 \text{ часа}$$

При использовании тест-кейсов итоговое время превышает 80% выделенного времени на тестирование. В данном случае выбирается минимальная по времени связка – диаграмма состояний и переходов и интеллект-карты.

Реальное время тестирования – 187 часа, что превышает на 15% оцененное время.

Вывод

В третьей главе приведены результаты использования алгоритма. Разработанный алгоритм позволяет дать оценку, которая совпадет с погрешностью 10-15% с реальным временем тестирования.

При накоплении данных, в дальнейшем можно улучшить алгоритм изменив проценты отбрасывания методом и инструментов на каждом этапе, а также модифицировать методы оценки.

ЗАКЛЮЧЕНИЕ

В данной диссертационной работе было проведено исследование классификации IT-проектов, были выбраны инструменты тестирования и методы проектирования тестовых случаем, также разработан алгоритм для выбора оптимального тест-комплекта, приведены результаты использования алгоритма.

В первой главе рассматриваются различные типы проектов, а также методы проектирования и написания тестовых случаев.

Рассмотренные методы проектирования являются базовыми в тест-анализе. Так же для них необходим низкий порог входа, что позволяет тестировщикам любой квалификации использовать их. Одним из результатов было участие в II научно-практической конференции аспирантов с темой «Чек-листы и чит-листы как инструмент тестирования»

Касательно классификации IT-проектов, то для оценки тестирования необходимо знать о времени и бюджете проектов, чему удовлетворяет деление по масштабу и длительности.

Во второй главе описан алгоритм, позволяющий получить полную оценку тестирования, разбитую на этапы и набор инструментов для формирования тестового комплекта. Формальная постановка задачи позволяет автоматизировать процесс оценки тестирования.

Также из описанных методов оценки проектирования был выбран – метод грубой экспертной оценки. Данный метод не привязывается к количеству тестировщиков на проекте и является быстрым в применении.

Для оценки времени выполнения тестов был выбран метод индуктивной оценки. Данный метод учитывает не только время на проверку тестовых случаев, но время на подготовку окружения и ознакомления с системой (если вы тестируете продукт в первый раз).

В третьей главе приведены результаты использования алгоритма. Разработанный алгоритм позволяет дать оценку, которая совпадет с погрешностью 10-15% с реальным временем тестирования.

При накоплении данных, в дальнейшем можно улучшить алгоритм изменив проценты отбрасывания методом и инструментов на каждом этапе, а также модифицировать методы оценки.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Glenford, J. M. The Art of Software Testing / J. Myers Glenford – New York: John Wiley & Sons, Inc., 1979. – 300 с.

2. Copeland, Lee. A Practitioner's Guide to Software Test Design / Lee Copeland – London: Artech House Publishers, 2003. – 320 с.

3. Patton, Ron. Software Testing: учебное пособие / Ron Patton – USA: Sams Publishing, 2001. – 389 с.

4. Whittaker, James A. Exploratory Software Testing / James A. Whittaker – USA: Pearson Education, 2009. - 270 с.

5. Блэк, Р. Ключевые процессы тестирования. Планирование, подготовка, проведение, совершенствование: пер. с англ / Рекс Блэк – Москва: Лори, 2011. – 544 с.

6. Куликов, С. Тестирование программного обеспечения. Базовый курс. Версия книги 1.0.3 от 07.09.2015 [Электронный ресурс]. Режим доступа: https://svyatoslav.biz/software_testing_book/ (дата обращения: 15.05.2017).

7. Официальный сайт веб-сервиса Sitechco [Электронный ресурс]. Режим доступа: http://www.sitechco.ru/ (дата обращения: 20.01.2017).

8. Канер, С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений: Пер. с англ. / Сэм Канер, Джек Фолк, Енг Кек Нгуен. - К.: ДиаСофт, 2001. - 544 с.

9. Бейзер, Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер. — СбП: Питер, 2004. — 320 с.

10. Стандартный глоссарий терминов, используемых в тестировании программного обеспечения [Электронный ресурс] / пер. А. Александров [и др.]. 2014. Режим доступа: http://www.rstqb.org/fileadmin/user_upload/redaktion/rstqb_ru/downloads/ISTQB_Glossary_Russian_v2.3.pdf (дата обращения: 15.09.2015).

11. Калбертсон, Роберт. Быстрое тестирование: Пер. с англ./Роберт Калбертсон, Крис Браун, Гэри Кобб. - Москва: Издательский дом «Вильямс», 2007. -374 с.

12. Тамре, Луиза. Введение в тестирование программного обеспечения: Пер. с англ. -Москва: Издательский дом «Вильямс», 2003. -368 с.

13. Про Тестинг - Тестирование Программного Обеспечения: [Электронный ресурс], Режим доступа: http://www.protesting.ru/ (Дата обращения: 27.04.2017).

14. Постовалова, А.С. Чек-листы и чит-листы как инструмент тестирования / А.С. Постовалова // Сборник трудов II научно-практической конференции аспирантов, преподавателей, ученых конференции. – 2016. – 37-38 с.

15. Управление проектами: основы профессиональных знаний. Национальные требования к компетентности специалистов/Под ред. В.И. Воропаева. -М.: СОВНЕТ, Кубс Групп, 2001. -265 с.

16. Хилл, Чапел, Мифический человеко-месяц, или Как создаются программные системы: пер. с англ./ Фредерик Брукс, Хилл Чапер. – Санкт-Петербург: Символ-Плюс, 2010. – 304 с.

17. Савкин, В. Принципы управления качеством программ. // «Открытые системы», №6, 2008 г. -С.49-53.

18. Сергеева, А. Инструменты тестировщика, или С чего начать новичку. // «Системный администратор», №7-8, 2014 г. - 70-74 с.

19. Software Testing Tutorial: [Электронный ресурс], Режим доступа: https://www.tutorialspoint.com/software_testing/index.htm (Дата обращения: 02.05.2017).

20. База знаний по тестированию: [Электронный ресурс], Режим доступа: http://qalight.com.ua/baza-znaniy/qa-qc-i-testirovanie/ (Дата обращения: 02.05.2017).

21. Galin, D. Software Quality Assurance: from theory to implementation. / G. Galin. – London: Pearson Education Limited, 2004. – 590 с.

22. Майерс, Г. Искусство тестирования программ/Пер. с англ. под ред. Б.А. Позина. -М.: Финансы и статистика, 1982. -176 с.

23. Лайза, Криспин. Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд / пер. с англ под ред Н. Мухин / Лайза Криспин, Джанет Грегори – Москва: Вильямс, 2016. – 464 с.

24. Обзор методов разработки алгоритмов: [Электронный ресурс], режим доступа: https://petrsu.ru/files/document/uploads/algol.pdf (дата обращения: 15.05.2017)

25. ГОСТ 28806-90 «Качество программных средств. Термины и определения». -М.: ИПК Изд-во стандартов, 2001. -35 с.

26. Планирование трудозатрат на тестирование: [Электронный ресурс], Режим доступа: https://habrahabr.ru/company/ibm/blog/94686/ (Дата обращения: 02.04.2017).

27. ISO/IEC 9126-1:2001. Software engineering -Software product quality - Part 1: Quality model.

28. Сокращение затрат на обеспечение качества программных продуктов: [Электронный ресур], Режим доступа: https://habrahabr.ru/company/ibm/blog/94686/ (дата обращения: 28.03.2017).

29. Оценка трудоемкости и сроков разработки ПО: [Электронный ресурс], Режим доступа: http://www.arkhipenkov.ru/resources/sw_project_estimation.pdf (дата обращения: 28.03.2017).

30. Тест-дизайн: [Электронный ресурс], Режим доступа: https://qaevolution.ru/testovaya-dokumentaciya/test-dizajn/ (дата обращения 28.04.2017).

31. Ad-hoc testing: [Электронный ресурс], Режим доступа: https://qaevolution.ru/testirovanie-po/vidy-testirovaniya-po/ad-hoc-testing/ (дата обращения 28.04.2017).

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра Информатики
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.С. Кузнецов
«____» _____2017 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме магистерской диссертации

Студенту Постоваловой Александре Сергеевне
фамилия, имя, отчество

Группа КИ15-04-01М Направление (специальность) 09.04.04
номер код
Программная инженерия
наименование

Тема выпускной квалификационной работы Алгоритм определения оптимального набора инструментов тестирования программного обеспечения

Утверждена приказом по университету № 4337/с от 04 апреля 2017

Руководитель ВКР О.А. Антамошкин,доц. кафедры Информатики СФУ, к.т.н
инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР

Перечень разделов ВКР Анализ предметной области; обобщенный алгоритм определения набора инструментов тестирования ПО; применение алгоритма построения тест-комплекта на реальных задачах;
Перечень графического материала

Руководитель ВКР _____ О.А. Антамошкин
подпись инициалы и фамилия

Задание принял к исполнению _____ А.С.Постовалова
подпись инициалы и фамилия студента

« ___ » _____ 2017 г.

Рецензия

на выпускную квалификационную работу студентаИКИТ СФУ,
Постоваловой Александры Сергеевны,
на тему: «Определение оптимального набора инструментов тестирования программного обеспечения».

Рецензируемая квалификационная работа содержит 60 страницы, 31 использованных источников, 1 рисунок, 1 таблицу и 17 формул.

Актуальность тематики данной работы обуславливается тем, что на сегодня тестирование программного продукта сейчас является важным этапом разработки, и в связи с этим необходимо точно знать требуемые затраты времени.

В работе обработано достаточное количество научного материала, на высоком теоретическом и методологическом уровне проведен анализсистемы взаимодействия распределенного учреждения, анализ документооборота и информационной системы.Материал в выпускной квалификационной работе изложен с соблюдением внутренней логики, между разделами прослеживается логическая взаимосвязь.

Автор выпускной квалификационной работы показал отличный уровень владения теоретическими положениями по выбранной теме исследования и показал способность формулировать собственную точку зрения.

Постоваловой А.С. были выявлены и проанализированы основные критерии влияющие на оценку тестирования, также был подобран инструментарий необходимый для формирования тест-комплекта. Предложенный алгоритм выбора инструментов написания тестовых случаев и методов проектирования позволяет полностью решить поставленные задачи, тем самым достигнут цели работы.

Рассматриваемаявыпускная квалификационная работа полностью соответствует предъявляемым требованиям, заслуживает **отличной**оценки иможет бытьрекомендована к защите, а ее автор Постовалова А.С – присвоения квалификации «магистр» по специальности 09.04.04 «Программная инженерия», магистерской программы 09.04.04.01 «Программное обеспечение вычислительной техники и автоматизированных систем».

Рецензент
д-р.техн.наук, профессор,
зав.кафедрыЭиИТМИУБПЭ СФУ_____ Ступина А.А.

Заявление о согласии выпускника на размещение выпускной квалификационной работы в электронно-библиотечной среде ФГАОУ ВО СФУ

1 Я, Постовалова Александра Сергеевна

фамилия, имя, отчество полностью

студент (ка) ИКИТ, КИ15-04-01М

институт/ группа

Федерального государственного автономного образовательного учреждения высшего образования «Сибирский федеральный университет» (далее – ФГАОУ ВО СФУ), разрешаю ФГАОУ ВО СФУ безвозмездно воспроизводить и размещать (доводить до всеобщего сведения) в полном объеме написанную мною в рамках выполнения образовательной программы

указать выпускную квалификационную работу бакалавра, дипломную работу специалиста, дипломный проект специалиста, магистерскую диссертацию

на тему: Алгоритм определения оптимального набора инструментов тестирования программного обеспечения

название работы

в открытом доступе в электронно-библиотечной среде (на веб-сайте СФУ), таким образом, чтобы любой пользователь данного портала мог получить доступ к выпускной квалификационной работе (далее – ВКР) из любого места и в любое время по собственному выбору, в течение всего срока действия исключительного права на выпускную работу.

2 Я подтверждаю, что выпускная работа написана мною лично, в соответствии с правилами академической этики и не нарушает авторских прав иных лиц.

«19» 05.17

подпись

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра Информатики
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.С. Кузнецов
«13» 06 2017 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Алгоритм определения оптимального набора инструментов тестирования программного обеспечения

09.04.04 «Программная инженерия»
код и наименование направления
09.04.04.01 «Программное обеспечение вычислительной техники и автоматизированных систем»
код и наименование магистерской программы

Научный руководитель _____ 9.06.17 доц. кафедры Информатики СФУ, к.т.н О.А.Антамошкин
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 9.06.17 А.С. Постовалова
подпись, дата инициалы, фамилия

Рецензент _____ 9.06.17 зав. кафедры ЭиИТМ СФУ, д.т.н А.А. Ступина
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2017