**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**

**(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | ДОПУСКАЕТСЯ К ЗАЩИТЕ: | | | | | | | | | | | | | | |
| Факультет |  | |  | Заведующий кафедрой | | | | | |  | | | |  | | | | | | |
|  | индекс факультета | |  |  | | | | |  | | | | | | индекс кафедры | | |
| Выпускающая кафедра |  | |  | |  | | |  | | | |  | | | | | | | |
|  | индекс кафедры | |  | | Фамилия ИО | |  | | | | подпись | | | | | | | | | | |
| Группа |  | |  | «\_\_\_\_\_» | |  | | | | | | | | | | 20\_\_\_ г. | | |
|  | индекс группы | |  |  | |  | | | | | | |  | | | |

**отчет**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | о научно-исследовательской работе в семестре | | | | | | | | | | | | | | | | | | |  | | | |
|  | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | |
| Фамилия, имя, отчество обучающегося | | | | | | | | | | | | | | | | | | |
| **обучающегося по**  **направлению/специальности** | | | | | | | |  | |  | |  | | | | | | | | |
| нужное подчеркнуть | | | | | | | | код | | |  | | полное наименование направления/специальности | | | | |
|  | | | | | | | | | | | | | | | | |
| **Руководитель НИРС:** | | |  | | | | | | | | | | | | | | | | |
|  | | | | Фамилия ИО, ученая степень, ученое звание, должность | | | | | | | | | | | | | | | | | |
| **Срок прохождения НИРС:** | | | | | | с | |  | | | | | г. |  | по |  | | г. | | | | |
| **Должность обучающегося:** | | | |  | | | | | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Руководитель НИРС:** | | | | |  | | |  |  | | | | |  | | | | | |
|  | | | |  |  | | |  | |  | | |  | |  | | | | |
| Подпись | | | |  | Фамилия ИО | | |  | |  | | | |  | |  | | |
| **Оценка:** | | |  |  |  | |  |  | |  | | | |  | |  | | |
|  | | | |  |  | | |  | |  | | | |  | |  | | |
| «\_\_\_» |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |  | 20\_\_г. |  |  | |  |  | | | | |  |  | | |

САНКТ-ПЕТЕРБУРГ

2018 г.

УТВЕРЖДАЮ:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Заведующий кафедрой | | | | | | | \_\_\_\_\_\_\_\_ | | |
|  | | | |  | | индекс кафедры | | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |  | | \_\_\_\_\_\_\_\_\_\_\_ | | | | |
| Фамилия ИО | | |  | | подпись | | | |
| « \_\_\_\_» |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | 20\_\_\_г. | |

**Задание**

|  |  |  |
| --- | --- | --- |
| **на** |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Обучающемуся группы** | | |  | | |  |  | | | | | | | | | | | | |
|  | | | группа | | |  | Фамилия Имя Отчество | | | | | | | | | | | | |
| **Направление/специальность** | | |  | | | | |  |  | | | | | | | | | | |
| нужное подчеркнуть | | | код | | | | | | |  | полное наименование направления/специальности | | | | | | |
|  | | | | | | | | | | | | | | | | | | | |
| **Руководитель НИРС** |  | | | | | | | | | | | | | | | | | | |
|  | Фамилия ИО, ученая степень, ученое звание, должность | | | | | | | | | | | | | | | | | | |
| **Тема задания на НИРС:** | |  | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | |
| **Срок прохождения НИРС:** | | | | с |  | | | | | | | | г. |  | по |  | | | г. | |
| **Место прохождения НИРС:** | | | |  | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | |
| **Должность обучающегося:** | |  | | | | | | | | | | | | | | | | | |
| 1. **Виды работ и требования к их выполнению:** | | | | | | | | | | | |  | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | |
| 1. **Виды отчетных материалов и требования к их выполнению:** | | | | | | | | | | | | | | | | |  | | |
|  | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | |

1. **ПЛАН-ГРАФИК НИРС**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № этапа | | | Наименование этапа | | | | | | Срок завершения этапа | | | | Виды работ | | | | | | | Форма  отчетности | | | |
|  | | |  | | | | | |  | | | |  | | | | | | |  | | | |
|  | | |  | | | | | |  | | | |  | | | | | | |  | | | |
|  | | |  | | | | | |  | | | |  | | | | | | |  | | | |
|  | | |  | | | | | |  | | | |  | | | | | | |  | | | |
|  | | |  | | | | | |  | | | |  | | | | | | |  | | | |
| **Дата выдачи задания:** | | | | | | | | | | |  | | | **Срок сдачи отчетных документов:** | | | | | | | | | | |
| « | |  | | » |  | 20\_\_ | | г. | |  | | | « | |  | » |  | | 20\_\_ | | г. | |
| **Руководитель НИРС:** | | | | | | |  | | | | | **Обучающийся:** | | | | | |  | | | |
| « |  | | | » |  | 20\_\_ | | г. | |  | | | « | |  | » |  | | 20\_\_ | | г. | |

**Дневник**

|  |  |  |
| --- | --- | --- |
|  | о научно-исследовательской работе в семестре |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Обучающегося группы** | |  | | |  | |  | | | | | | | | |
|  | | группа | | |  | | Фамилия Имя Отчество | | | | | | | | |
| **По направлению/специальности** | | |  | | | | |  |  | | | | | | |
| нужное подчеркнуть | | | код | | | | |  | полное наименование направления/специальности | | | | | | |
|  | | | | | | | | | | | | | | | |
| **Срок прохождения НИРС:** | | | | с | |  | | | | г. |  | по |  | | г. | |
| **Место прохождения НИРС:** | | | |  | | | | | | | | | | | |
|  | | | | полное наименование организации | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **В должности:** |  | | | | | | | | | | | | |
|  | | | | указать должность | | | | | | | | | | | |

|  |  |  |
| --- | --- | --- |
| Дата | Вид работ | Примечание |
| 3.09 – 20.10 | Обзор литературы по тематике диссертации |  |
| 21.10 – 4.11 | Обзор существующих методик тестирования ПО |  |
| 5.11 – 20.11 | Анализ требований к тестированию ПО |  |
| 21.11 – 7.12 | Оформление отчета |  |

**Обучающийся** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc531907686)

[1 Понятие тестирования ПО 7](#_Toc531907687)

[2 Обзор существующих методик тестирования ПО 13](#_Toc531907688)

[2.1 Каскадная модель (Линейная последовательная модель жизненного цикла ПО) 13](#_Toc531907689)

[2.2 V-Model (Модель верификации и валидации) 14](#_Toc531907690)

[2.3 Инкрементная модель 16](#_Toc531907691)

[2.4 Спиральная модель 17](#_Toc531907692)

[2.5 Agile 18](#_Toc531907693)

[2.6 Экстремальное программирование (XP, Extreme Programming) 19](#_Toc531907694)

[2.7 Scrum 20](#_Toc531907695)

[3 Анализ требований к тестированию ПО 21](#_Toc531907696)

[ЗАКЛЮЧЕНИЕ 22](#_Toc531907697)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc531907698)

## ВВЕДЕНИЕ

Необходимость проведения тестирования программного обеспечения (ПО) впервые была определена в США в 90-е годы. Это было обусловлено быстрыми темпами развития систем автоматизированной разработки программного обеспечения и сетевых технологий, что привело к росту рынка производства программного обеспечения, следовательно, увеличилась конкуренция между производителями. Производители программного обеспечения стали уделять особое внимание качеству создаваемых продуктов, так как пользователь мог выбирать, у какого из производителей приобрести требуемое программное обеспечение по достаточно приемлемым ценам. Производителям ПО должны были решить задачу разработки программного обеспечения в короткие сроки с соответствующим качеством.

Такой подход требовал проводить качественное тестирование программного обеспечения. Некоторые компании создавали системы управления качеством, используемые для совершенствования производственных процессов создания и постоянного повышения качества ПО.

На сегодняшний день тестирование стало неотъемлемой частью процесса производства ПО. Тестирование используется для обнаружения наибольшего количества ошибок, а, следовательно, повышения качества продукта по всем его характеристикам. Целью любого тестирования является обеспечение корректного выполнения работы ПО. Также задачей процесса тестирования является снижение стоимости разработки с помощью раннего обнаружения ошибок и недочетов ПО.

Независимо от количества существующих методик тестирования, нет одной единой техники, с помощью которой можно было бы выявить, что ПО не содержит ошибок.

Целью научно-исследовательской работы является обзор литературы по тематике диссертации и выделение предметной области.

Исходя из сформулированной цели необходимо решить следующие задачи:

* обзор литературы по тематике диссертации;
* анализ существующих методик тестирования;
* анализ требований к тестированию ПО.

## 1 Понятие тестирования ПО

В настоящий момент существует множество определений процесса тестирования программного обеспечения, ниже приведены наиболее распространенные из них.

В соответствии со стандартом Institute of Electrical and Electronics Engineers (IEEE Std 829-1983), тестирование – это процесс анализа ПО, направленный на выявление отличий между его реально существующими и требуемыми свойствами (дефект) и на оценку свойств ПО [1].

Согласно [2], тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий две различные цели:

* продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;
* выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Таким образом, тестирование программного обеспечения направлено на нахождение дефектов (багов) и демонстрации соответствия требованиям к программному обеспечению.

Можно привести еще одно определение тестирования [3]: тестирование – это проверка соответствия программного обеспечения требованиям, осуществляемая с помощью наблюдения за его работой в специальных, искусственно построенных ситуациях. Такого рода ситуации называют тестовыми или просто тестами.

Программным дефектом называется ошибка в программном продукте, вследствие которой продукт ведет себя непредвиденно (неверно). Большинство дефектов возникают из-за допущенной ошибки в программном коде или логической ошибки, допущенной во время проектирования [4].

Тестирование программного обеспечения является одной из наиболее важных стадий разработки и жизненного цикла программного обеспечения.

Модель жизненного цикла программного обеспечения — структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта [5].

На рисунке 1 приведена спиральная модель жизненного цикла ПО.

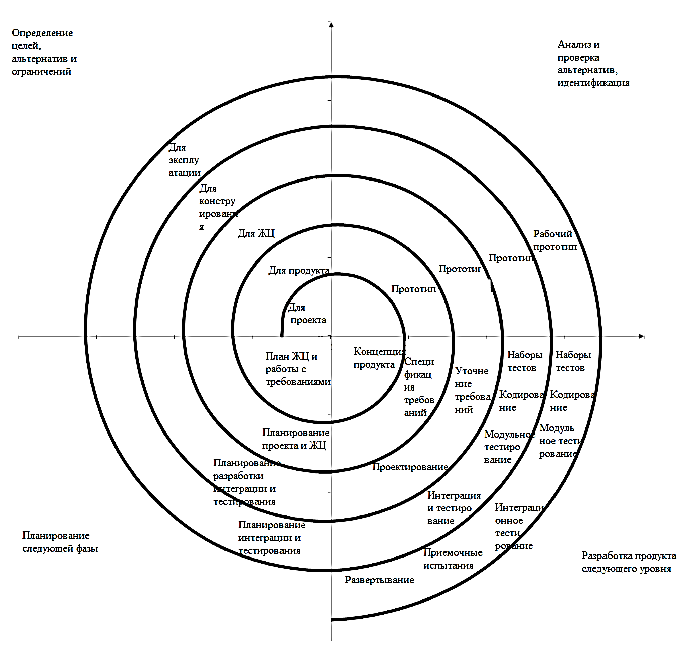


Рисунок 1 – Спиральная модель жизненного цикла ПО

Процесс тестирования ПО проводится задолго до стадии «Тестирование» в жизненном цикле процесса разработки программного обеспечения. На стадии разработки документации на программное изделие, специалисту по тестированию передается документация для составления тест-плана. Тест-план – это документ, который описывает работы по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе тестирования оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения [6]. Разработанный тест-план должен содержать следующие разделы:

* объект тестирования (описание системы объекта тестирования, оборудования, приложения);
* детализация объекта тестирования (список функций, описание тестируемой системы и ее компонентов);
* время проведения тестирования (непосредственная последовательность проведения тестирования);
* объем тестирования;
* критерии начала тестирования (готовность тестового стенда, готовность требуемого функционала, и так далее);
* критерии окончания тестирования.

Основой тестирования программного обеспечения является набор тестовых процедур, в которых определены начальные условия и ожидаемые результаты, разработанные для проверки соответствующей функции. Тестовые процедуры могут проверять различные аспекты функционирования программы – от правильной работы отдельной функции до реализации бизнес-логики во взаимодействии с другими программными комплексами.

Тесты разрабатываются специалистами по тестированию на основании документации на программное изделие, а также частных спецификаций. Тесты организованы как наборы входных данных и ожидаемого результата. Ожидаемый результат работы программы необходимо описывать заранее, то есть до запуска программного обеспечения. Набор тестов всегда конечен, то есть процесс тестирования является конечной процедурой. Тестирование необходимо проводить в рамках проекта, не слишком увеличивая сроки разработки и стоимость программного обеспечения, поэтому набор тестов должен быть относительно небольшим. При проведении тестирования программного обеспечения должны быть составлены оценки качества программного обеспечения – критерии полноты тестирования. Они описывают важность события для того, чтобы оценивать качество. Так же необходимо составлять тесты для неправильных или непредусмотренных входных данных для обнаружения ошибок при непредусмотренных действиях пользователя.

На рисунке 2 схематически представлен процесс тестирования. На основе проверяемых требований и критерии полноты тестирования составляется набор тестов, в ходе которого выполняется каждый из тестов, а затем комплексный анализ их результатов.

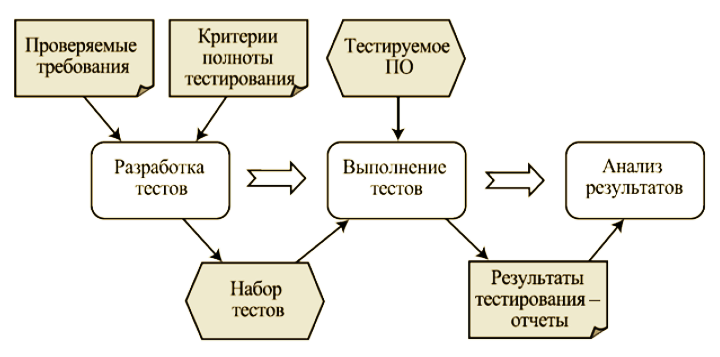


Рисунок 2 – Схема процесса тестирования

Фундаментальный принцип разработки тестов: тест является хорошим, если высока вероятность обнаружить дефект. Это является подтверждением того, что процесс тестирования направлен на поиск дефектов.

Тестирование программного обеспечения на протяжении всего жизненного цикла ПО проводится на разных уровнях для успешной реализации и сдачи проекта. Уровни тестирования определяют объект тестирования: над отдельными модулями, группами модулей или целой системой. В соответствии с [7] выделяют следующие уровни тестирования:

1) Компонентное (модульное) тестирование (Unit Testing)

Компонентное (модульное) тестирование проверяет отдельные части программного обеспечения, которые могут быть протестированы по-отдельности, то есть независимо от их окружения. Обычно модульное тестирование проводит разработчик.

2) Интеграционное тестирование (Integration Testing)

Интеграционное тестирование используется для проверки связи между компонентами, а так же для проверки взаимодействия различных частей системы. Интеграционное тестирование делится по уровням:

* компонентный уровень (используется для проверки взаимодействия между компонентами после проведения компонентного тестирования);
* системный уровень (используется для проверки взаимодействия между системами после проведения системного тестирования).

Существует два подхода к интеграционному тестированию:

* снизу-вверх (сначала выполняется тестирование низкоуровневых модулей, процедур или функций);
* сверху-вниз (сначала выполняется тестирование высокоуровневых модулей, и постепенно добавляются низкоуровневые).

3) Системное тестирование

Системное тестирование – это тестирование функциональных и нефункциональных требований к системе в целом. Системное тестирование проводится на полностью готовой системе с целью проверки соответствия исходным требованиям.

Возможные дефекты, выявляемые при системном тестировании: неверное использование ресурсов системы, несовместимость с окружением, непредусмотренные сценарии использования, неверная или отсутствующая функциональность, неудобство использования. Системное тестирование проводится с помощью различных разрабатываемых интерфейсов программного обеспечения.

4) Приемочные испытания

Приемочные испытания – это тестирование готового программного обеспечения заказчиками в условиях, в которых будет функционировать программное обеспечение, как правило, в рамках опытной эксплуатации.

Виды тестирования делятся на 3 группы [6]:

1) Функциональные;

2) Нефункциональные;

3) Связанные с изменениями.

Функциональные виды тестирования основаны на функциях и взаимодействиях с другими системами. Функциональные виды тестирования могут проводиться на всех уровнях тестирования (модульном, интеграционном, системном, приемочном). Функциональные виды тестирования проверяют внешнее поведение программного обеспечения.

Нефункциональные виды тестирования оценивают второстепенные свойства программного обеспечения.

Связанные с изменениями виды тестирования направлены на проверку исправления дефекта, который был найден ранее, а так же работоспособности всего программного обеспечения после исправления дефекта.

## 2 Обзор существующих методик тестирования ПО

В соответствии с [8] можно дать определение методики: методика – совокупность, система конкретных методических приемов, используемых для решения теоретических задач или способов, средств целесообразного проведения какой-либо работы.

Также в источнике [8] приведены требования к методике:

* реалистичность;
* воспроизводимость;
* внятность;
* соответствие целям и задачам планируемого действия;
* обоснованность;
* результативность.

Существующие в настоящее время методы тестирования ПО не позволяют однозначно и полностью выявить все ошибки и дефекты и установить корректность функционирования анализируемой программы. В связи с тем, что по мере роста программы обнаруживать и исправлять ошибки и дефекты становится все сложнее и дороже, для обеспечения высокого качества ПО тестирование должно выполняться не только как самостоятельный этап разработки ПО, но и заранее, на этапах анализа требований, проектирования и кодирования. Зачастую на тестирование в общей сложности уходит более половины времени, затраченного на проект в целом.

Рассмотрим разные подходы к тестированию ПО.

## 2.1 Каскадная модель (Линейная последовательная модель жизненного цикла ПО)

Каскадная модель (Waterfall Model) является одной из наиболее старых моделей, которую можно применять не только для разработки или тестирования ПО, но также практически для любого другого проекта. Его базовым принципом является последовательный порядок выполнения задач. Это значит, что мы можем переходить к следующему шагу разработки или тестирования только после того, как предыдущий был успешно завершен. Эта модель подходит для небольших проектов и применима только в том случае, если все требования точно определены. Главными достоинствами этой методологии являются экономическая эффективность, простота использования и управления документацией.

Процесс тестирования ПО начинается после завершения процесса разработки. На этой стадии все необходимые тесты переносятся с юнитов на системное тестирование для того, чтобы контролировать работу компонентов как по отдельности, так и в комплексе.



Рисунок 3 – Схема каскадной модели тестирования

Помимо упомянутых выше достоинств, данный подход к тестированию также имеет и свои недостатки. Всегда существует вероятность обнаружения критических ошибок в процессе тестирования. Это может привести к необходимости полностью изменить один из компонентов системы или даже всю логику проекта. Но подобная задача невозможна в случае каскадной модели, поскольку возвращение на предыдущий шаг в этой методологии запрещено.

## 2.2 V-Model (Модель верификации и валидации)

Как и каскадная модель, методика V-Model основана на прямой последовательности шагов. Основным отличием между этими двумя методологиями является то, что тестирование в данном случае планируется параллельно с соответствующей стадией разработки. Согласно этой методологии тестирования ПО, процесс начинается как только определены требования и становится возможным начать статическое тестирование, т.е. верификацию и обзор, что позволяет избежать возможных дефектов ПО на поздних стадиях. Соответствующий план тестирования создается для каждого уровня разработки ПО, что определяет ожидаемые результаты, а также критерии входа и выхода для данного продукта.

Схема данной модели показывает принцип разделения задач на две части. Те, которые относятся к дизайну и разработке, размещены слева. Задачи, относящиеся к тестированию ПО, размещены справа.

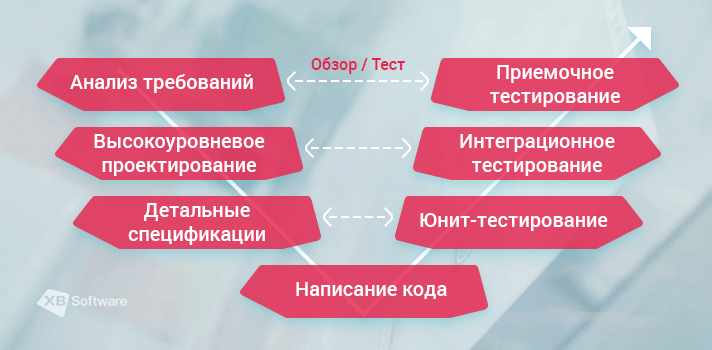


Рисунок 4 – Схема V-Model

Основные этапы этой методологии могут изменяться, однако обычно они включают следующие:

* этап определения требований. Приемочное тестирование относится к этому этапу. Его основная задача состоит в оценке готовности системы к финальному использованию;
* этап, на котором происходит высокоуровневое проектирование, или High-Level Design (HDL). Этот этап относится к системному тестированию и включает оценку соблюдения требований к интегрированным системам;
* фаза детального дизайна (Detailed Design) параллельна фазе интеграционного тестирования, во время которой происходит проверка взаимодействий между различными компонентами системы;
* после этапа написания кода начинается другой важный шаг — юнит-тестирование. Очень важно убедиться в том, что поведение отдельных частей и компонентов ПО корректно и соответствует требованиям.

Единственным недостатком рассмотренной методологии тестирования является отсутствие готовых решений, которые можно было бы применить, чтобы избавиться от дефектов ПО, обнаруженных на этапе тестирования.

## 2.3 Инкрементная модель

Данная методология может быть описана, как мультикаскадная модель тестирования ПО. Рабочий процесс разделяется на некоторое количество циклов, каждый из которых также делится на модули. Каждая итерация добавляет определенный функционал к ПО. Инкремент состоит из трех циклов:

* дизайн и разработка;
* тестирование;
* реализация.

В этой модели возможна одновременная разработка разных версий продукта. Например, первая версия может проходить этап тестирования в то время, как вторая версия находится на стадии разработки. Третья версия в то же самое время может проходить этап дизайна. Этот процесс может продолжаться до самого завершения проекта.

Очевидно, что данная методология требует обнаружения максимально возможного количества ошибок в тестируемом ПО настолько быстро, насколько это возможно. Так же, как и фаза реализации, которая требует подтверждения готовности продукта к доставке к конечному пользователю. Все эти факторы существенно увеличивают весомость требований к тестированию.

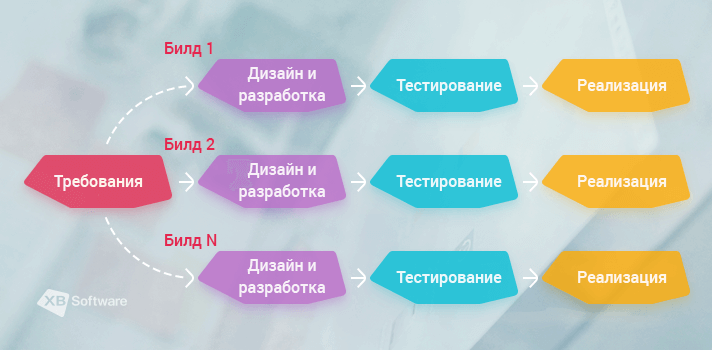


Рисунок 5 – Схема инкрементной модели

В сравнении с предыдущими методологиями, инкрементная модель имеет несколько важных преимуществ. Она более гибкая, изменение требований ведет к меньшим затратам, а процесс тестирования ПО является более эффективным, поскольку гораздо проще проводить тестирование и дебаггинг за счет использования небольших итераций. Тем не менее, стоит отметить, что общая стоимость все же выше, чем в случае каскадной модели.

## 2.4 Спиральная модель

Спиральная модель – это методология тестирования ПО, которая основана на инкрементном подходе и прототипировании. Она состоит из четырех этапов:

* планирование;
* анализ рисков;
* разработка;
* оценка.

Сразу после того, как первый цикл завершен, начинается второй. Тестирование ПО начинается еще на этапе планирования и длится до стадии оценки. Основным преимуществом спиральной модели является то, что первые результаты тестирования появляются незамедлительно после появления результатов тестов на третьем этапе каждого цикла, что помогает гарантировать корректную оценку качества. Тем не менее, важно помнить о том, что эта модель может быть довольно затратной и не подходит для маленьких проектов.



Рисунок 6 – Схема спиральной модели

Несмотря на то, что эта модель является довольно старой, она остается полезной как для тестирования, так и для разработки. Более того, главная цель многих методологий тестирования ПО, включая спиральную модель, изменилась в последнее время, т.е. использование их не только для поиска дефектов в приложениях, но также и для выяснения причин, их вызвавших. Такой подход помогает разработчикам работать более эффективно и быстро устранять ошибки.

## 2.5 Agile

Методология гибкой (Agile) разработки и тестирование ПО может быть описана как набор подходов, ориентированных на использование интерактивной разработки, динамического формирования требований и обеспечения их осуществления как результата постоянного взаимодействия внутри самоорганизующейся рабочей группы. Большинство гибких методологий разработки ПО нацелены на минимизацию рисков посредством разработки в рамках коротких итераций. Одним из главных принципов этой гибкой стратегии является возможность быстрого реагирования на возможные изменения, нежели стремление положиться на долгосрочное планирование.

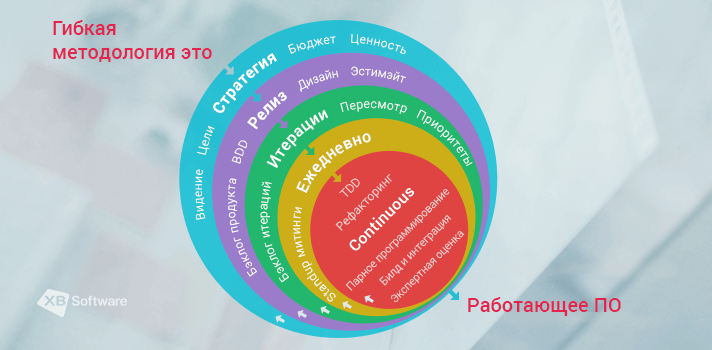


Рисунок 7 – Схема Agile тестирования

## 2.6 Экстремальное программирование (XP, Extreme Programming)

Экстремальное программирование является одним их примеров гибкой разработки ПО. Отличительной особенностью этой методологии является “парное программирование”, ситуация, когда один разработчик работает над кодом, в то время как его коллега постоянно проводит обзор написанного кода. Процесс тестирования ПО является довольно важным, поскольку начинается даже раньше, чем написана первая строка кода. Каждый модуль приложения должен иметь юнит-тест, чтобы большинство ошибок могло быть исправлено на стадии написания кода. Другим отличительным свойством является то, что тест определяет код, а не наоборот. Это значит, что определенная часть кода может быть признана завершенной только в том случае, если все тесты пройдены успешно. В противном случае, код отклоняется.

Главными достоинствами такой методологии являются постоянное тестирование и короткие релизы, что помогает обеспечить высокое качество кода.

## 2.7 Scrum

Scrum — часть методологии Agile, итеративный инкрементный фреймворк, созданный для управления процессом разработки ПО. Согласно принципам Scrum, команда тестировщиков должна участвовать в следующих этапах:

* участие в Scrum планировании;
* поддержка в юнит-тестировании;
* тестирование пользовательских историй;
* сотрудничество с заказчиком и владельцем продукта для определения критериев приемлемости;
* предоставление автоматического тестировании.

Более того, участники QA-отдела должны присутствовать на всех ежедневных собраниях, как и другие члены команды, чтобы обсудить, что было протестировано и сделано вчера, что будет протестировано сегодня, а также общий прогресс тестирования.

В то же время принципы Agile методологии в Scrum привели к появлению специфических особенностей:

* оценка усилий, необходимых для каждой пользовательской истории является обязательной;
* тестировщик должен быть внимательным к требованиям, поскольку они могут постоянно изменяться;
* риск регрессии возрастает вместе с частыми изменениями в коде;
* одновременность планирования и выполнения тестов;
* недопонимание между членами команды в случае если требования заказчика не до конца ясны.

## 3 Анализ требований к тестированию ПО

Исходя из предыдущих разделов, выносятся требования к тестированию ПО:

* тестирование должно повысить вероятность того, что тестируемый продукт будет работать в заданных условиях;
* тестирование должно повысить вероятность того, что тестируемый продукт будет соответствовать требованиям технического задания;
* тестирование должно подтверждать, что после сборки кода (нового или исправленного) устанавливаемый продукт запускается и выполняет основные функции;
* тестирование должно повышать вероятность того, что интерфейс ПО интуитивно понятен и не содержит ошибок, в поля вводятся только определенные типы данных;
* тестирование должно повышать вероятность того, что при отказе какого-либо ресурса приложение пользователя будет работать, то есть отказ какого-либо ресурса не повлияет на работу программного комплекса;
* тестирование должно повышать вероятность того, что каждый компонент в отдельности осуществляется без дефектов;
* тестирование должно повышать вероятность того, что связь между компонентами работает без дефектов;
* тестирование должно повышать вероятность того, что при высокой нагрузке на каналы передачи данных, передача данных будет осуществляться с приемлемой задержкой;
* тестирование должно повышать вероятность того, что при использовании ПО невозможно получить несанкционированный доступ к обрабатываемой информации.

## ЗАКЛЮЧЕНИЕ

В ходе решения поставленной задачи были рассмотрены существующие методики тестирования ПО, проведен их анализ и выявлены присущие им недостатки. Также в данной работе были сформулированы требования к тестированию ПО.

Значимость работы заключается в том, что на основе полученных знаний:

1) можно в короткий срок ознакомиться с необходимой теорией по тестированию ПО, которая может помочь в успешном прохождении технического собеседования для устройства на работу специалистом по тестированию;

2) по собранному материалу можно оформить методическое пособие для студентов и включить в программу обучения в качестве дополнительного курса «Тестирование ПО».

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) IEEE Standart for Software Test Documentation. [Электронный ресурс] <https://standards.ieee.org/findstds/standard/829-1983.html> – дата обращения 7.11.18

2) Савин Р. Тестирование Дот Ком, или пособие по жестокому обращению с багами в интернет-стартапах/Р.Савин – М.: Дело, 2007. – 312 с.

3) Котляров В.П. Основы современного тестирования программного обеспечения: учебное пособие/В.П.Котляров, Т.В.Коликова – СПб.: Питер, 2004. – 170 с.

4) Липаев В.В. Тестирование крупных комплексов программ на соответствие требованиям: учебник/В.В.Липаев – М.: ИПЦ «Глобус», 2008. – 316 с.

5) Модели жизненного цикла программного обеспечения. [Электронный ресурс] <https://habrahabr.ru/post/111674.html> – дата обращения 9.11.18

6) Сэм Канер. Testing Computer Software/Сэм Канер, Джек Фолк. –М.:ДиаСофт, 2001. – 544 с.

7) Уровни тестирования программного обеспечения. [Электронный ресурс] <http://www.protesting.ru/testing/testlevels.html> – дата обращения 11.11.18

8) Коджаспирова Г.М., Коджаспиров А.Р. Словарь по педагогике/Г.М.Коджаспирова, А.Р.Коджаспиров. – Ростов н/Д.:ИКЦ «Март», 2005. – 320с.