



1920

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Филиал федерального государственного бюджетного образовательного учреждения
высшего образования

«Кубанский государственный университет» в г. Геленджике

УТВЕРЖДАЮ

Проректор по работе с филиалами


« 24 »  А.А. Рылов

Рабочая программа дисциплины

МДК.01.02 «Поддержка и тестирование модулей программного обеспечения»

09.02.07 «Информационные системы и программирование»

Рабочая программа МДК.01.02 «Поддержка и тестирование модулей программного обеспечения» разработана на основе федерального государственного образовательного стандарта среднего профессионального образования (ФГОС СПО) по специальности 09.02.07 «Информационные системы и программирование», утвержденного приказом Министерства образования и науки от 9 декабря 2016 года № 1547 (зарегистрирован Министерством юстиции Российской Федерации 26 декабря 2016г., регистрационный №44936) (далее – ФГОС СПО).

Дисциплина МДК.01.02 «Поддержка и тестирование модулей программного обеспечения»

Форма обучения	очная
Учебный год	2024-2025
2,3 курсы	4, 5 семестр
Лекции	110 час.
Практические занятия	62 час.
Консультации	4 час.
Экзамен	6 час.
форма итогового контроля	дифференцированный зачёт, экзамен

Составитель: преподаватель

Поддубная Е.А.

Утверждена на заседании предметной (цикловой) комиссии профессиональных дисциплин программирования в компьютерных системах

Протокол № 10 от 24 мая 2024 г.

Председатель предметной (цикловой) комиссии профессиональных дисциплин специальности **09.02.07 «Информационные системы и программирование» и 09.02.03 «Программирование в компьютерных системах»**

Л.А. Благова

подпись

Рецензенты:

Заместитель директора по научной работе филиала ФГБОУ ВО «КубГУ» в г.

Геленджике



Л.В. Галицкая, к.т.н.


Директор ООО «Современные
информационные технологии»




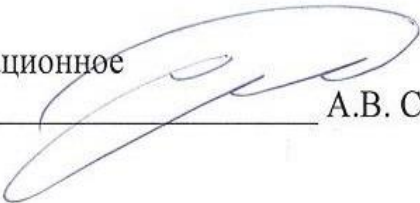
А.В.Сметанин

ЛИСТ
согласования рабочей учебной программы по дисциплине
МДК.01.02 «Поддержка и тестирование модулей программного обеспечения»
Специальность среднего профессионального образования:
09.02.03 Программирование в компьютерных системах

СОГЛАСОВАНО:

Заместитель директора по УР филиала _____  Т.А. Резуненко

Заведующая сектором библиотеки филиала _____  Л.Г. Соколова

Инженер-электроник (программно-информационное
обеспечение образовательной программы) _____  А.В. Сметанин

СОДЕРЖАНИЕ

1. ПАСПОРТ РАБОЧЕЙ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ	4
1.1. Область применения программы	4
1.2. Место учебной дисциплины в структуре программы подготовки специалистов среднего звена	4
1.3. Цели и задачи учебной дисциплины - требования к результатам освоения дисциплины	4
1.4. Перечень планируемых результатов обучения по дисциплине (перечень формируемых компетенций)	5
2. СТРУКТУРА И СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ	9
2.1. Объем учебной дисциплины и виды учебной работы	9
2.2. Структура дисциплины:	9
2.3. Тематический план и содержание учебной дисциплины	10
2.4. Содержание разделов дисциплины	12
2.4.1. Занятия лекционного типа	12
2.4.2. Занятия семинарского типа	12
2.4.3. Практические (лабораторные) занятия	12
2.4.4. Содержание самостоятельной работы	13
2.4.5. Содержание учебно-методического обеспечения для самостоятельной работы	
3. ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ	16
3.1. Образовательные технологии при проведении лекция	16
3.2. Образовательные технологии при проведении практических занятий	16
4. УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ ДИСЦИПЛИНЫ	18
4.1. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине	18
4.2. Информационное обеспечение реализации программы	18
5. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ УЧЕБНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ	21
5.1. Основная литература	21
5.2. Дополнительная литература	22
5.3. Периодические издания	23
5.4. Перечень ресурсов информационно-телекоммуникационной сети Интернет, необходимых для освоения дисциплины	
6. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ	25
7. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ КОНТРОЛЯ УСПЕВАЕМОСТИ	30
7.1. Паспорт фонда оценочных средств	30
7.2. Критерии оценки знаний	30
7.3. Оценочные средства для проведения текущей аттестации	31
7.4. Оценочные средства для проведения промежуточной аттестации	32
7.4.1. Примерные вопросы для проведения промежуточной аттестации	32
7.4.2. Примерные задачи для проведения промежуточной аттестации	40
8. ДОПОЛНИТЕЛЬНОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ	51

1. ПАСПОРТ РАБОЧЕЙ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ

МДК.01.02 «Поддержка и тестирование программных модулей»

1.1 Область применения программы

Рабочая программа МДК.01.02 «Поддержка и тестирование программных модулей» является частью программы подготовки специалистов среднего звена в соответствии с ФГОС СПО по специальности 09.02.07. «Информационные системы и программирование»

1.2 Место учебной дисциплины в структуре программы подготовки специалистов среднего звена

Учебная дисциплина МДК.01.02 «Поддержка и тестирование программных модулей» принадлежит профессиональному модулю ПМ.01. «Разработка модулей программного обеспечения для компьютерных систем». Она обеспечивает профессиональный уровень подготовки специалиста и соответствует развитию их профессионально значимых качеств.

Для освоения дисциплины обучающиеся используют знания, умения и навыки, сформированные при изучении МДК.01.01 «Разработка программных модулей» и ОП.04 «Основы алгоритмизации и программирования».

1.3 Цель и планируемые результаты освоения дисциплины:

Для основного вида деятельности «Разработка модулей программного обеспечения для компьютерных систем» осваиваются частично ПК.1.3. «Выполнять отладку программных модулей с использованием специализированных программных средств», для основного вида деятельности «Осуществление интеграции программных модулей» осваиваются частично компетенции ПК 2.1. «Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент», ПК 2.3. «Выполнять отладку программного модуля с использованием специализированных программных средств», ПК 2.4. «Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения», ПК 4.2. Осуществлять измерения эксплуатационных характеристик программного обеспечения компьютерных систем

Максимальная учебная нагрузка обучающегося – 182 часа, в том числе

- обязательная аудиторная нагрузка – 172 часа;
- самостоятельная работа обучающегося – не предусмотрено
- консультации – 4 часа;
- экзамен – 6 часов

1.4. Перечень планируемых результатов обучения по дисциплине (перечень формируемых компетенций)

В результате освоения учебной дисциплины обучающийся должен

уметь:

- применять логические операции, формулы логики, законы алгебры логики;
- формулировать задачи логического характера и применять средства математической логики для их решения.

В результате освоения учебной дисциплины обучающийся должен

знать:

- основные принципы математической логики, теории множеств и теории алгоритмов;
- формулы алгебры высказываний;
- методы минимизации алгебраических преобразований;
- основы языка и алгебры предикатов;
- основные принципы теории множеств.

В результате освоения учебной дисциплины обучающийся осваивает элементы общих компетенций.

1.4.1.Перечень общих компетенция

Код компетенции	Формулировка компетенции	Знания, умения ⁴
ОК 01	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам	<p>Умения: распознавать задачу и/или проблему в профессиональном и/или социальном контексте; анализировать задачу и/или проблему и выделять её составные части; определять этапы решения задачи; выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы; составить план действия; определить необходимые ресурсы; владеть актуальными методами работы в профессиональной и смежных сферах; реализовать составленный план; оценивать результат и последствия своих действий (самостоятельно или с помощью наставника)</p> <p>Знания: актуальный профессиональный и социальный контекст, в котором приходится работать и жить; Основные источники информации и ресурсы для решения задач и проблем в профессиональном и/или социальном контексте; алгоритмы выполнения работ в профессиональной и смежных областях; методы работы в профессиональной и смежных сферах; структуру плана для решения задач; деятельности</p>

ОК 02	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности	Умения: определять задачи для поиска информации; определять необходимые источники информации; планировать процесс поиска; структурировать получаемую информацию; выделять наиболее значимое в перечне информации; оценивать практическую значимость результатов поиска; оформлять результаты поиска
		Знания: номенклатура информационных источников, Применяемых в профессиональной деятельности; Приемы структурирования информации; формат оформления результатов поиска информации

ОК 03	Планировать и реализовывать собственное профессиональное и личностное развитие.	Умения: определять актуальность нормативно-правовой документации в профессиональной деятельности; применять современную научную профессиональную терминологию; определять и выстраивать траектории профессионального развития и самообразования
		Знания: содержание актуальной нормативно-правовой документации; современная научная и профессиональная профессионального развития и самообразования
ОК 04	Работать в коллективе и команде, эффективно взаимодействовать с коллегами, Руководством, клиентами	Умения: организовывать работу коллектива и команды; Взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности
		Знания: психологические основы деятельности коллектива, психологические особенности личности; основы проектной деятельности
ОК 05	Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.	Умения: грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке, проявлять толерантность в рабочем коллективе
		Знания: особенности социального и культурного контекста; правила оформления документов и построения устных сообщений.
ОК 09	Использовать информационные технологии в профессиональной деятельности	Умения: применять средства информационных технологий для решения профессиональных задач; использовать современное программное обеспечение
		Знания: современные средства и устройства информатизации; порядок их применения и программное обеспечение в профессиональной деятельности

1.4.2. Перечень профессиональных компетенций

В результате освоения учебной дисциплины обучающийся осваивает элементы профессиональных компетенций: ПК 1.3, 2.1, 2.3, 2.4, 4.2.

Перечень профессиональных компетенций, элементы которых формируются в рамках учебной дисциплины

Код	Наименование видов деятельности и профессиональных компетенций
ОВД 1	Разработка модулей программного обеспечения для компьютерных систем
ПК 1.3	Выполнять отладку программных модулей с использованием специализированных программных средств
ОВД 2	Осуществление интеграции программных модулей
ПК 2.1	Выполнять тестирование программных модулей
ПК 2.3.	Выполнять отладку программного модуля с использованием специализированных программных средств
ПК 2.4	Осуществлять рефакторинг и оптимизацию программного кода
ОВД 4	Сопровождение и обслуживание программного обеспечения компьютерных систем
ПК 4.2.	Осуществлять измерения эксплуатационных характеристик программного обеспечения компьютерных систем
ПК 5.1.	Собирать исходные данные для разработки проектной документации на информационную систему

В результате освоения МДК 01.02 студент должен:

Иметь практический опыт	Осуществлять измерения эксплуатационных характеристик программного обеспечения компьютерных систем. Разрабатывать и оформлять требования к программным модулям по предложенной документации. Разрабатывать тестовые наборы (пакеты) для программного модуля. Разрабатывать тестовые сценарии программного средства. Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования. Отлаживать программные модули. Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования. Разрабатывать тестовые наборы (пакеты) для программного модуля. Разрабатывать тестовые сценарии программного средства. Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования. Измерять эксплуатационные характеристики программного обеспечения компьютерных систем на соответствие требованиям
уметь	Выполнять отладку и тестирование программы на уровне модуля. Оформлять документацию на программные средства. Анализировать проектную и техническую документацию. Использовать специализированные графические средства построения анализа архитектуры программных продуктов. Организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов. Определять источники и приемники данных. Проводить сравнительный анализ. Выполнять отладку, используя методы и инструменты условной компиляции (классы Debug и Trace). Оценивать размер

	<p>минимального набора тестов. Разрабатывать тестовые пакеты и тестовые сценарии. Выявлять ошибки в системных компонентах на основе спецификаций. Использовать выбранную систему контроля версий. Использовать методы для получения кода с заданной функциональностью и степенью качества. Анализировать проектную и техническую документацию. Использовать инструментальные средства отладки программных продуктов. Определять источники и приемники данных. Выполнять тестирование интеграции. Организовывать постобработку данных. Использовать приемы работы в системах контроля версий. Выполнять отладку, используя методы и инструменты условной компиляции. Выявлять ошибки в системных компонентах на основе спецификаций.</p> <p>Использовать выбранную систему контроля версий. Анализировать проектную и техническую документацию. Выполнять тестирование интеграции. Организовывать постобработку данных. Использовать приемы работы в системах контроля версий. Оценивать размер минимального набора тестов. Разрабатывать тестовые пакеты и тестовые сценарии. Выполнять ручное и автоматизированное тестирование программного модуля. Выявлять ошибки в системных компонентах на основе спецификаций. Измерять и анализировать эксплуатационные характеристики качества программного обеспечения</p>
<p>знать</p>	<p>Модели процесса разработки программного обеспечения. Основные принципы процесса разработки программного обеспечения. Основные подходы к интегрированию программных модулей. Виды и варианты интеграционных решений. Современные технологии и инструменты интеграции. Основные протоколы доступа к данным. Методы и способы идентификации сбоев ошибок при интеграции приложений. Методы отладочных классов. Стандарты качества программной документации. Основы организации инспектирования и верификации. Встроенные и основные специализированные инструменты анализа качества программных продуктов. Графические средства проектирования архитектуры программных продуктов. Методы организации работы в команде разработчиков. Модели процесса разработки программного обеспечения. Основные принципы процесса разработки программного обеспечения. Основные подходы к интегрированию программных модулей. Основы верификации и аттестации программного обеспечения. Методы и способы идентификации сбоев и ошибок при интеграции приложений. Основные методы отладки. Методы и схемы обработки исключительных ситуаций. Приемы работы с инструментальными средствами тестирования и отладки. Стандарты качества программной документации. Основы организации инспектирования и верификации. Встроенные и основные специализированные инструменты анализа качества программных продуктов. Методы организации работы в команде разработчиков. Модели процесса разработки программного обеспечения. Основные принципы процесса разработки программного обеспечения. Основные подходы к интегрированию программных модулей. Основы верификации и аттестации программного обеспечения. Методы и способы идентификации сбоев и ошибок при интеграции приложений. Методы и схемы обработки исключительных ситуаций. Основные методы и виды тестирования программных продуктов. Приемы работы с инструментальными средствами тестирования и отладки.</p>

	Стандарты качества программной документации. Основы организации инспектирования и верификации. Встроенные и основные специализированные инструменты анализа качества программных продуктов. Методы организации работы в команде разработчиков. Основные методы и средства эффективного анализа функционирования программного обеспечения. Основные принципы контроля конфигурации и поддержки целостности конфигурации ПО
--	---

В результате освоения учебной дисциплины обучающийся осваивает элементы основных видов деятельности:

Основной вид деятельности	Требования к знаниям, умениям, практическим действиям
ОВД 1. «Разработка модулей программного обеспечения для компьютерных систем»	Указаны выше
ОВД 2. «Осуществление интеграции программных модулей»	Указаны выше
ОВД 4 «Сопровождение и обслуживание программного обеспечения компьютерных систем»	Указаны выше

Технологии формирования ОК

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

Предоставить обучающемуся возможность на практических занятиях для самостоятельного выбора способа решения задачи

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

Предоставить обучающемуся возможность на практических занятиях для самостоятельного выбора способа подготовки информации для решения задачи

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие.

Предоставить обучающемуся возможность на лекциях и практических занятиях для выражать свою заинтересованность в обучении и реализовывать ее

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

Предоставить обучающемуся возможность на практических занятиях, на учебной и производственной практиках для решения задач в коллективе, помогать в случае затруднений

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

Предоставить обучающемуся возможность на лекциях практических занятиях для общения на государственном языке

ОК 09. Использовать информационные технологии в профессиональной деятельности.

Предоставить обучающимся возможности на практических занятиях выполнять задания средствами ИТ. Предоставлять студентам возможность самостоятельно осуществлять поиск, анализ и оценку информации при выполнении практической и самостоятельной работы

Изучать профессиональные стандарты на государственном языке (русском) и знакомить обучающихся с международными стандартами на английском языке

Технологии формирования ПК

Изучение теории ручного и автоматизированного тестирования и закрепления ее на опыте

2 ь СТРУКТУРА И СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ

2.1 Объем учебной дисциплины и виды учебной работы

Вид учебной работы	Объем в часах
Объем образовательной программы	182
в том числе:	
теоретическое обучение (+ текущая аттестация – зачёт)	110
практические занятия	62
консультации	4
Промежуточная аттестация (экзамен)	6

2.2 Структура дисциплины

Наименование разделов и тем	Всего	Количество аудиторных часов		Самостоятельная работа обучающегося (час)
		Теоретическое обучение	Практические и лабораторные занятия	
Раздел 1. Отладка и тестирование программного обеспечения	118	68	50	-
Тема 1.1. Понятие разработки ПО	8	6	2	-
Тема 1.2. Понятие тестирования	8	6	2	-
Тема 1.3. Тестирование требований	14	6	8	-
Тема 1.4. Классификация тестирования	26	14	12	
Тема 1.5. Отчёты о тестировании	20	12	8	
Тема 1.6. Методы отладки и тестирования	18	12	6	
Тема 1.7. Автоматизированное тестирование	24	12	12	
Раздел 2. Документирование	54	42	12	
Тема 2.1. Документирование программного обеспечения в соответствии с Единой системой программной документации	12	8	4	
Тема 2.2. Документирование основных процессов ЖЦ ПО	16	12	4	

Тема 2.3. Документирование вспомогательных процессов ЖЦ ПО	14	12	2	
Тема 2.4. Документирование сертификации	12	10	2	
Всего по дисциплине	172	110	62	

2.3 Тематический план и содержание учебной дисциплины МДК.01.02 «Поддержка и тестирование модулей программного обеспечения»

МДК.01.02 Поддержка и тестирование программных модулей			
Раздел 1. Отладка и тестирование программного обеспечения			
Тема 1.1. Понятие разработки ПО	Содержание учебного материала	Уровень освоения	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	1.Лекция 1. Понятие программного обеспечения и его классификация	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	2.Лекция 2. Жизненный цикл ПО.	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	3. Лекция 3. Этапы разработки ПО	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Практические занятия		
	4.Практическая работа 1. Проверочная работа по теме 1.1.	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
Тема 1.2. Понятие тестирования	Содержание учебного материала		
	5. Лекция 4. Тестирование как часть процесса верификации программного обеспечения.	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	6. Лекция 5. Обязанности тестировщика	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	7. Лекция 6. Жизненный цикл тестирования	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Практические занятия		
	8. Практическая работа 2. Проверочная работа по теории	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
Тема 1.3. Тестирование требований	Содержание учебного материала		
	9.Лекция 7. Понятие требований и их качество (уровни и типы)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	10. Лекция 8. Техника тестирования требований	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	11.Лекция 9. Типичные ошибки при анализе и тестировании требований и причины их появления	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Практические занятия		ПК 1.3, 2.1, 2.3, 2.4, 5.1
	12. Практическая работа 3. Проверочная работа по теории	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	13. Практическая работа 4. Составление требований к программе	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	14. Практическая работа 5. Проверка составленных требований	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
15. Практическая работа 6. Анализ выявленных ошибок	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1	
Тема 1.4. Классификация тестирования	Содержание учебного материала		
	16. Лекция 10. Классификация тестирования	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	17. Лекция 11. Виды тестирования (по запуску кода на исполнение, по доступу к коду и	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1

	архитектуре приложения, по степени автоматизации, по уровню тестирования)		
	18. Лекция 12. Виды тестирования (уровню функционального тестирования, по принципам работы с приложением, по природе приложения, по привлечению конечных пользователей)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	19. Лекция 13. Виды тестирования (по степени формализации, по целям и задачам, по техникам и подходам, по хронологии)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	20. Лекция 14. Виды тестирования (по знанию системы, по исполнителям, по времени проведения, по производительности)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	21. Лекция 15. Альтернативные и дополнительные классификации	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	22. Лекция 16. Классификация по принадлежности к методам «белого» и «чёрного» ящикам	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Практические занятия		
	23. Практическая работа 7. Проверочная работа	3	
	24. Практическая работа 8. Тестирование программы методом «белого ящика»	3	
	25. Практическая работа 9. Перекрёстная проверка результатов тестирования	3	
	26. Практическая работа 10. Тестирование программы методом «чёрного» ящика.	3	
	27. Практическая работа 11. Перекрёстная проверка результатов тестирования	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
Тема 1.5. Отчёты о тестировании	Содержание учебного материала		
	28. Лекция 17. Виды ошибок (ошибки (синтаксические, компоновки, выполнения, определения данных, накопления погрешностей, проектирования), дефекты, сбои, отказы и т.д.)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	29. Лекция 18. Отчёт о дефектах, его атрибуты и ЖЦ	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	30. Лекция 19. Инструментальные средства управления отчётами о дефектах	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	31. Лекция 20. Свойства качественных отчётов и логика их создания. Типичные ошибки при написании отчёта о дефектах	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	32. Лекция 21. Планирование тестирования (тест-план)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	33. Лекция 22. Оценка трудозатрат на планирование и тестирование	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Практические занятия		
	34. Практическая работа 12. Проверочная работа	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	35. Практическая работа 13. Составление проекта отчёта о дефектах	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	36. Практическая работа 14. Перекрёстный анализ	3	ПК 1.3, 2.1,

	составленных отчётов		2.3, 2.4, 5.1
	37. Практическая работа 15. Составление тест-плана	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	38. Практическая работа 16. Оценка трудозатрат на планирование и тестирование по составленным планам	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
Тема 1.6. Методы отладки и тестирования	Содержание учебного материала		
	39. Лекция 23. Методы отладки: (ручного тестирования, индукции, дедукции, обратного прослеживания)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	40. Лекция 24. Методы тестирования (позитивные и негативные тест-кейсы, классы эквивалентности и граничные условия)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	41. Лекция 25. Методы тестирования (доменное тестирование и комбинация параметров)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	42. Лекция 26. Методы тестирования (попарное тестирование и поиск комбинаций, исследовательское тестирование)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	43. Лекция 27. Регрессионное тестирование и его виды	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	44. Лекция 28. Поиск причин возникновения дефектов	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Практические занятия		
	45. Практическая работа 17. Проверочная работа по теории методов тестирования	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	46. Практическая работа 18. Отладка программы всеми известными методами и составление отчёта о результатах	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	47. Практическая работа 19. Тестирование одной программы 9-ю методами и составление отчёта о результатах	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
Тема 1.7. Автоматизированное тестирование	Содержание учебного материала		
	48. Лекция 29. Автоматизация тестирования, её задачи, преимущества и ограничения. Пирамида тестирования.	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	49. Лекция 30. Этапы автоматизированного тестирования	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	50. Лекция 31. Разработка тест-кейсов. Сценарный тест	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	51. Лекция 32. Виды автоматизированного тестирования (регрессионное, кроссбраузерное и кроссплатформенное, локации и производительности)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	52. Лекция 33. Технологии автоматизации тестирования, требования к автоматизированному тесту.	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Дифференцированный зачёт		

	53. Лекция 34. Бесплатные инструменты автоматизированного тестирования (SilkTest, HP Quick Test Professional, Test Complete, Selenium и т.д.)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Практические занятия		
	54. Практическая работа 20. Проверочная работа по теории автоматизированного тестирования	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	55. Практическая работа 21. Разработка тест-кейсов и сценариев тестирования для программы	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	56. Практическая работа 22. Ознакомление с инструментами автоматизированного тестирования	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	57. Практическая работа 23. Изучение интерфейса инструмента автоматизированного тестирования	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	58. Практическая работа 24. Выполнение автоматизированного тестирования программы и составление отчета	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	59. Практическая работа 25. Защита отчёта о результатах автоматизированного тестирования	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
Раздел 2. Документирование			
Тема 2.1. Документирование программного обеспечения в соответствии с Единой системой программной документации	Содержание учебного материала		
	60. Лекция 35. Понятие стандартизации	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	61. Лекция 36. Структура ЕСПД	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	62. Лекция 37. Структура стандарта в ЕСПД.	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	63. Лекция 38. Стандарты документирование сложных ПС (международные ISO 9294, 12182, 12207, 15910, 9127; IEEE1063 -1987)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Практические занятия		
	64. Практическая работа 26. Проверочная работа по теории документирования по ЕСПД	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
65. Практическая работа 27. Проверочная работа по теории документирования по международным стандартам	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1	
Тема 2.2. Документирование основных процессов ЖЦ ПО	Содержание учебного материала		
	66. Лекция 39. Документирование предварительных требований заказчика, обследование и описание системы, целей разработки ПС, концепция и основные предложения по созданию базовой версии, предварительный план разработки базовой версии,	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	67. Лекция 40. Техническое задание	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	68. Лекция 41. Документирование проектирования и выбора характеристик качества ПС, план обеспечения качества ПС	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	69. Лекция 42. Спецификация требований к	2	ПК 1.3, 2.1,

		системе и комплексу программ		2.3, 2.4, 5.1
		70. Лекция 43. Пояснительная записка к предварительному или детальному проекту программного средства.	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
		71. Лекция 44. Документирование процесса разработки и программирования компонент программных средств	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
		Практические занятия		
		72. Практическая работа 28. Составление проекта разработки программного средства	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
		73. Практическая работа 29. Защита проекта разработки программного средства	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
		Содержание учебного материала		
		74. Лекция 45. Руководство программистам	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
		75. Лекция 46. Обеспечение верификации и тестирования ПС, план, исходные данные и отчёты о результатах. Акт приёма ПС в промышленную эксплуатацию	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
		76. Лекция 47. Документирование верификации, отчёт о результатах разработки, акт завершения работ по проекту, акт приёмки ПС в промышленную эксплуатацию	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
		77. Лекция 48. Документирование сопровождения и конфигурационного управления версиями ПС (описание среды ЖЦ и конфигурации ПС, отчеты пользователей о выявленных дефектах и предложения по их корректировке, описание корректировок и результатов их тестирования, протоколы о выпуске новой версии ПС, отчет о результатах эксплуатации снятой с сопровождения версии)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
Тема 2.3.	Документирование вспомогательных процессов ЖЦ ПО	78. Лекция 49. Документирование эксплуатации ПС (инструкция по формированию и ведению информации БД, Паспорт на ПС, руководство по обучению специалистов применению ПС)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
		79. Лекция 50. Требования к формированию Пользовательской документации (руководство системного администратора, руководство пользователя)	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
		Практические занятия		
		80. Практическая работа 30. Составление руководств системного администратора и пользователя	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
Тема 2.4.	Документирование	Содержание учебного материала		ПК 1.3, 2.1, 2.3, 2.4, 5.1
		81. Лекция 51. Сущность сертификации	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1

сертификации	82. Лекция 52. Проведение сертификации	2	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	83. Лекция 53. Правовые и организационно-методические основы сертификации		ПК 1.3, 2.1, 2.3, 2.4, 5.1
	84. Лекция 54. Деятельность ИСО и МЭК в области сертификации.		ПК 1.3, 2.1, 2.3, 2.4, 5.1
	85. Лекция 55. Документирование сертификации		ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Практические занятия		
	86. Практическая работа 31. Оформление документации на программные средства с использованием инструментальных средств	3	ПК 1.3, 2.1, 2.3, 2.4, 5.1
	Всего по дисциплине	172	

2.4 Содержание разделов дисциплины

2.4.1. Занятия лекционного типа

Номер раздела	Наименование раздела	Содержание раздела	Форма текущего контроля
Раздел 1. Отладка и тестирование программного			
	Тема 1.1. Понятие разработки ПО	1. Понятие программного обеспечения и его классификация	Т, У
		2. Жизненный цикл ПО.	Т, У
		3. Этапы разработки ПО	Т, У
	Тема 1.2. Понятие тестирования	4. Тестирование как часть процесса верификации программного обеспечения.	Т, У
		5. Обязанности тестировщика	Т, У
		6. Жизненный цикл тестирования	Т, У
	Тема 1.3. Тестирование требований	7. Понятие требований и их качество (уровни и типы)	Т, У
		8. Техника тестирования требований	Т, У
		9. Типичные ошибки при анализе и тестировании требований и причины их появления	Т, У
	Тема 1.4. Классификация тестирования	10. Классификация тестирования	Т, У
		11. Виды тестирования (по запуску кода на исполнение, по доступу к коду и архитектуре приложения, по степени автоматизации, по уровню тестирования)	Т, У
		12. Виды тестирования (уровню функционального тестирования, по принципам работы с приложением, по природе приложения, по привлечению конечных пользователей)	Т, У
		13. Виды тестирования (по степени формализации, по целям и задачам, по техникам и подходам, по хронологии)	Т, У
		14. Виды тестирования (по знанию системы, по исполнителям, по времени проведения, по производительности)	Т, У
		15. Альтернативные и дополнительные классификации	Т, У
		16. Классификация по принадлежности к методам «белого» и «чёрного» ящикам	Т, У

Тема 1.5. Отчёты о тестировании	17. Виды ошибок (ошибки (синтаксические, компоновки, выполнения, определения данных, накопления погрешностей, проектирования), дефекты, сбои, отказы и т.д.)	Т, У	
	18. Отчёт о дефектах, его атрибуты и ЖЦ	Т, У	
	19. Инструментальные средства управления отчётами о дефектах	Т, У	
	20. Свойства качественных отчётов и логика их создания. Типичные ошибки при написании отчёта о дефектах	Т, У	
	21. Планирование тестирования (тест-план)	Т, У	
	22. Оценка трудозатрат на планирование и тестирование	Т, У	
	Тема 1.6. Методы отладки и тестирования	23. Методы отладки: (ручного тестирования, индукции, дедукции, обратного прослеживания)	Т, У
		24. Методы тестирования (позитивные и негативные тест-кейсы, классы эквивалентности и граничные условия)	Т, У
		25. Методы тестирования (доменное тестирование и комбинация параметров)	Т, У
		26. Методы тестирования (попарное тестирование и поиск комбинаций, исследовательское тестирование)	Т, У
		27. Регрессионное тестирование и его виды	Т, У
		28. Поиск причин возникновения дефектов	Т, У
	Тема 1.7. Автоматизированное тестирование	29. Автоматизация тестирования, её задачи, преимущества и ограничения. Пирамида тестирования.	Т, У
		30. Этапы автоматизированного тестирования	Т, У
		31. Разработка тест-кейсов. Сценарный тест	Т, У
		32. Виды автоматизированного тестирования (регрессионное, кроссбраузерное и кроссплатформенное, локации и производительности)	Т, У
		33. Технологии автоматизации тестирования, требования к автоматизированному тесту. Дифференцированный зачёт	
		34. Бесплатные инструменты автоматизированного тестирования (SilkTest, HP Quick Test Professional, Test Complete, Selenium и т.д.)	Т, У
	2Раздел 2. Документирование		
	Тема 2.1. Документирование программного обеспечения в соответствии с ЕСПД	35. Понятие стандартизации	Т, У
		36. Структура ЕСПД	Т, У
		37. Структура стандарта в ЕСПД.	Т, У
38. Стандарты документирования сложных ПС (международные ISO 9294, 12182, 12207, 15910, 9127; IEEE1063 -1987)		Т, У	
Тема 2.2. Документирование основных процессов ЖЦ ПО	39. Документирование предварительных требований заказчика, обследование и описание системы, целей разработки ПС, концепция и основные предложения по созданию базовой версии, предварительный план разработки базовой версии,	Т, У	
	40. Техническое задание	Т, У	
	41. Документирование проектирования и выбора характеристик качества ПС, план обеспечения качества ПС	Т, У	
	42. Спецификация требований к системе и комплексу	Т, У	

		программ	
		43. Пояснительная записка к предварительному или детальному проекту программного средства.	Т, У
		44. Документирование процесса разработки и программирования компонент программных средств	Т, У
Тема 2.3. Документирование вспомогательных процессов ЖЦ ПО		45. Руководство программистам	Т, У
		46. Обеспечение верификации и тестирования ПС, план, исходные данные и отчёты о результатах. Акт приёма ПС в промышленную эксплуатацию	Т, У
		47. Документирование верификации, отчёт о результатах разработки, акт завершения работ по проекту, акт приёмки ПС в промышленную эксплуатацию	Т, У
		48. Документирование сопровождения и конфигурационного управления версиями ПС (описание среды ЖЦ и конфигурации ПС, отчеты пользователей о выявленных дефектах и предложения по их корректировке, описание корректировок и результатов их тестирования, протоколы о выпуске новой версии ПС, отчет о результатах эксплуатации снятой с сопровождения версии)	Т, У
		49. Документирование эксплуатации ПС (инструкция по формированию и ведению информации БД, Паспорт на ПС, руководство по обучению специалистов применению ПС)	Т, У
		50. Требования к формированию Пользовательской документации (руководство системного администратора, руководство пользователя)	Т, У
Тема 2.4. Документирование сертификации		51. Сущность сертификации	Т, У
		52. Проведение сертификации	Т, У
		53. Правовые и организационно-методические основы сертификации	Т, У
		54. Деятельность ИСО и МЭК в области сертификации.	Т, У
		55. Документирование сертификации	Т, У
Примечание: Т – тестирование, Р – написание реферата, У – устный опрос, КР – контрольная работа			

2.4.2. Занятия семинарского типа

Не предусмотрено

2.4.3. Практические занятия

Номер раздела	Наименование раздела	Содержание раздела	Форма текущего контроля
1	2	3	4
1	Раздел 1. Отладка и тестирование программного обеспечения		
	Тема 1.1. Понятие разработки ПО	1. Проверочная работа по теме 1.1	
	Тема 1.2. Понятие тестирования	1. Проверочная работа по теме 1.2	
	Тема 1.3. Тестирование требований	3. Проверочная работа по теории	ПР, У
		4. Составление требований к программе	ПР, У
		5. Проверка составленных требований	ПР, У
		6. Анализ выявленных ошибок	ПР, У

	Тема 1.4. Классификация тестирования	8.Тестирование программы методом «белого ящика»	ПР, У	
		9.Перекры́стная проверка результатов тестирования	ПР, У	
		10.Тестирование программы методом «чёрного» ящика.	ПР, У	
		11.Перекры́стная проверка результатов тестирования	ПР, У	
	Тема 1.5. Отчёты тестирования	13.Составление проекта отчёта о дефектах	ПР, У	
		14.Перекры́стный анализ составленных отчётов	ПР, У	
		15. Составление тест-плана	ПР, У	
	Тема 1.6. Методы отладки и тестирования	16. Оценка трудозатрат на планирование и тестирование по составленным планам	ПР, У	
		17.Проверочная работа по теории методов тестирования	ПР, У	
		18.Отладка программы всеми известными методами и составление отчёта о результатах	ПР, У	
	Тема 1.7. Автоматизи- рованное тестирование	19.Тестирование одной программы 9-ю методами и составление отчёта о результатах	ПР, У	
		20.Проверочная работа по теории автоматизированного тестирования	ПР, У	
		21.Разработка тест-кейсов и сценариев тестирования для программы	ПР, У	
		22. Ознакомление с инструментами автоматизированного тестирования	ПР, У	
		23.Изучение интерфейса инструмента автоматизированного тестирования	ПР, У	
		24.Выполнение автоматизированного тестирования программы и составление отчета	ПР, У	
		25.Защита отчёта о результатах автоматизированного тестирования	ПР, У	
	Раздел 2. Документирование			
		Тема 2.1. Документирование программного обеспечения в соответствии с ЕСПД	26. Проверочная работа по теории документирования по ЕСПД	ПР, У
			27.Проверочная работа по теории документирования по международным стандартам	ПР, У
		Тема 2.2. Документирование основных процессов ЖЦ ПО	28.Составление проекта разработки программного средства	ПР, У
			29.Защита проекта разработки программного средства	ПР, У
		Тема 2.3. Документирование вспомогательных процессов ЖЦ ПО	30.Составление руководств системного администратора и пользователя	ПР, У
		Тема 2.4. Документирование сертификации	31.Оформление документации на программные средства с использованием инструментальных средств	ПР, У
			Примечание: Т – тестирование, Р – написание реферата, У – устный опрос, КР – контрольная работа	

2.4.4. Содержание самостоятельной работы

Подготовка к зачёту и к экзамену

Теоретические вопросы:

1. Понятие тестирования. Принципы, виды и методы тестирования программных продуктов
2. Принцип построения тестового набора данных и составления отладочных заданий.
3. Оформление протокола тестирования.
4. Структурное тестирование.
5. Пошаговое и монолитное тестирование.
6. Оценочное тестирование. Виды и принципы проведения оценочного тестирования.
7. Нисходящее и восходящее тестирование. Критерии формирования тестовых наборов
8. Системное и функциональное тестирование.
9. Определение количества ошибок в ПП и числа необходимых тестов
10. Тестирование программного продукта методом «белого ящика»
11. Тестирование программного продукта методом «чёрного ящика»
12. Понятие отладки программных продуктов.
13. Принципы отладки программных продуктов.
14. Классификация ошибок. Локализация ошибок
15. Методы отладки программного продукта
16. Методы ручного тестирования
17. Метод обратного прослеживания
18. Метод индукции. Метод дедукции.
19. Инструментальные средства отладки ПП
20. Системное программирование, системное ПО.
21. Формализация задачи и разработка алгоритма.
22. Жизненный цикл ПО. Основные этапы разработки ПО.
23. Модели жизненного цикла программного средства.
24. Основные понятия структурного программирования.
25. Модуль. Структура модуля.
26. Списки. Объявление списка, инициализация списка, печать
27. Стеки. Объявление стека, инициализация стека. Добавление элемента в стек.
28. Очереди. Объявление, инициализация очереди. Добавление элемента в очередь.
29. Создание и заполнение внешнего файла, чтение данных из внешнего файла.
30. Текстовые файлы.
31. Структура и способы описания языков программирования высокого уровня.
32. Подпрограмма – процедура. Подпрограмма- функция.
33. Формальные и фактические параметры.
34. Локальные и глобальные переменные.
35. Разработка программного продукта с использованием подпрограммы-процедуры.
36. Модульное программирование.
37. Методы разработки программных модулей.
38. Осуществление разработки кода программного модуля на современных языках программирования
39. Реализация процедур и функций работы с бинарным деревом.
40. Разработка программного продукта с использованием модуля.
41. Объектно-ориентированное проектирование.
42. Документирование результатов анализа и проектирования.
43. Основы языка UML (Unified Modeling Language).
44. Создание абстрактных типов данных. Диаграмма объекта.
45. Принципы объектно-ориентированного анализа: абстрагирование, инкапсуляция, наследование, полиморфизм, модульность, сохраняемость, параллелизм

46. Структура программы на языке Object Pascal (C++). Проект.
47. Компиляция программы и сборка исполняемого модуля.
48. Размещение программы и данных в памяти.
49. Структура исполняемого модуля.
50. Стандартная библиотека функций языка C++ Object Pascal (C++)..
51. Компиляция программы и сборка исполняемого модуля.
52. Размещение программы и данных в памяти.
53. Виртуальные функции и абстрактные базовые классы.
54. Множественное наследование.
55. Ассоциативные массивы.
56. Объекты-функции и предикаты.
57. Цикл разработки прикладного программного обеспечения: концептуализация, анализ, проектирование, кодирование, тестирование, эволюция, сопровождение.
58. Критерии оценки качества программы.
59. Средства и инструменты разработки программного обеспечения.
60. Разработка кода программного продукта на основе готовой спецификации на уровне модуля.
61. Ознакомление с технологией тестирования программных продуктов
62. Выполнение отладки и тестирования программы на уровне модуля
63. Использование инструментальных средств на этапе отладки программного продукта
64. Тестирование программного модуля по определенному сценарию
65. Использование инструментальных средств автоматизации процесса оформления документации.
66. Создание документации к программам. Системы автоматического создания документации. Использование комментариев в программах.
67. Создание собственных модулей. Выкладка их в общий репозиторий на PyPi. Создание инсталляционных пакетов.
68. Тестирование приложений. Тестирование черного и белого ящика.

Практические задания:

1. Дан массив A из n целых чисел. Найти сумму максимального и минимального элемента в массиве.(Поиск максимума и минимума реализовать с помощью подпрограмм-функций).
2. Дан файл целых чисел. Выбрать наибольшее из чисел, принадлежащее интервалу [a,b]. Концы интервала a и b вводятся с клавиатуры.
3. Дан текстовый файл F1. Переписать его содержимое в файл F2, сохраняя строчную структуру и удаляя пустые строки.
4. Даны две символьные строки S1 и S2, содержащие только строчные латинские буквы. Построить строку S3, в которую войдут только общие символы S1 и S2 в алфавитном порядке и без повторений.
5. Дан файл целых чисел. Определить, сколько раз в нем повторяется максимальное значение.
6. Дан файл целых чисел. Определить, сколько раз в нем повторяется максимальное значение.
7. По координатам вершин треугольника вычислить его периметр, используя подпрограмму вычисления длины отрезка, соединяющего две точки. (длина

- отрезка= $\text{sgrt}(\text{sgr}(x_2-x_1)+\text{sgr}(y_2-y_1))$, где (x_1,y_1) - координаты одной точки, (x_2,y_2) -координаты второй точки отрезка).
8. Дан файл целых чисел F1. Создать два новых файла F2 и F3 из положительных и отрицательных чисел соответственно
 9. Даны два файла целых чисел. Определить, в каком из них больше положительных, отрицательных и нулевых значений.
 10. Составить рекурсивную подпрограмму вычисления $N!$
 11. Дана вещественная матрица размера $m*n$. Найти значение наибольшего по модулю элемента матрицы и указать его местоположение в матрице.
 12. Определить среднее арифметическое чисел, хранящихся в файле Note.txt.
 13. Дан список L, из N целых чисел. Удалить первое вхождение максимального элемента в списке.
 14. Дан список L, из N целых чисел. Удалить первое вхождение минимального элемента в списке.
 15. Дан текстовый файл Note.txt. Определить длину самой длинной строки этого файла.
 16. Разработать и произвести отладку программы: Найти сумму бесконечного ряда. Суммировать до тех пор, пока сумма не станет больше заданного $p>0$. Вывести эти числа.
 17. Разработать и произвести отладку программы для определения $N!-M!$. $N! = 1*2*3*4*.....*n$
 18. Разработать и произвести отладку программы: Вычислить сумму квадратов всех целых чисел, пока сумма квадратов меньше заданного числа A. Вывести эти числа.
 19. Разработать и произвести отладку программы: Произведение первых четных чисел равно P, сколько сомножителей взято.
 20. Разработать и произвести отладку программы: Определить все двузначные числа, сумма квадратов цифр которых кратны числу 15.
 21. Разработать и произвести отладку программы: Даны два одномерных массива одинаковой длины. Получить третий массив такой же размерности, каждый элемент которого равен сумме соответствующих элементов данных массивов.
 22. Разработать и произвести отладку программы: дан одномерный массив чисел. Определите сумму элементов, принадлежащих промежутку от A до B (A и B водить с клавиатуры).
 23. Разработать и произвести отладку программы определения количества элементов массива, больших среднего арифметического всех его элементов.
 24. Разработать и произвести отладку программы: Дан массив P целых чисел из n элементов, заполненный случайным образом числами из промежутка $[-10,10]$. Из элементов массива P сформировать массив M той же размерности по правилу: если номер четный, то $M_i=i*P_i$, если нечетный, то $M_i=-P_i$. Исходный и скорректированный массив вывести на экран.
 25. Разработать и произвести отладку программы: ан массив P целых чисел из n элементов, заполненный случайным образом числами из промежутка $[-30,30]$. Из элементов массива P сформировать массив M из четных чисел. Исходный и скорректированный массивы вывести на экран.
 26. Разработать и произвести отладку программы: ан массив P целых чисел из n элементов, заполненный случайным образом числами из промежутка $[-10,10]$. Из элементов массива P сформировать массив M той же размерности по возрастанию. Исходный и скорректированный массивы вывести на экран.
 27. Разработать и произвести отладку программы, печатающей все делители целого числа в порядке убывания.

28. Разработать и произвести отладку программы, печатающей все делители целого числа в порядке возрастания
29. Разработать и произвести отладку программы для решения квадратного уравнения.
30. Создать и отладить приложение – конвертор перевода суммы денег из долларов в рубли.
31. Разработать и произвести отладку программы для вычисления делителей натурального числа N . Вывести сами делители, их количество.
32. Разработать и произвести отладку программы, вычисляющей сумму 1-й и последней цифр натурального числа N . Вывести эти цифры и сумму.
33. Создать и отладить приложение для решения квадратного уравнения.
34. Разработать и произвести отладку программы, находящей все простые числа в заданном диапазоне.
35. Разработать и произвести отладку программы, находящей все нечетные числа в заданном диапазоне и их количество.
36. Разработать и произвести отладку программы, находящей все четные числа в заданном диапазоне и их количество.
37. Разработать и произвести отладку программы, которая задает размер линейного массива, заполняет этот массив случайными целыми числами, выводит список элементов массива; заменить отрицательные числа на 0, положительные – на 1.
38. Разработать и произвести отладку программы, которая задает размер линейного массива, заполняет этот массив случайными целыми числами, выводит список элементов массива; отсортировать массив по убыванию.
39. Разработать и произвести отладку программы, которая задает размер линейного массива, заполняет этот массив случайными целыми числами, выводит список элементов массива; отсортировать массив по возрастанию
40. Разработать и произвести отладку программы, которая задает размер линейного массива, заполняет этот массив случайными целыми числами, выводит список элементов массива; поменять местами два элемента массива с номерами k_1 и k_2 .
41. Разработать и произвести отладку программы, которая задает размер линейного массива, заполняет этот массив случайными целыми числами, выводит список элементов массива, определяет минимальный и максимальный элементы массива.
42. Разработать и произвести отладку программы, которая задает размер линейного массива, заполняет этот массив случайными целыми числами, выводит список элементов массива, определяет сумму всех элементов и количество положительных элементов.

3. ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

3.1. Образовательные технологии при проведении лекций

Для реализации компетентного подхода предусматривается использование в учебном процессе активных и интерактивных форм проведения аудиторных и внеаудиторных занятий с целью формирования и развития профессиональных навыков обучающихся.

Преподавание дисциплины организовано по модульно-блочному принципу.

В процессе преподавания применяются образовательные технологии развития абстрактного, логического и критического мышления. Обязательны компьютерные практикумы по разделам дисциплины, тестирование, тематические презентации, интерактивные технологии.

В учебном процессе наряду с традиционными образовательными технологиями используются компьютерное мультимедийное оборудование и интернет

№	Тема	Виды применяемых образовательных технологий	Кол-во час
1	2	3	4
1	Тема 1.1. Понятие разработки ПО	МБТ, ИКТ	8*
2	Тема 1.2. Понятие тестирования	МБТ, ИКТ	8*
3	Тема 1.3. Тестирование требований	МБТ, ИКТ	14*
4	Тема 1.4. Классификация тестирования	МБТ, ИКТ	26*
5	Тема 1.5. Отчёты о тестировании	МБТ, ИКТ	20*
6	Тема 1.6. Методы отладки и тестирования	МБТ, ИКТ	18*
7	Тема 1.7. Автоматизированное тестирование	МБТ, ИКТ	24*
8	Тема 2.1. Документирование программного обеспечения в соответствии с Единой системой программной документации	МБТ, ИКТ	12*
9	Тема 2.2. Документирование основных процессов ЖЦ ПО	МБТ, ИКТ	16*
10	Тема 2.3. Документирование вспомогательных процессов ЖЦ ПО	МБТ, ИКТ	14*
11	Тема 2.4. Документирование сертификации	МБТ, ИКТ	12*
Итого по курсу			172*
в том числе интерактивное обучение при необходимости*			

3.2. Образовательные технологии при проведении практических занятий (лабораторных работ)

№	Тема занятия	Виды технологий	Часы
1	Проверочная работа по теме 1.1	ИКТ	
2	Проверочная работа по теме 1.2	ИКТ	
3	Проверочная работа по теории	ИКТ	
4	Составление требований к программе	ИКТ	
5	Проверка составленных требований	ИКТ	
6	Анализ выявленных ошибок	ИКТ	
7	Тестирование программы методом «белого ящика»	ИКТ	
8	Перекрёстная проверка результатов тестирования	ИКТ	
9	Тестирование программы методом «чёрного» ящика.	ИКТ	
10	Перекрёстная проверка результатов тестирования	ИКТ	
11	Составление проекта отчёта о дефектах	ИКТ	
12	Перекрёстный анализ составленных отчётов	ИКТ	
13	Составление тест-плана	ИКТ	
14	Оценка трудозатрат на планирование и тестирование по составленным планам	ИКТ	
15	Проверочная работа по теории методов тестирования	ИКТ	
16	Отладка программы всеми известными методами и составление отчёта о результатах	ИКТ	
17	Тестирование одной программы 9-ю методами и составление отчёта о результатах	ИКТ	
18	Проверочная работа по теории автоматизированного тестирования	ИКТ	
19	Разработка тест-кейсов и сценариев тестирования для программы	ИКТ	
20	Ознакомление с инструментами автоматизированного тестирования	ИКТ	
21	Изучение интерфейса инструмента автоматизированного тестирования	ИКТ	
22	Выполнение автоматизированного тестирования программы и составление отчета	ИКТ	
23	Защита отчёта о результатах автоматизированного тестирования	ИКТ	
24	Проверочная работа по теории документирования по ЕСПД	ИКТ	
25	Проверочная работа по теории документирования по международным стандартам	ИКТ	
26	Составление проекта разработки программного средства	ИКТ	
27	Защита проекта разработки программного средства	ИКТ	
28	Составление руководств системного администратора и пользователя	ИКТ	
29	Оформление документации на программные средства с использованием инструментальных средств	ИКТ	
	Примечание: Т – тестирование, Р – написание реферата, У – устный опрос, КР – контрольная работа		

4. УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ «МДК.01.02 «Поддержка и тестирование модулей программного обеспечения»

4.1. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине

Для реализации программы учебной дисциплины должны быть предусмотрены следующие специальные помещения:

Кабинет для преподавания дисциплины «Основы разработки баз данных», оснащенный оборудованием и техническими средствами обучения:

- посадочные места по количеству обучающихся;
- рабочее место преподавателя;
- необходимая для проведения практических занятий методическая и справочная литература (в т.ч. в электронном виде).
- компьютеры;
- выход в интернет;
- мультимедийный проектор, экран;
- мультимедийные презентации.

4.2 Информационное обеспечение реализации программы

Для реализации программы библиотечный фонд образовательной организации должен иметь печатные и/или электронные образовательные и информационные ресурсы, рекомендуемых для использования в образовательном процессе

5. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ УЧЕБНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ (Добавлена нова литература 11.10.2023)

5.1 Основная литература

1. Гниденко, И. Г. Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 248 с. — (Профессиональное образование). — ISBN 978-5-534-18131-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/539215>

2. Федорова, Г. Н. **Разработка модулей программного обеспечения для компьютерных систем** : учебник для студентов учреждений среднего профессионального образования / Г. Н. Федорова. - 4-е изд., перераб. - Москва : Академия, 2020. - 383 с. : ил. - (Профессиональное образование. ТОП-50). - Словарь терминов: с. 372-377. - Библиогр.: с. 378. - ISBN 978-5-4468-8692-0 . - Текст : непосредственный. (25)
3. Федорова, Г. Н. **Разработка модулей программного обеспечения для компьютерных систем** : учебник для студентов учреждений среднего профессионального образования / Г. Н. Федорова. - 5-е изд., стер. - Москва : Академия, 2023. - 383 с. : ил. - (Профессиональное образование. ТОП-50). - Словарь терминов: с. 372-377. - Библиогр.: с. 378. - ISBN 978-5-0054-0485-5 - Текст : непосредственный. 25 экз.

5.2 Дополнительная литература

1. Акопов, А. С. **Компьютерное моделирование** : учебник и практикум для среднего профессионального образования / А. С. Акопов. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 426 с. — (Профессиональное образование). — ISBN 978-5-534-18369-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/534872>
2. Богатырев, В. А. **Информационные системы и технологии. Теория надежности** : учебное пособие для вузов / В. А. Богатырев. — 2-е изд. — Москва : Издательство Юрайт, 2024. — 366 с. — (Высшее образование). — ISBN 978-5-534-15951-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/510320>
3. **Проектирование информационных систем** : учебник и практикум для среднего профессионального образования / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 293 с. — (Профессиональное образование). — ISBN 978-5-534-16217-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/538370>
4. Советов, Б. Я. **Информационные технологии** : учебник для среднего профессионального образования / Б. Я. Советов, В. В. Цехановский. — 7-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 327 с. — (Профессиональное образование). — ISBN 978-5-534-06399-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/536599>
5. Тузовский, А. Ф. **Объектно-ориентированное программирование** : учебное пособие для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2024. — 213 с. — (Высшее образование). — ISBN 978-5-534-16316-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/537332>

5.3 Периодические издания

1. Открытые системы.- URL: <http://biblioclub.ru/index.php?page=journal&jid=436083>
2. Информатика в школе .- URL: <http://dlib.eastview.com/browse/publication/18988>

3. Программные продукты и системы.- URL:
<http://dlib.eastview.com/browse/publication/64086>
4. Информатика и образование.- URL: <http://dlib.eastview.com/browse/publication/18946>
5. Системный администратор.- URL: <http://dlib.eastview.com/browse/publication/66751>
6. Computerword Россия.- URL: <http://dlib.eastview.com/browse/publication/64081>
7. Мир ПК.- URL: <http://dlib.eastview.com/browse/publication/64067>
8. Информационно-управляющие системы.- URL:
<http://dlib.eastview.com/browse/publication/71235>
9. Журнал сетевых решений LAN.- URL:
<http://dlib.eastview.com/browse/publication/64078>
10. Информатика и образование.- URL: <http://dlib.eastview.com/browse/publication/18946>
11. Windows IT Pro/ Re.- URL: <http://biblioclub.ru/index.php?page=journal&jid=138741>
12. Прикладная информатика.- URL: http://elibrary.ru/title_about.asp?id=25599

5.4 Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. ЭБС «Университетская библиотека ONLINE»: сайт. – URL:<http://biblioclub.ru>
2. ЭБС Издательства «Лань»: сайт. – URL:<http://e.lanbook.com>
3. ЭБС «Юрайт»: сайт. –URL:<https://urait.ru/>
4. ЭБС «BOOK.ru»: сайт. – URL: <https://www.book.ru>
5. ЭБС «ZNANIUM.COM»: сайт. – URL: <https://www.znanium.com>
6. Базы данных компании «Ист Вью»: сайт . –URL: <http://dlib.eastview.com>
7. Научная электронная библиотека «eLibrary.ru»: сайт. – URL: <http://elibrary.ru/>
8. Электронная библиотека "Издательского дома "Гребенников". - URL:
<http://www.grebennikon.ru/>
9. Университетская информационная система РОССИЯ (УИС Россия). - URL:
<http://uisrussia.msu.ru/>
10. "Лекториум ТВ" - видеолекции ведущих лекторов России. - URL:
<http://www.lektorium.tv/>
11. База учебных планов, учебно-методических комплексов, публикаций и конференций КубГУ. - URL: <http://docspace.kubsu.ru/>
12. Российское образование [Федеральный портал]. - URL: <https://www.edu.ru/>

**6. КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ УЧЕБНОЙ
ДИСЦИПЛИНЫ МДК.01.02 «Поддержка и тестирование модулей программного
обеспечения»**

6.1 Критерии оценивания практических работ

Раздел модуля 2. Технологии тестирования программных модулей			
<p>ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств</p>	<p>Оценка «отлично» - выполнена отладка модуля (Дополнительно для квалификаций "Программист" и "Специалист по тестированию в области информационных технологий": с использованием инструментария среды проектирования); с пояснением особенностей отладочных классов; сохранены и представлены результаты отладки.</p> <p>Оценка «хорошо» - выполнена отладка модуля (Дополнительно для квалификаций "Программист" и "Специалист по тестированию в области информационных технологий": с использованием инструментария среды проектирования); сохранены и представлены результаты отладки.</p> <p>Оценка «удовлетворительно» - выполнена отладка модуля, пояснены ее результаты.</p>	<p>Экзамен/зачет в форме собеседования: практическое задание по выполнению отладки предложенного программного модуля</p> <p>Защита отчетов по практическим и лабораторным работам</p> <p>Интерпретация результатов наблюдений за деятельностью обучающегося в процессе практики</p>	
	<p>для квалификации "Специалист по тестированию в области информационных технологий": выполнено функциональное тестирование, выполнена и представлена оценка тестового покрытия.</p> <p>Оценка «удовлетворительно» - выполнено тестирование модуля и оформлены результаты тестирования. Дополнительно для квалификации "Специалист по тестированию в области информационных технологий": выполнено функциональное тестирование, выполнена и представлена оценка тестового покрытия с некоторыми погрешностями.</p>	<p>крытия.</p> <p>Защита отчетов по практическим и лабораторным работам</p> <p>Интерпретация результатов наблюдений за деятельностью обу-</p>	

		чающегося в процессе практики
ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода	<p>Оценка «отлично» - определены качественные характеристики программного кода с помощью инструментальных средств; выявлены фрагменты некачественного кода; выполнен рефакторинг на уровнях переменных, функций, классов, алгоритмических структур; проведена оптимизация и подтверждено повышение качества программного кода.</p> <p>Оценка «хорошо» - определены качественные характеристики программного кода с помощью инструментальных средств; выявлены фрагменты некачественного кода; выполнен рефакторинг на нескольких уровнях; проведена оптимизация и выполнена оценка качества полученного программного кода.</p> <p>Оценка «удовлетворительно» - определены качественные характеристики программного кода частично с помощью инструментальных средств; выявлено несколько фрагментов некачественного кода; выполнен рефакторинг на нескольких уровнях; проведена оптимизация и выполнена оценка качества полученного программного кода.</p>	<p>Экзамен/зачет в форме собеседования: практическое задание по оценке качества кода предложенного программного модуля, поиску некачественного программного кода, его анализу, оптимизации методами рефакторинга.</p> <p>Защита отчетов по практическим и лабораторным работам</p> <p>Интерпретация результатов наблюдений за деятельностью обу-</p>

		чающегося в процессе практики
--	--	-------------------------------------

Отметка «5» ставится, если:

- работа выполнена полностью, в логических рассуждениях и обосновании решения нет пробелов и ошибок;
- в решении нет ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но в обосновании шагов решения недостаточны;
- допущена 1-2 ошибки или 1 ошибка и два-три недочета в выкладках.
- Отметка «3» ставится, если:
- допущены 3 ошибки или 2 ошибки и более двух-трех недочетов в выкладках, но обучающийся владеет обязательными умениями по проверяемой теме.
- Отметка «2» ставится, если:
- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными умениями по данной теме в полной мере

6.2 Критерии оценивания конспектов

- Отметка «5» ставится, если:
- работа содержит полные ответы на все теоретические вопросы для составления конспекта;

Отметка «4» ставится, если:

- работа содержит неполный ответ хотя бы на один теоретический вопрос для составления конспекта;

Отметка «3» ставится, если:

- работа содержит неполные ответы на 2 теоретических вопроса для составления конспекта.

Отметка «2» ставится, если:

- работа содержит неполные ответы на 2 и более теоретических вопроса для составления конспекта.

6.3 Критерии оценивания презентаций

Оценка	5	4	3	2
Содержание	Работа полностью завершена	Почти полностью сделаны наиболее важные компоненты работы	Не все важнейшие компоненты работы выполнены	Работа сделана фрагментарно и с помощью преподавателя
	Работа демонстрирует глубокое понимание описываемых процессов	Работа демонстрирует понимание основных моментов, хотя некоторые детали не уточняются	Работа демонстрирует понимание, но неполное	Работа демонстрирует минимальное понимание
	Грамотно используется научная лексика	Научная лексика используется, но иногда не корректно	Научная терминология или используется мало или используется некорректно.	Минимум научных терминов
Грамотность	Нет ошибок: ни грамматических, ни синтаксических	Минимальное количество ошибок	Есть ошибки, мешающие восприятию	Много ошибок, делающих материал трудночитаемым
Дизайн	Имеются постоянные элементы дизайна. Дизайн подчеркивает содержание	Имеются постоянные элементы дизайна. Дизайн соответствует содержанию.	Нет постоянных элементов дизайна. Дизайн может и не соответствовать содержанию.	Элементы дизайна мешают содержанию, накладываясь на него.
	Все параметры шрифта хорошо подобраны (текст хорошо читается)	Параметры шрифта подобраны. Шрифт читаем	Параметры шрифта недостаточно хорошо подобраны, могут мешать восприятию	Параметры не подобраны. Делают текст трудночитаемым

7 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Обучающиеся для полноценного освоения учебного курса должны составлять конспекты как при прослушивании его теоретической (лекционной) части, так и при подготовке к практическим (семинарским) занятиям. Желательно, чтобы конспекты лекций и семинаров записывались в логической последовательности изучения курса и содержались в одной тетради. Это обеспечит более полную подготовку, как к текущим учебным занятиям, так и сессионному контролю знаний.

Самостоятельная работа является важнейшей формой учебно-познавательного процесса. Цель заданий для самостоятельной работы – закрепить и расширить знания, умения, навыки, приобретенные в результате изучения дисциплины; овладеть умением использовать полученные знания в практической работе; получить первичные навыки профессиональной деятельности.

Началом организации любой самостоятельной работы должно быть привитие навыков и умений грамотной работы с учебной и научной литературой. Этот процесс, в первую очередь, связан с нахождением необходимой для успешного овладения учебным материалом литературой. Обучающийся должен изучить список нормативно-правовых актов и экономической литературы, рекомендуемый по учебной дисциплине; уметь пользоваться фондами библиотек и справочно-библиографическими изданиями.

Задания для самостоятельной работы выполняются в письменном виде во внеаудиторное время. Работа должна носить творческий характер, при ее оценке преподаватель в первую очередь оценивает обоснованность и оригинальность выводов. В письменной работе по теме задания обучающийся должен полно и всесторонне рассмотреть все аспекты темы, четко сформулировать и аргументировать свою позицию по исследуемым вопросам. Выбор конкретного задания для самостоятельной работы проводит преподаватель, ведущий практические занятия в соответствии с перечнем, указанным в планах практических занятий.

Обучение осуществляется по модульно-блочной технологии (лекции, практики) с включением инновационных элементов.

С точки зрения используемых методов лекции подразделяются следующим образом: информационно-объяснительная лекция, повествовательная, лекция-беседа, проблемная лекция и т. д.

Устное изложение учебного материала на лекции должно конспектироваться. Слушать лекцию нужно уметь – поддерживать своё внимание, понять и запомнить услышанное, уловить паузы. В процессе изложения преподавателем лекции студент должен выяснить все непонятные вопросы. Записывать содержание лекции нужно обязательно – записи помогают поддерживать внимание, способствуют пониманию и запоминанию услышанного, приводят знание в систему, служат опорой для перехода к более глубокому самостоятельному изучению предмета.

Методические рекомендации по конспектированию лекций:

- запись должна быть системной, представлять собой сокращённый вариант лекции преподавателя. Необходимо слушать, обдумывать и записывать одновременно;
- запись ведётся очень быстро, чётко, по возможности короткими выражениями;
- не прекращая слушать преподавателя, нужно записывать то, что необходимо усвоить. Нельзя записывать сразу же высказанную мысль преподавателя, следует её понять и после этого кратко записать своими словами или словами преподавателя. Важно, чтобы в ней не был потерян основной смысл сказанного;

- имена, даты, названия, выводы, определения записываются точно;
- следует обратить внимание на оформление записи лекции. Для каждого предмета заводится общая тетрадь. Отличным от остального цвета следует выделять отдельные мысли и заголовки, сокращать отдельные слова и предложения, использовать условные знаки, буквы латинского и греческого алфавитов, а также некоторые приёмы стенографического сокращения слов.

Практические занятия по дисциплине «Стандартизация, сертификация и техническое документирование» проводятся в основном по схеме:

- устный опрос по теории в начале занятия (обсуждение теоретических проблемных вопросов по теме);
- работа в группах по разрешению различных ситуаций по теме занятия;
- решение практических задач индивидуально;
- подведение итогов занятия (или рефлексия);
- индивидуальные задания для подготовки к следующим практическим занятиям.

Цель практического занятия - научить студентов применять теоретические знания при решении практических задач на основе реальных данных.

На практических занятиях преобладают следующие методы:

- вербальные (преобладающим методом должно быть объяснение);
- практические (письменные задания, групповые задания и т. п.).

Важным для студента является умение рационально подбирать необходимую учебную литературу. Основными литературными источниками являются:

- библиотечные фонды филиала КубГУ в г. Геленджике;
- электронная библиотечная система «Университетская библиотека онлайн»;
- электронная библиотечная система Издательства «Лань».

Поиск книг в библиотеке необходимо начинать с изучения предметного каталога и создания списка книг, пособий, методических материалов по теме изучения.

Просмотр книги начинается с титульного листа, следующего после обложки. На нём обычно помещаются все основные данные, характеризующие книгу: название, автор, выходные данные, данные о переиздании и т.д. На обороте титульного листа даётся аннотация, в которой указывается тематика вопросов, освещённых в книге, определяется круг читателей, на который она рассчитана. Большое значение имеет предисловие книги, которое знакомит читателя с личностью автора, историей создания книги, раскрывает содержание.

Прочитав предисловие и получив общее представление о книге, следует обратиться к оглавлению. Оглавление книги знакомит обучаемого с содержанием и логической структурой книги, позволяет выбрать нужный материал для изучения. Год издания книги позволяет судить о новизне материала. В книге могут быть примечания, которые содержат различные дополнительные сведения. Они печатаются вне основного текста и разъясняют отдельные вопросы. Предметные и алфавитные указатели значительно облегчают повторение изложенного в книге материала. В конце книги может располагаться вспомогательный материал. К нему обычно относятся инструкции, приложения, схемы, ситуационные задачи, вопросы для самоконтроля и т.д.

Для лучшего представления и запоминания материала целесообразно вести записи и конспекты различного содержания, а именно:

- пометки, замечания, выделение главного;
- план, тезисы, выписки, цитаты;
- конспект, рабочая запись, реферат, доклад, лекция и т.д.

Читать учебник необходимо вдумчиво, внимательно, не пропуская текста, стараясь понять каждую фразу, одновременно разбирая примеры, схемы, таблицы, рисунки, приведённые в учебнике.

Одним из важнейших средств, способствующих закреплению знаний, является краткая запись прочитанного материала – составление конспекта. Конспект – это краткое связное изложение содержания темы, учебника или его части, без подробностей

и второстепенных деталей. По своей структуре и последовательности конспект должен соответствовать плану учебника. Поэтому важно сначала составить план, а потом писать конспект в виде ответа на вопросы плана. Если учебник разделён на небольшие озаглавленные части, то заголовки можно рассматривать как пункты плана, а из текста каждой части следует записать те мысли, которые раскрывают смысл заголовка.

Требования к конспекту:

- краткость, сжатость, целесообразность каждого записываемого слова;
- содержательность записи - записываемые мысли следует формулировать кратко, но без ущерба для смысла. Объём конспекта, как правило, меньше изучаемого текста в 7-15 раз;
- конспект может быть, как простым, так и сложным по структуре – это зависит от содержания книги и цели её изучения.

Методические рекомендации по конспектированию:

- прежде чем начать составлять конспект, нужно ознакомиться с книгой, прочитать её сначала до конца, понять прочитанное;
- на обложке тетради записываются название конспектируемой книги и имя автора, составляется план конспектируемого текста;
- записи лучше делать при прочтении не одного-двух абзацев, а целого параграфа или главы;
- конспектирование ведётся не с целью иметь определённые записи, а для более полного овладения содержанием изучаемого текста, поэтому в записях отмечается и выделяется всё то новое, интересное и нужное, что особенно привлекло внимание;
- после того, как сделана запись содержания параграфа, главы, следует перечитать её, затем снова обращаться к тексту и проверить себя, правильно ли изложено содержание.

Техника конспектирования:

- конспектируя книгу большого объёма, запись следует вести в общей тетради;
- на каждой странице слева оставляют поля шириной 25-30 мм для записи коротких подзаголовков, кратких замечаний, вопросов;
- каждая страница тетради нумеруется;
- для повышения читаемости записи оставляют интервалы между строками, абзацами, новую мысль начинают с «красной» строки;
- при конспектировании широко используют различные сокращения и условные знаки, но не в ущерб смыслу записанного. Рекомендуется применять

общеупотребительные сокращения, например: м.б. – может быть; гос. – государственный; д.б. – должно быть и т.д.

- не следует сокращать имена и названия, кроме очень часто повторяющихся;
- в конспекте не должно быть механического переписывания текста без продумывания его содержания и смыслового анализа.

Для написания реферата необходимо выбрать тему, согласовать ее с преподавателем, подобрать несколько источников по теме, выполнить анализ источников по решению проблемы, обосновать свою точку зрения на решение проблемы.

8 ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ КОНТРОЛЯ УСПЕВАЕМОСТИ

8.1 Паспорт фонда оценочных средств

№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или ее части)	Наименование оценочного средства
1.	Раздел 1. Отладка и тестирование программного обеспечения	ОК 1, 2,4,5,9,10 ПК 1.3, 2.1, 2.3, 2.4 ,4.2, 5.1	
2.	Тема 1.1. Понятие разработки ПО	ОК 1, 2,4,5,9,10	Тестирование, анализ выполнения практических работ
3.	Тема 1.2. Понятие тестирования	ПК 1.3, 2.1, 2.3, 2.4 ,4.2, 5.1	Тестирование, анализ выполнения практических работ
4.	Тема 1.3. Тестирование требований	ОК 1, 2,4,5,9,10	Тестирование, анализ выполнения практических работ
5.	Тема 1.4. Классификация тестирования	ПК 1.3, 2.1, 2.3, 2.4 ,4.2, 5.1	Тестирование, анализ выполнения практических работ, зачёт
6.	Тема 1.5. Отчёты о тестировании	ОК 1, 2,4,5,9,10	Тестирование, анализ выполнения практических работ, зачёт
7.	Тема 1.6. Методы отладки и тестирования	ПК 1.3, 2.1, 2.3, 2.4 ,4.2, 5.1	Тестирование, анализ выполнения практических работ, зачёт
8.	Тема 1.7. Автоматизированное тестирование	ОК 1, 2,4,5,9,10	Тестирование, анализ выполнения практических работ, зачёт
9.	Раздел 2. Документирование	ПК 1.3, 2.1, 2.3, 2.4 ,4.2, 5.1	
10.	Тема 2.1. Документирование программного обеспечения в соответствии с Единой системой программной документации	ОК 1, 2,4,5,9,10	Тестирование, анализ выполнения практических работ, зачёт
11.	Тема 2.2. Документирование основных процессов ЖЦ ПО	ПК 1.3, 2.1, 2.3, 2.4 ,4.2, 5.1	Тестирование, анализ выполнения практических работ, зачёт

№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или ее части)	Наименование оценочного средства
12.	Тема 2.3. Документирование вспомогательных процессов ЖЦ ПО	ОК 1, 2,4,5,9,10	Тестирование, анализ выполнения практических работ, зачёт
13.	Тема 2.4. Документирование сертификации	ОК 1, 2,4,5,9,10	Тестирование, анализ выполнения практических работ, зачёт
14.	Всего по дисциплине	ПК 1.3, 2.1, 2.3, 2.4 .2, 5.1	

8.2. Критерии оценки знаний

Контроль и оценка результатов освоения модуля осуществляется преподавателем в процессе проведения практических занятий, тестирования, а также выполнения студентами индивидуальных самостоятельных заданий.

Тест. Система стандартизированных заданий, позволяющая автоматизировать процедуру измерения уровня знаний и умений обучающегося. Тест оценивается по количеству правильных ответов (не менее 50%).

Критерии оценки знаний студентов в целом по дисциплине:

«отлично» - выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы модуля и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений;

«хорошо» - выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности;

«удовлетворительно» - выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, недостаточно правильные формулировки базовых понятий, нарушения логической последовательности в изложении программного материала, но при этом он владеет основными разделами учебной программы, необходимыми для дальнейшего обучения и может применять полученные знания по образцу в стандартной ситуации;

«неудовлетворительно» - выставляется студенту, который не знает большей части основного содержания учебной программы модуля, допускает грубые ошибки в

формулировках основных понятий дисциплины и не умеет использовать полученные знания при решении типовых практических задач.

8.3. Оценочные средств для проведения текущей аттестации

Дайте ответы на вопросы

1.Тестирование программного обеспечения (Software Testing) - проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. [IEEE Guide to Software Engineering Body of Knowledge, SWEBOOK, 2004] В более широком смысле, тестирование - это одна из техник контроля качества, включающая в себя активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis).

2.Верификация (Verification) - это процесс оценки системы или её компонентов с целью определения удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале этого этапа [IEEE]. Т.е. выполняются ли наши цели, сроки, задачи по разработке проекта, определенные в начале текущей фазы.

3.Валидация (Validation) - это определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе [BS7925-1].

4.Тест дизайн (Test Design) - это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования.

5.Тестовый случай (Test Case) - это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

6.Баг/Дефект Репорт (Bug Report) - это документ, описывающий ситуацию или последовательность действий приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

7.Тестовое Покрытие (Test Coverage) - это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.

8.Время Прохождения Тест Кейса (Test Case Pass Time) - это время от начала прохождения шагов тест кейса до получения результата теста.

9.Цели тестирования (Обнаружение дефектов, Повышение уверенности в уровне качества, Предоставление информации для принятия решений, Предотвращение дефектов)

10.Принципы тестирования: (Тестирование показывает наличие дефектов, Исчерпывающее тестирование невозможно, Раннее тестирование, Скопление дефектов, Повторные прогоны тестов находят меньше ошибок, заблуждение об отсутствии дефектов, Тестирование зависит от контекста, Заблуждение об отсутствии ошибок)

11. Цели тестирования (Увеличение приемлемого уровня пользовательского доверия в том, что программа функционирует корректно во всех необходимых обстоятельствах: корректное поведение; определяется из требований к продукту; уровень доверия; наглядность, уровень остаточного обнаружения дефектов, требования к надёжности, что бы это всё ни значило, необходимые обстоятельства - требование реального окружения; реалистичная среда тестирования (схожие наборы данных и т.п.).

12. Уровни восприятия тестирования в компании.

Уровень 0 - (Не отличает некорректное поведение и ошибки в программе. Не учитывает требования надежности и безопасности)

Уровень 1 - предназначение – показать корректность ПО

Уровень 2 - Демонстрация ошибок

Уровень 3 - Тестирование может показать наличие ошибок

13. Риски использования ПО (Последствия незначительные, последствия катастрофические)

14. Участники тестирования, их роль, квалификация и обязанности

1. Проектирование тестов – На основании формальных критериев – На основании знаний предметной области, опыта и экспертизы

2. Автоматизация тестов – Знание средств, скриптов

3. Исполнение тестов – Нет специальных требований к квалификации

4. Анализ результатов – Знания предметной области)

15. Мониторинг прогресса и контроль тестирования (ISTQB) (Целью мониторинга тестирования является предоставление результата и обзора процесса тестирования. Информация отслеживается вручную или автоматически и может быть использована для измерения критериев выхода, таких как покрытие. Метрики также могут быть использованы для оценки прогресса тестирования по сравнению с запланированным расписанием и бюджетом. Контроль тестирования описывает любые направляющие или корректирующие действия, принятые как результат по полученной и собранной информации и значениям метрик. Контроль тестирования может затрагивать любые действия по тестированию, а также воздействовать на другие действия и задачи жизненного цикла ПО)

16. Тестовый случай, тестовый сценарий и тестовое покрытие.

(Тестовый случай (Test Case - это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части. Под тест кейсом понимается структура вида: Action > Expected Result > Test Result

Тестовый сценарий - последовательность тестовых случаев; состоит из набора входных значений, предусловий выполнения, ожидаемых результатов и постусловий, определяемых для покрытия определенных тестовых условий (или тестового условия) или целей (цели) тестирования.

Тестовое Покрытие - это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.)

Тема 1.3. Тестирование требований

ЛАБОРАТОРНАЯ РАБОТА №3 Тестирование требований

Цель работы: изучить критерии качества требований, выполнить тестирование спецификации требований.

Теоретические сведения

Требования обеспечивают основу для последующего тестирования - процесса анализа программного средства на предмет соответствия зафиксированным требованиям и/или ожиданиям и нуждам пользователя или заказчика. От качества сформированных требований зависит качество программного обеспечения, т.к.

требования к программному продукту являются базой для генерации тестов и обнаружения дефектов, представляющих собой любое отклонение от спецификации.

В связи с вышеизложенным при разработке программного обеспечения тестирование необходимо выполнять уже на стадии разработки спецификации (рисунок 1).



Рисунок 1 - Жизненный цикл проекта

Тестирование требований является важным этапом разработки продукта, так как это приводит к:

- 1) снижению риска получить продукт, не отвечающий ожиданиям заказчика или нуждам конечных пользователей;
- 2) снижению затрат на разработку и тестирование продукта;
- 3) сокращению сроков сдачи готового продукта;
- 4) налаживанию взаимопонимания при создании продукта между всеми вовлеченными исполнителями.

Выделяют 9 критериев качества требований:

- 1) корректность;
- 2) недвусмысленность;
- 3) полнота;
- 4) непротиворечивость;
- 5) упорядоченность по важности и стабильности;
- 6) проверяемость;
- 7) модифицируемость;
- 8) трассируемость;

Корректные требования. Набор требований к программному обеспечению является корректным тогда и только тогда, когда каждое требование, сформулированное в нем, представляет нечто, требуемое от создаваемой системы. Данная формулировка отражена на рисунке 2.

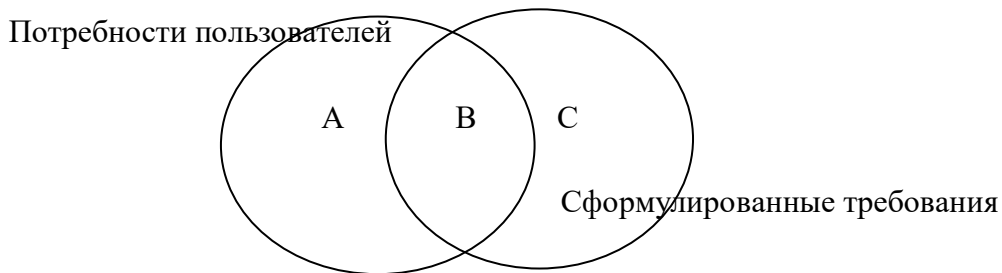


Рисунок 2 - Множества потребностей и требований

Если левый круг (область А) представляет множество потребностей пользователя, а правый (область С) — требования, то корректные требования будут находиться в области пересечения кругов, область В.

Недвусмысленные требования. Требование является недвусмысленным тогда и только тогда, когда его можно однозначно интерпретировать. Хотя главным свойством любого требования по праву считается корректность, неоднозначность зачастую представляет собой более сложную проблему. Если формулировка требований может по-разному интерпретироваться разработчиками, пользователями и другими участниками проекта, вполне может оказаться, что построенная система будет полностью отличаться от того, что представлял себе пользователь.

Полнота набора требований. Набор требований является полным тогда и только тогда, когда он описывает все важные требования, интересующие пользователя, в том числе требования, связанные с функциональными возможностями, производительностью, ограничениями проектирования, атрибутами или внешними интерфейсами. Полный набор требований должен также задавать требуемый ответ программы на всевозможные классы ввода — как правильные, так и неправильные — во всевозможных ситуациях. Помимо этого, он должен содержать полные ссылки и подписи всех рисунков, таблиц и диаграмм набора требований, а также определения всех терминов и единиц измерения.

Непротиворечивость набора требований. Множество требований является внутренне непротиворечивым, когда ни одно его подмножество, состоящее из отдельных требований, не противоречит другим подмножествам. Конфликты могут иметь различную форму и проявляться на различных уровнях детализации; если набор требований был написан достаточно формально и поддерживается соответствующими автоматическими средствами, конфликт иногда удается обнаружить посредством механического анализа. Но, скорее всего, разработчикам вместе с другими участниками проекта придется провести проверку множества требований вручную, чтобы удалить все потенциальные конфликты.

Упорядочение требований по их важности и стабильности. В высококачественном наборе требований разработчики, клиенты и другие заинтересованные лица упорядочивают отдельные требования по их важности для

клиента и стабильности. Этот процесс упорядочения особенно важен для управления масштабом. Если ресурсы недостаточны, чтобы в пределах выделенного времени и бюджета реализовать все требования, очень полезно знать, какие требования являются не столь уж обязательными, а какие пользователь считает критическими.

Проверяемые требования. Требование в целом является проверяемым, когда каждое из составляющих его элементарных требований является проверяемым, т.е. когда можно протестировать каждое из них и выяснить, действительно ли они выполняются.

Модифицируемый набор требований. Множество требований является модифицируемым, когда его структура и стиль таковы, что любое изменение требований можно произвести просто, полно и согласованно, не нарушая существующей структуры и стиля всего множества. Для этого требуется, чтобы пакет требований имел минимальную избыточность и был хорошо организован, с соответствующим содержанием, индексом и возможностью перекрестных ссылок.

Трассируемые требования. Требование в целом является трассируемым, когда ясно происхождение каждого из составляющих его элементарных требований и существует механизм, который делает возможным обращение к этому требованию при дальнейших действиях по разработке. На практике это обычно означает, что каждое требование имеет уникальный номер или идентификатор. Возможность трассировки имеет огромное значение. Разработчики могут использовать ее как для достижения лучшего понимания проекта, так и для обеспечения более высокой степени уверенности, что все требования выполняются данной реализацией.

Существуют различные методы тестирования требований:

1. Метод просмотра (универсальный метод, выполняется бизнес-аналитиком или тестировщиком):
 - Ознакомление с требованиями.
 - Проверка требований по критериям качества.
 - Оформление дефектов.
 - Оформление отчета.
2. Метод экспертизы (выполняется при участии команды из бизнес-аналитиков, представителей заказчика, разработчиков, лояльных пользователей, тестировщиков):
 - Планирование.
 - Обзорная встреча.
 - Подготовка.
 - Совещание.
 - Переработка.
 - Завершающий этап.
3. Метод составления вариантов тестирования (выполняется тестировщиком). Варианты тестирования занимают промежуточную позицию между User Case и Test Case, помимо использования для тестирования требований в дальнейшем легко расширяются до Test Cases и составляют основу тестовой документации.

Порядок выполнения работы

1. Получить задание у преподавателя.
2. Протестировать спецификацию методом просмотра.
3. Оформить отчет и защитить лабораторную работу.

Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Отчет по тестированию спецификации.
4. Выводы по работе.

Контрольные вопросы

1. Как выглядит жизненный цикл проекта?
2. Какие выделяют критерии качества?
3. Какие требования считаются проверяемыми?
4. Какие требования считаются модифицируемыми?
5. Какие требования считаются корректными?
6. Какие требования считаются недвусмысленными?
7. Какие требования считаются полными?
8. Какие требования считаются непротиворечивыми?
9. Какие требования считаются упорядоченными по важности и стабильности?
10. Какие требования считаются трассируемыми?
11. Какие существуют методы тестирования требований?
12. Какие этапы включает в себя метод просмотра при тестировании требований?

Тема 1.4.

Классификация тестирования

Цель работы: изучить классификацию видов тестирования, практически закрепить эти знания путем генерации тестов различных видов, научиться планировать тестовые активности в зависимости от специфики поставляемой на тестирование функциональности.

Теоретические сведения

Тестирование - процесс, направленный на оценку корректности, полноты и качества разработанного программного обеспечения.

Тестирование можно классифицировать по очень большому количеству признаков. Далее приведен обобщенный список видов тестирования по различным основаниям.

Типы тестов по покрытию (по глубине)

Smoke test — тестирование системы для определения корректной работы базовых функций программы в целом, без углубления в детали. При проведении теста определяется пригодность сборки для дальнейшего тестирования.

Minimal Acceptance Test (MAT, Positive test): тестирование системы или ее части только на валидных данных (валидные данные - это данные, которые необходимо использовать для корректной работы модуля/функции). При тестировании проверяется правильной работы всех функций и модулей с валидными данными.

Для крупных и сложных приложений используется ограниченный набор сценариев и функций.

Acceptance Test (AT): полное тестирование системы или ее части как на корректных, так и на некорректных данных/сценариях. Вид теста, направленный на подтверждение того, что приложение может использоваться по назначению при любых

условиях.

Тест на этом уровне покрывает все возможные сценарии тестирования: проверку работоспособности модулей при вводе корректных значений; проверку при вводе некорректных значений; использование форматов данных отличных от тех, которые указаны в требованиях; проверку исключительных ситуаций, сообщений об ошибках; тестирование на различных комбинациях входных параметров; проверку всех классов эквивалентности; тестирование граничных значений интервалов; сценарии не предусмотренные спецификацией и т.д.

Тестовые активности (типы тестов по покрытию (по ширине)):

Defect Validation - проверка результата исправления дефектов. Включает в себя проверку на воспроизводимость дефектов, которые были исправлены в новой сборке продукта, а также проверку того, что исправление не повлияло на ранее работавшую функциональность

New Feature Test (NFT, AT of NF)- определение качества поставленной на тестирование новой функциональности, которая ранее не тестировалась. Данный тип тестирования включает в себя: проведение полного теста (AT) непосредственно новой функциональности; тестирование новой функциональности на соответствие документации; проверку всевозможных взаимодействий ранее реализованной функциональности с новыми модулями и функциями.

Regression testing (регрессионное тестирование) - проводится с целью оценки качества ранее реализованной функциональности. Включает в себя проверку стабильности ранее реализованной функциональности после внесения изменений, например добавления новой функциональности, исправление дефектов, оптимизация кода, разворачивание приложения на новом окружении. Регрессионное тестирование может быть проведено на уровне Smoke, MAT или AT.

Типы тестов по знанию коду

Черный ящик - тестирование системы, функциональное или нефункциональное, без знания внутренней структуры и компонентов системы. У тестировщика нет доступа к внутренней структуре и коду приложения либо в процессе тестирования он не обращается к ним.

Белый ящик - тестирование основанное на анализе внутренней структуры компонентов или системы. У тестировщика есть доступ к внутренней структуре и коду приложения.

Серый ящик - комбинация методов белого и черного ящика, состоящая в том, что к части кода архитектуры у тестировщика есть, а к части кода - нет.

Типы тестов по степени автоматизации

Ручное - тестирование, в котором тест-кейсы выполняются тестировщиком вручную без использования средств автоматизации.

Автоматизированное - набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования. Тест-кейсы частично или полностью выполняет специальное инструментальное средство.

Типы тестов по изолированности компонентов

Unit/component (модульное) - тестирование отдельных компонентов (модулей)

программного обеспечения.

Integration (интеграционное) - тестируется взаимодействие между интегрированными компонентами или системами.

System (системное) - тестируется работоспособность системы в целом с целью проверки того, что она соответствует установленным требованиям.

Типы тестов по подготовленности.

Интуитивное тестирование выполняется без подготовки к тестам, без определения ожидаемых результатов, проектирования тестовых сценариев.

Исследовательское тестирование - метод проектирования тестовых сценариев во время выполнения этих сценариев. Тестировщик совершает проверки, продумывает их, придумывает новые проверки, часто использует для этого полученную информацию.

Тестирование по документации - тестирование по подготовленным тестовым сценариям, руководству по осуществлению тестов.

Типы тестов по месту и времени проведения

User Acceptance Testing (UAT) (приемочное тестирование) - формальное тестирование по отношению к потребностям, требованиям и бизнес процессам пользователя, проводимое с целью определения соответствия системы критериям приёмки и дать возможность пользователям, заказчикам или иным авторизованным лицам определить, принимать систему.

Alpha Testing (альфа-тестирование) - моделируемое или действительное функциональное тестирование, выполняется в организации, разрабатывающей продукт, но не проектной командой (это может быть независимая команда тестировщиков, потенциальные пользователи, заказчики). Альфа тестирование часто применяется к коробочному программному обеспечению в качестве внутреннего приемочного тестирования.

Beta Testing (бета-тестирование) - эксплуатационное тестирование потенциальными или существующими клиентами/заказчиками на внешней стороне (в среде, где продукт будет использоваться) никак связанными с разработчиками, с целью определения действительно ли компонент или система удовлетворяет требованиям клиента/заказчика и вписывается в бизнес-процессы. Бета-тестирование часто проводится как форма внешнего приемочного тестирования готового программного обеспечения для того, чтобы получить отзывы рынка.

Типы тестов по объекту тестирования

Functional testing (функциональное тестирование) - это тестирование, основанное на анализе спецификации, функциональности компонента или системы. Функциональным можно назвать любой вид тестирования, который согласно требованиям проверяет правильную работу.

Safety testing (тестирование безопасности) - тестирование программного продукта с целью определить его безопасность (безопасность - способность программного продукта при использовании оговоренным образом оставаться в рамках приемлемого риска причинения вреда здоровью, бизнесу, программам, собственности или окружающей среде).

Security testing (тестирование защищенности) - это тестирование с целью оценить защищенность программного продукта. Тестирование защищенности проверяет

фактическую реакцию защитных механизмов, встроенных в систему, на проникновение.

Compatibility testing (тестирование совместимости) - процесс тестирования для определения возможности взаимодействия программного продукта, проверка работоспособности приложения в различных средах (браузеры и их версии, операционные системы, их типа, версии и разрядность)

Виды тестов:

J кросс-браузерное тестирование (различные браузеры или версии браузеров)

J кросс-платформенное тестирование (различные операционные системы или версии операционных систем)

Нефункциональное тестирование - это проверка характеристик программы. Иначе говоря, когда проверяется не именно правильность работы, а какие-либо свойства (внешний вид и удобство пользования, скорость работы и т.п.).

1. Тестирование пользовательского интерфейса (GUI)

тестирование, выполняемое путем взаимодействия с системой через графический интерфейс пользователя.

J навигация

J цвета, графика, оформление

J содержание выводимой информации

J поведение курсора и горячие клавиши

J отображение различного количества данных (нет данных, минимальное и максимальное количество)

J изменение размеров окна или разрешения экрана

2. Тестирование удобства использования (Usability Testing) - тестирование с целью определения степени понятности, легкости в изучении и использовании, привлекательности программного продукта для пользователя при условии использования в заданных условиях эксплуатации.

J визуальное оформление

J навигация

J логичность

3. Тестирование доступности (Accessibility testing) - тестирование, которое определяет степень легкости, с которой пользователи с ограниченными способностями могут использовать систему или ее компоненты.

4. Тестирование интернационализации - тестирование способности продукта работать в локализованных средах (способность изменять элементы интерфейса в зависимости от длины и направления текста, менять сортировки/форматы под различные локали и т.д.). (Максим Черняк).

Интернационализация - это процесс, упрощающий дальнейшую адаптацию продукта к языковым и культурным особенностям региона, отличного от того, в котором разрабатывался продукт. Это адаптация продукта для потенциального использования практически в любом месте, Интернационализация производится на начальных этапах разработки, в то время как локализация — для каждого целевого языка.

5. Тестирование локализации (Localization testing) - тестирование, проводимое с целью проверить качество перевода продукта с одного языка на другой.

6. Тестирование производительности или нагрузочное

тестирование - процесс тестирования с целью определения производительности программного продукта.

Виды тестов:

И нагрузочное тестирование (Performance and Load testing) - вид тестирования производительности, проводимый с целью оценки поведения компонента или системы при возрастающей нагрузке, например количестве параллельных пользователей и/или операций, а также определения какую нагрузку может выдержать компонент или система;

И объемное тестирование (Volume testing) - позволяет получить оценку производительности при увеличении объемов данных в базе данных приложения;

И тестирование стабильности и надежности (Stability / Reliability testing) - позволяет проверять работоспособность приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки.

И стрессовое тестирование (Stress testing) - вид тестирования производительности, оценивающий систему или компонент на граничных значениях рабочих нагрузок или за их пределами, или же в состоянии ограниченных ресурсов, таких как память или доступ к серверу.

7. Тестирование требований (Requirements testing) - проверка требований на соответствие основным характеристикам качества.

8. Тестирование прототипа (Prototyte testing) - метод выявления структурных, логических ошибок и ошибок проектирования на ранней стадии развития продукта до начала фактической разработки.

9. Тестирование установки (Installability testing) и лицензирования - процесс тестирования устанавливаемости программного продукта.

Виды тестов:

И формальный тест программы установки приложения (проверка пользовательского интерфейса, навигации, удобства пользования, соответствия общепринятым стандартам оформления);

И функциональный тест программы установки;

И тестирование механизма лицензирования и функций защиты от пиратства;

И проверка стабильности приложения после установки.

10. Тестирование на отказ и восстановление (Failover and Recovery Testing) - тестирование при помощи эмуляции отказов системы или реально вызываемых отказов в управляемом окружении.

Тестирование программного продукта включает следующие этапы:

1. Изучение и анализ предмета тестирования.
2. Планирование тестирования.
3. Исполнение тестирования.

Изучение и анализ предмета тестирования начинается еще до утверждения спецификации и продолжается на стадии разработки (кодирования) программного обеспечения. Конечной целью этапа изучение и анализ предмета тестирования является получение ответов на два вопроса:

- какие функциональности предстоит протестировать,
- как эти функциональности работают.

Планирование тестирования происходит на стадии разработки (кодирования) программного обеспечения. На стадии планирования тестирования перед тестировщиком стоит задача поиска компромисса между объемом тестирования, который возможен в теории, и объемом тестирования, который возможен на практике. На данной стадии необходимо ответить на вопрос: как будем тестировать? Результатом планирования тестирования является тестовая документация.

Выполнение тестирования происходит на стадии тестирования и представляет собой практический поиск дефектов с использованием тестовой документации, составленной ранее.

Для всех программных продуктов выполняют следующие типы тестов и их композиции.

Для **первого билда** рекомендуется проводить Smoke+AT готовой функциональности: поверхностное тестирование (Smoke Test) выполняется для определения пригодности сборки для дальнейшего тестирования; полное тестирование системы или ее части как на корректных, так и на некорректных данных/сценариях (Acceptance Test, AT) позволяет обнаружить дефекты и внести запись о них в багтрекинг-систему.

Для последующих билдов композиции тестов могут быть следующими:

- Если не была добавлена новая функциональность, то: DV+MAT. Т.е., выполняется проверка исправления дефектов программистом (Defect Validation, DV), а также проверка работоспособности остальной функциональности после исправления дефектов на позитивных сценариях (Minimal Acceptance Test, MAT).
- Если была добавлена новая функциональность, то: Smoke+DV+NFT+Regression Test. В частности, выполняется поверхностное тестирование (Smoke Test), проверка исправления дефектов программистом (Defect Validation, DV), тестирование новых функциональностей (New Feature Testing, NFT), проверка старых функциональностей, т.е. регрессионное тестирование (Regression Test).
- Если была добавлена новая функциональность, то возможен также вариант: DV+NFT+Resression test, т.е. без выполнения Smoke Test.

В зависимости от типа и специфики приложения (web, desktop, mobile) выполняют специализированные тесты (например, кроссбраузерное или кроссплатформенное тестирование, тестирование локализации интернационализации и др.).

Порядок выполнения работы

1. Получить задание у преподавателя.
2. Выполнить генерацию тестов различных видов для конкретного объекта реального мира (пример приведен на рисунке 1).
3. Спланировать тестовые активности для следующих задач:
 - 3.1 Поставлен на тестирование модуль 1, модуль 2, модуль 3.
 - 3.2 Проведены исправления (fix) для заведенных дефектов, доставлена новая функциональность - модуль 4.
 - 3.3 Заказчик решил расширять рынки сбыта и просит осуществить поддержку для Великобритании (кроме уже существующей Беларуси).
- 3.4 Заказчик хочет убедиться, что ПО держит нагрузку в 2000 пользователей

Тема 1.5. Отчёты о тестировании

Документирование результатов тестирования

Цель работы: составить итоговый отчет о результатах тестирования web-приложения.

Теоретические сведения

Итоговый отчет можно разделить на части с соответствующей информацией:

- Приветствие.
- Общая информация (Common Information).
- Тестовое окружение (Test Platform).
- Рекомендации QA (QA Recommendations).
- Детализированная информация (Detailed Information).
- Окончание содержимого.

Приветствие

Свое письмо с отчетом необходимо начать с приветствия всех адресатов.

Если по каким-либо причинам произошла задержка данных отчета, либо не весь запланированный функционал был проверен, то эту информацию необходимо предоставить в начале письма. Следует извиниться за задержку и указать адекватные причины произошедшего. Также в самом начале письма следует указывать, если были какие-то внешние факторы, препятствующие проверке какой-то части функционала.

Если во время тестирования не произошло никаких форс-мажорных обстоятельств, то достаточно обычного вежливого приветствия и далее уже переход к следующим пунктам.

Общая информация (Common Information)

В данной части отчета описывается, какие виды тестов проводились. Зачастую указываются модули, которые тестировались или функционал. Стоит удостовериться, не забыта ли какая-то часть функционала, особенно это актуально, когда нужно собрать итоговый отчет, соединив в себе данные о разных видах тестов и функционале.

Тестовое окружение (Test Platform)

Как правило, в этой части указываются:

- Название проекта.
- Номер сборки.
- Ссылка на проект (сборку). Необходимо убедиться, что зайдя по этой ссылке вы действительно попадаете на проект или можете установить приложение.

При указании данных в этой части отчета нужно быть очень внимательным, т.к. неправильная ссылка на сборку или неверный номер сборки не дают достоверной информации всем заинтересованным людям, а также затрудняют работу человеку, собирающему финальный отчет.

Рекомендации QA (QA Recommendations)

Данная часть отчета является наиболее важной, т.к. здесь отражается общее состояние сборки. Здесь показывается аналитическая работа тестировщика, его рекомендации по улучшению функционала, наиболее слабые места и наиболее критичные дефекты, динамика изменения качества проекта.

В этом разделе должна быть информация о следующем:

- Указан функционал (часть функционала), который заблокирован для проверки. Даны пояснения почему этот функционал не проверен (указаны наиболее

критичные дефекты).

- Произведен анализ качества проверенного функционала. Следует указать, улучшилось оно или ухудшилось по сравнению с предыдущей версией, какое качество на сегодняшний момент, какие факторы повлияли на выставление именно такого качества сборки.
- Если качество сборки ухудшилось, то обязательно должны быть указаны регрессионные места.
- Наиболее нестабильные части функционала следует выделить и указать причину, по которой они таковыми являются.
- Даны рекомендации по тому функционалу и дефектам, скорейшее исправление которых является наиболее приоритетным.
- Список наиболее критичных для сборки дефектов, с указанием названия и их критичности.
- Для отчета уровня Smoke обязательно указать весь нестабильный функционал. Если сборка является релизной или предрелизной, то любое ухудшение качества является критичным и важно об этом сообщить менеджеру как можно раньше. Помимо всего вышеуказанного для релизных и предрелизных сборок в отчете о качестве продукта важно указывать следующее:

- Дана информация о всех проблемах, характерных сборке. Проведен анализ, насколько оставшиеся проблемы являются критичными для конечного пользователя.
- Указаны дефекты, которые следует исправить, чтобы качество конечной сборки было выше.

Детализированная информация (Detailed Information)

В данной части отчета описывается более подробная информация о проверенных частях функционала, устанавливается качество каждой проверенной части функционала(модуля) в отдельности. В зависимости от типа проводимых тестов, эта часть отчета будет отличаться.

Smoke

При оценке качества функционала на уровне Smoke теста, оно может быть либо Приемлемым, либо Неприемлемым. Качество сборки зависит от нескольких факторов:

- Если это релизная или предрелизная сборка, то для выставления Приемлемого качества на уровне Smoke не должно быть найдено функциональных дефектов.
- Наличие нового функционала. Новый функционал, который впервые поставляется на тестирование, не должен содержать дефектов уровня Smoke для выставления Приемлемого качества всей сборки.
- Чтобы установить сборке Приемлемое качество, не должно быть дефектов уровня Smoke у того функционала, по которому планируется проводить полные тесты.
- Все наиболее важные части функционала отработывают корректно, тогда качество всего функционала на уровне Smoke может быть оценено, как Приемлемое.

В части о детализированной информации качества сборки следует более подробно описать проблемы, которые были найдены во время теста.

DV

В этой части отчета указывается качество о проведении валидации дефектов.

Здесь должна быть следующая информация:

- Общее количество всех дефектов, поступивших на проверку.
- Количество неисправленных дефектов и их процент от общего количества.
- Список дефектов, которые не были проверены и причины, по которым этого не было сделано.
- Наглядная таблица с неисправленными дефектами.

По вышеуказанным результатам выставляется качество теста. Если процент неисправленных дефектов < 10%, то качество Приемлемое, если > 10%, то качество Неприемлемое.

NFT

При проведении полного теста нового функционала качество отдельно проверенного функционала может быть: Высокое, Среднее, Низкое.

В отчете следует отдельно указывать информацию о качестве каждой части нового функционала. В этой части отчета должна быть следующая информация:

- Дана общая оценка реализации нового функционала (сгруппированная по качеству).
- Подробная (детальная) информация о качестве каждой из частей новой функциональности.
- Проведен анализ каждой из новых функций в отдельности.
- Даны ясные пояснения о выставлении соответствующего качества.
- Даны рекомендации по улучшению качества (какие проблемы следует исправить).
- Показана таблица с новыми функциями (название), их качеством, статусом функции из CQ.

AT, MAT, Regression

Если проводились тесты указанных уровней, то в первую очередь при написании отчета нужно анализировать динамику изменения качества проверенной функциональности в сравнении с более ранними версиями сборки.

Также как и у предыдущего вида тестов, качество этих может быть: Высокое, Среднее, Низкое.

Для указанных видов тестов в данной части отчета должна быть описана информация следующего характера:

- Дана сравнительная характеристика каждой из частей функционала в сравнении с предыдущими версиями сборки.
- Подробная (детальная) информация о качестве каждой из частей проверенной функциональности.
- Даны ясные пояснения о выставлении соответствующего качества каждой функции в отдельности.
- Даны рекомендации по улучшению качества (какие проблемы следует исправить).

Окончание содержимого

В завершении содержимое отчета должно включать в себя информацию следующего характера:

- Ссылка на тест-план.
- Ссылка на документ feature matrix (если таковой имеется).
- Ссылка на документ со статистикой (если таковой имеется).

- Общее количество всех новых дефектов.
- Подпись высылающего отчет.

Данные ссылки должны быть корректными, необходимо проверить достоверную ли информацию получает пользователь, открывший ссылку. Следует обращать особое внимание на подпись, удостоверьтесь, что указана именно ваша подпись либо какая-то универсальная для определенного проекта подпись.

Порядок выполнения работы

1. Получить задание у преподавателя.
 2. Составить итоговый отчет по результатам тестирования web- приложения.
 3. Оформить отчет и защитить лабораторную работу.
- Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Итоговый отчет о результатах тестирования web-приложения.
4. Выводы по работе.

Контрольные вопросы

1. Какая структура итогового отчета о результатах тестирования?
2. Что содержится в разделе Приветствие?
3. Что содержится в разделе Общая информация?
4. Что содержится в разделе Тестовое окружение?
5. Что содержится в разделе Рекомендации QA?
6. Что содержится в разделе Детализированная информация?
7. Что содержится в разделе Окончание содержимого?

Тема 1.6. Методы отладки и тестирования **Документирование результатов тестирования**

Цель работы: составить итоговый отчет о результатах тестирования web-приложения.

Теоретические сведения

Итоговый отчет можно разделить на части с соответствующей информацией:

- Приветствие.
- Общая информация (Common Information).
- Тестовое окружение (Test Platform).
- Рекомендации QA (QA Recommendations).
- Детализированная информация (Detailed Information).
- Окончание содержимого.

Приветствие

Свое письмо с отчетом необходимо начать с приветствия всех адресатов.

Если по каким-либо причинам произошла задержка данных отчета, либо не весь запланированный функционал был проверен, то эту информацию необходимо предоставить в начале письма. Следует извиниться за задержку и указать адекватные причины произошедшего. Также в самом начале письма следует указывать, если были

какие-то внешние факторы, препятствующие проверке какой-то части функционала.

Если во время тестирования не произошло никаких форс-мажорных обстоятельств, то достаточно обычного вежливого приветствия и далее уже переход к следующим пунктам.

Общая информация (Common Information)

В данной части отчета описывается, какие виды тестов проводились. Зачастую указываются модули, которые тестировались или функционал. Стоит удостовериться, не забыта ли какая-то часть функционала, особенно это актуально, когда нужно собрать итоговый отчет, соединив в себе данные о разных видах тестов и функционале.

Тестовое окружение (Test Platform)

Как правило, в этой части указываются:

- Название проекта.
- Номер сборки.
- Ссылка на проект (сборку). Необходимо убедиться, что зайдя по этой ссылке вы действительно попадаете на проект или можете установить приложение.

При указании данных в этой части отчета нужно быть очень внимательным, т.к. неправильная ссылка на сборку или неверный номер сборки не дают достоверной информации всем заинтересованным людям, а также затрудняют работу человеку, собирающему финальный отчет.

Рекомендации QA (QA Recommendations)

Данная часть отчета является наиболее важной, т.к. здесь отражается общее состояние сборки. Здесь показывается аналитическая работа тестировщика, его рекомендации по улучшению функционала, наиболее слабые места и наиболее критичные дефекты, динамика изменения качества проекта.

В этом разделе должна быть информация о следующем:

- Указан функционал (часть функционала), который заблокирован для проверки. Даны пояснения почему этот функционал не проверен (указаны наиболее критичные дефекты).
- Произведен анализ качества проверенного функционала. Следует указать, улучшилось оно или ухудшилось по сравнению с предыдущей версией, какое качество на сегодняшний момент, какие факторы повлияли на выставление именно такого качества сборки.
- Если качество сборки ухудшилось, то обязательно должны быть указаны регрессионные места.
- Наиболее нестабильные части функционала следует выделить и указать причину, по которой они таковыми являются.
- Даны рекомендации по тому функционалу и дефектам, скорейшее исправление которых является наиболее приоритетным.
- Список наиболее критичных для сборки дефектов, с указанием названия и их критичности.
- Для отчета уровня Smoke обязательно указать весь нестабильный функционал.

Если сборка является релизной или предрелизной, то любое ухудшение качества является критичным и важно об этом сообщить менеджеру как можно раньше. Помимо всего вышеуказанного для релизных и предрелизных сборок в отчете о качестве продукта важно указывать следующее:

- Дана информация о всех проблемах, характерных сборке. Проведен анализ, насколько оставшиеся проблемы являются критичными для конечного

пользователя.

- Указаны дефекты, которые следует исправить, чтобы качество конечной сборки было выше.

Детализированная информация (Detailed Information)

В данной части отчета описывается более подробная информация о проверенных частях функционала, устанавливается качество каждой проверенной части функционала(модуля) в отдельности. В зависимости от типа проводимых тестов, эта часть отчета будет отличаться.

Smoke

При оценке качества функционала на уровне Smoke теста, оно может быть либо Приемлемым, либо Неприемлемым. Качество сборки зависит от нескольких факторов:

- Если это релизная или предрелизная сборка, то для выставления Приемлемого качества на уровне Smoke не должно быть найдено функциональных дефектов.
- Наличие нового функционала. Новый функционал, который впервые поставляется на тестирование, не должен содержать дефектов уровня Smoke для выставления Приемлемого качества всей сборки.
- Чтобы установить сборке Приемлемое качество, не должно быть дефектов уровня Smoke у того функционала, по которому планируется проводить полные тесты.
- Все наиболее важные части функционала обрабатывают корректно, тогда качество всего функционала на уровне Smoke может быть оценено, как Приемлемое.

В части о детализированной информации качества сборки следует более подробно описать проблемы, которые были найдены во время теста.

DV

В этой части отчета указывается качество о проведении валидации дефектов.

Здесь должна быть следующая информация:

- Общее количество всех дефектов, поступивших на проверку.
- Количество неисправленных дефектов и их процент от общего количества.
- Список дефектов, которые не были проверены и причины, по которым этого не было сделано.
- Наглядная таблица с неисправленными дефектами.

По вышеуказанным результатам выставляется качество теста. Если процент неисправленных дефектов < 10%, то качество Приемлемое, если > 10%, то качество Неприемлемое.

NFT

При проведении полного теста нового функционала качество отдельно проверенного функционала может быть: Высокое, Среднее, Низкое.

В отчете следует отдельно указывать информацию о качестве каждой части нового функционала. В этой части отчета должна быть следующая информация:

- Дана общая оценка реализации нового функционала (сгруппированная по качеству).
- Подробная (детальная) информация о качестве каждой из частей новой функциональности.
- Проведен анализ каждой из новых функций в отдельности.
- Даны ясные пояснения о выставлении соответствующего качества.

- Даны рекомендации по улучшению качества (какие проблемы следует исправить).
- Показана таблица с новыми функциями (название), их качеством, статусом функции из CQ.

AT, MAT, Regression

Если проводились тесты указанных уровней, то в первую очередь при написании отчета нужно анализировать динамику изменения качества проверенной функциональности в сравнении с более ранними версиями сборки.

Также как и у предыдущего вида тестов, качество этих может быть: Высокое, Среднее, Низкое.

Для указанных видов тестов в данной части отчета должна быть описана информация следующего характера:

- Дана сравнительная характеристика каждой из частей функционала в сравнении с предыдущими версиями сборки.
- Подробная (детальная) информация о качестве каждой из частей проверенной функциональности.
- Даны ясные пояснения о выставлении соответствующего качества каждой функции в отдельности.
- Даны рекомендации по улучшению качества (какие проблемы следует исправить).

Окончание содержимого

В завершении содержимое отчета должно включать в себя информацию следующего характера:

- Ссылка на тест-план.
- Ссылка на документ feature matrix (если таковой имеется).
- Ссылка на документ со статистикой (если таковой имеется).
- Общее количество всех новых дефектов.

Тема 1.7.

Автоматизированное тестирование Тема 1.7.

- Автоматизированное тестирование Подпись высылающего отчет.

Данные ссылки должны быть корректными, необходимо проверить достоверную ли информацию получает пользователь, открывший ссылку. Следует обращать особое внимание на подпись, удостоверьтесь, что указана именно ваша подпись либо какая-то универсальная для определенного проекта подпись.

Порядок выполнения работы

4. Получить задание у преподавателя.
 5. Составить итоговый отчет по результатам тестирования web- приложения.
 6. Оформить отчет и защитить лабораторную работу.
- Содержание отчета

5. Цель работы.
6. Краткие теоретические сведения.
7. Итоговый отчет о результатах тестирования web-приложения.

8. Выводы по работе.
Контрольные вопросы

8. Какая структура итогового отчета о результатах тестирования?
9. Что содержится в разделе Приветствие?
10. Что содержится в разделе Общая информация?
11. Что содержится в разделе Тестовое окружение?
12. Что содержится в разделе Рекомендации QA?
13. Что содержится в разделе Детализированная информация?
14. Что содержится в разделе Окончание содержимого?

Тема 1.7. Автоматизированное тестирование

Ответить на вопросы (с примерными ответами)

1. Что такое автоматизированное тестирование?

Автоматизированное тестирование или автоматизация тестирования – это метод тестирования программного обеспечения, который выполняется с использованием специальных программных средств, которые, в свою очередь необходимы для выполнения набора тестовых примеров. Напротив, ручное тестирование выполняется человеком, сидящим перед компьютером и тщательно выполняющим каждый шаг теста «руками».

Программное обеспечение для автоматизации тестирования также может вводить тестовые данные в тестовую среду, сравнивать ожидаемые и фактические результаты и создавать подробные отчеты о тестах. Как правило, автоматизация тестирования требует значительных вложений денег и ресурсов.

2. Зачем нужна автоматизация?

Это хороший способ повысить эффективность, а также увеличить охват и скорость тестирования программного обеспечения, когда вам нужно повторять одни и те же тестовые сценарии.

- Ручное тестирование всех рабочих процессов, полей и негативных сценариев требует больше времени и денег (при определенных условиях).
- Сложно тестировать многоязычные сайты вручную.
- Не требует вмешательства человека. Запускаете и переходите к другим задачам.
- Увеличивает скорость выполнения тестов.
- Помогает увеличить охват тестированием.
- Ручное тестирование может наскучить, и следствиями станут потеря вовлеченности и появление ошибок.

3. Какие тестовые случаи стоит автоматизировать?

Для увеличения рентабельности инвестиций в автоматизацию тестовые случаи для автоматизации можно выбрать по следующим критериям:

- Высокие риски и сбои недопустимы – крайне актуально для банковской сферы.
- Тестовые сценарии, которые регулярно повторяются.
- Тестовые сценарии, которые очень сложны и утомительны для выполнения вручную.
- Тестовые примеры, отнимающие много времени.

Следующая категория тестовых случаев не подходит для автоматизации:

- Новые тестовые примеры, которые не выполнялись вручную хотя бы один раз.
- Сценарии тестирования, требования к которым часто меняются.
- Тестовые примеры, которые выполняются на разовой основе.

4.Процесс автоматизированного тестирования:

В процессе автоматизации выполняются следующие шаги:

- А)Выбор тестового инструмента
- Б)Определение объема автоматизации
- В)Планирование, дизайн и разработка
- Г)Выполнение теста
- Д)Техническое обслуживание

5.Выбор инструмента тестирования.

Выбор средства тестирования во многом зависит от технологии, на которой построено тестируемое приложение. Например, QTP не поддерживает Informatica. Таким образом, QTP нельзя использовать для тестирования приложений Informatica. Хорошая идея – провести Proof of Concept of Tool (демонстрация практической осуществимости) на AUT.

Определяем объем автоматизации. Объем автоматизации – это область тестируемого приложения, которая будет автоматизирована. Его помогают определить следующие пункты:

- Функции, важные для бизнеса
- Сценарии с большим объемом данных
- Общие функции приложений
- Техническая осуществимость
- Частота повторного использования бизнес-компонентов
- Сложность тестовых случаев
- Возможность использовать одни и те же тестовые сценарии для кросс-браузерного тестирования

6.Планирование, проектирование и разработка. На этом этапе вы создаете стратегию и план автоматизации, которые содержат следующие детали:

- Выбранные инструменты автоматизации
- Конструкция каркаса и его особенности
- Входящие и выходящие за рамки элементы автоматизации
- Подготовка стендов автоматизации
- График и временная шкала сценариев и выполнения
- Результаты тестирования автоматизации

7.Выполнение теста.

На этом этапе выполняются сценарии автоматизации. Сценариям необходимо ввести тестовые данные, прежде чем они будут запущены. После выполнения они предоставляют подробные отчеты об испытаниях

Выполнение может быть выполнено с использованием инструмента автоматизации напрямую или с помощью инструмента управления тестированием, который вызовет инструмент автоматизации.

Пример: Центр качества – это инструмент управления тестированием, который, в свою очередь, вызывает QTP для выполнения сценариев автоматизации. Скрипты могут выполняться на одной машине или на группе машин. Для экономии времени тестирование можно проводить ночью.

8.Обслуживание автоматизированного тестирования

Этот этап автоматизированного тестирования проводится для проверки того, как работают новые функции, добавленные в программное обеспечение: нормально или нет.

Сопровождение в автотестировании выполняется, когда добавляются новые сценарии автоматизации, и их необходимо проверять и поддерживать, чтобы повышать эффективность сценариев автоматизации с каждым последующим циклом выпуска.

9. Платформа для автоматизации

Фреймворк – это набор руководств по автоматизации, которые:

- поддерживают последовательность тестирования;
- улучшают структурирование теста;
- позволяют использовать минимальное количество кода;
- уменьшают затраты на обслуживание кода;
- повышают удобство повторного использования;
- дают возможность нетехническим тестировщикам участвовать в кодировании тестов;
- помогают сократить срок обучения использованию инструмента;
- включают данные везде, где это необходимо.

Для автоматизации тестирования программного обеспечения используют четыре типа фреймворков:

1. платформа автоматизации на основе данных;
2. фреймворк автоматизации на основе ключевых слов;
3. модульная платформа автоматизации;
4. гибридная среда автоматизации.

10. Рекомендации для эффективной автоматизации тестирования

Чтобы получить максимальную рентабельность инвестиций в автоматизацию, соблюдайте следующие правила:

- Объем автоматизации необходимо детально определить до начала проекта. Это позволит убедиться, что ожидания от автоматизации будут оправданы.
- Определите правильный инструмент автоматизации: инструмент не должен выбираться на основании его популярности, он должен соответствовать требованиям автоматизации на конкретном проекте.
- Выберите подходящий фреймворк.
- Стандарты создания сценариев. При написании сценариев для автоматизации необходимо соблюдать стандарты. Вот некоторые из них:
 - создайте единые скрипты, комментарии и отступы кода;
 - разработайте правила наименования тестовых сценариев;
 - прикладывайте необходимые документы, если, например, сложно понять прохождение тестового сценария без скриншота и/или спецификации.
- Определите метрики и следите за ними. Успех автоматизации нельзя определить лишь путем сравнения затраченных усилий, на тот или иной вид тестирования. Вот основные показатели:
 - процент обнаруженных дефектов;
 - время, необходимое для тестирования автоматизации выпуска каждого нового цикла;
 - минимальное время требуемое для выпуска;
 - индекс удовлетворенности клиентов;
 - улучшение производительности.

Приведенные выше рекомендации, если их соблюдать, позволят качественно выполнить автоматизацию тестирования.

11. Преимущества автоматизации тестирования

- На 70% быстрее, чем при ручном тестировании.
- Более широкий тестовый охват функций приложения.

- Надежные в результаты.
- Обеспечивает согласованность тестовых моделей.
- Экономит время и деньги.
- Повышает точность.
- Позволяет исполнять процесс тестирования без вмешательства человека.
- Повышает эффективность .
- увеличивает скорость исполнения тестирования.
- Повторно использует тестовые скрипты.
- Позволяет тестировать часто и тщательно.
- Большой цикл выполнения может быть достигнут за счет автоматизации.
- Сокращает время выхода продукта на рынок

12.Типы автоматизированного тестирования

- Смоук тестирование
- [Модульное тестирование](#)
- [Интеграционное тестирование](#)
- Функциональное тестирование
- Проверка ключевых слов
- Регрессионное тестирование
- Тестирование на основе данных
- Тестирование черного ящика

13.Как выбрать инструмент автоматизации?

Выбор подходящего инструмента может оказаться сложной задачей. Следующие критерии помогут вам выбрать лучший инструмент для ваших требований:

- поддержка окружающей среды;
- легкость использования;
- тестирование базы данных;
- идентификация объекта;
- тестирование изображений;
- тестирование восстановления после ошибок;
- отображение объектов;
- используемый язык сценариев;
- поддержка различных типов тестирования, в том числе функционального, тестового управления, мобильного и т. д.;
- поддержка нескольких фреймворков тестирования;
- легко отлаживать сценарии программного обеспечения автоматизации;
- умение распознавать предметы в любой среде;
- обширные отчеты об испытаниях и их результаты;
- минимизация затрат на обучение выбранным инструментам.

Выбор инструмента – одна из самых серьезных проблем, которую необходимо решить, прежде чем приступить непосредственно к автоматизации. Во-первых, определите требования, изучите различные инструменты и их возможности, установите ожидания от инструмента и сделайте Proof Of Concept.

14.Инструменты автоматизации тестирования

На рынке доступно множество инструментов для функционального и регрессионного тестирования. В следующих выпусках расскажем в пользу чего делают выбор наши практикующие специалисты.

[Ranorex Studio](#)

Это универсальный инструмент для автоматизации функциональных тестов пользовательского интерфейса, регрессионных тестов, тестов на основе данных и многого другого. Ranorex Studio включает простой в использовании интерфейс для автоматизации тестирования веб-приложений, настольных и мобильных приложений.

Особенности:

- Функциональный пользовательский интерфейс и сквозное тестирование на ПК, в Интернете и на мобильных устройствах
- Кроссбраузерное тестирование
- SAP, ERP, Delphi и унаследованные приложения.
- iOS и Android
- Запускайте тесты локально или удаленно, параллельно или распределяйте в Selenium Grid
- Надежная отчетность

Testim

«Самый быстрый путь к отказоустойчивым сквозным тестам – без кода, с кодированием или и тем, и другим. Testim позволяет создавать удивительно стабильные тесты без кода, которые используют наш ИИ, а также гибкость для экспорта тестов в виде кода. Такие клиенты, как Microsoft, NetApp, Wix и JFrog, ежемесячно проводят миллионы тестов на Testim.**Особенности**

- Вы можете использовать современный JavaScript API от Testim и свою IDE для отладки, настройки или рефакторинга тестов.
- Храните тесты в своей системе управления версиями, чтобы синхронизировать их с ветвями и запускать тесты при каждой фиксации.
- Интеграция с популярными инструментами»

21 Labs

«Это сложная самообучающаяся платформа автоматизации тестирования и аналитики для приложений iOS и Android.

Особенности:

- Быстрая и интеллектуальная разработка – создание с помощью ИИ дает пользователям возможность создавать автоматизированные функциональные тесты и тесты пользовательского интерфейса за считанные минуты.
- Результаты, которым вы доверяете – бесшовная система алгоритмических локаторов обеспечивает стабильные результаты во всех средах.
- Устранение проблем с обслуживанием и нестабильных результатов – самообучающееся обслуживание автоматически обновляет тесты и гарантирует, что ваша команда может сосредоточиться на разработке новых функций, полагаясь на результаты тестов.
- Выпускайте с уверенностью – производственная интеграция закрывает цикл обратной связи и анализирует фактическое покрытие. Используйте данные при выпуске.
- Полностью SaaS, не требует установки или устройств для создания или выполнения тестов. Предлагает беспрепятственный доступ к десяткам устройств».

Selenium

Это инструмент тестирования программного обеспечения, используемый для регрессионного тестирования. Это инструмент тестирования с открытым исходным кодом, который предоставляет возможность воспроизведения и записи для регрессионного тестирования. Селен IDE поддерживает только Mozilla Firefox веб – браузер

Особенности:

- Он обеспечивает возможность экспорта записанного скрипта на других языках, таких как Java, Ruby, RSpec, Python, C# и т. д.
- Его можно использовать с такими фреймворками, как JUnit и TestNG.

- Он может выполнять несколько тестов одновременно Автозаполнение для общих команд Selenium
- Пошаговые тесты
- Идентифицирует элемент с помощью идентификатора, имени, X-пути и т. Д. Храните тесты как Ruby Script, HTML и любой другой формат
- Он предоставляет возможность утверждать заголовков для каждой страницы
- Он поддерживает файл selenium user-extensions.js
- Это позволяет вставлять комментарии в середину скрипта для лучшего понимания и отладки.

QTP (MicroFocus UFT)

Широко используется для функционального и регрессионного тестирования, он касается всех основных программных приложений и сред. Чтобы упростить создание и обслуживание тестов, в нем используется концепция тестирования, управляемого ключевыми словами. Это позволяет тестировщику создавать тестовые примеры прямо из приложения.

Особенности:

- Нетехническому человеку проще адаптироваться и создавать рабочие тестовые примеры.
- Он быстрее устраняет дефекты, тщательно документируя и воспроизводя дефекты для разработчика.
- Сверхните создание тестов и документацию по тестам на одном сайте
- Параметризация проще, чем в WinRunner
- QTP поддерживает среду разработки .NET
- У него лучший механизм идентификации объекта
- Он может улучшить существующие сценарии QTP без доступности «Тестируемого приложения», используя активный экран.

Rational Functional Tester

Это объектно-ориентированный инструмент автоматизированного функционального тестирования, способный выполнять автоматическое функциональное, регрессионное тестирование, тестирование на основе данных и тестирование графического интерфейса. Основные особенности этого инструмента:

Особенности:

- Поддерживает широкий спектр протоколов и приложений, таких как Java, HTML, NET, Windows, SAP, Visual Basic и т. д.
- Может записывать и воспроизводить действия по запросу
- Он хорошо интегрируется с инструментами управления исходным кодом, такими как Rational Clear Case и Rational Team Concert. Он позволяет разработчикам создавать скрипт, связанный с ключевыми словами, чтобы его можно было использовать повторно. Редактор Eclipse Java Developer Toolkit
- Помогает команде кодировать тестовые сценарии на Java с помощью Eclipse.
- Поддерживает настраиваемые элементы управления через прокси SDK (Java / .Net)
- Поддерживает управление версиями, чтобы обеспечить параллельную разработку тестовых сценариев и одновременное использование географически распределенной командой.

Watir

Это программное обеспечение с открытым исходным кодом для регрессионного тестирования. Это позволяет вам писать тесты, которые легко читать и поддерживать. Watir поддерживает только Internet Explorer в Windows, а веб-драйвер Watir поддерживает Chrome, Firefox, IE, Opera и т. д.

Особенности:

- Он поддерживает несколько браузеров на разных платформах.
- Вместо того, чтобы использовать собственный сценарий поставщика, он использует полнофункциональный современный язык сценариев Ruby.
- Он поддерживает ваше веб-приложение независимо от того, на чем оно разработано.

SilkTest

Silk Test предназначен для выполнения функционального и регрессионного тестирования. Для приложений электронного бизнеса шелковый тест является ведущим продуктом для функционального тестирования. Это продукт поглощения Segue Software компанией Borland в 2006 году. Это объектно-ориентированный язык, как и C ++. Он использует концепцию объекта, классов и наследования. Его основная особенность включает

Особенности:

- Он состоит из всех файлов исходных скриптов.
- Он преобразует команды сценария в команды графического интерфейса. На одном компьютере команды могут выполняться на удаленном или хост-компьютере.
- Чтобы идентифицировать движение мыши вместе с нажатиями клавиш, можно запустить Silktest. Он может использовать как методы воспроизведения и записи, так и методы описательного программирования для получения диалогов.
- Он определяет все элементы управления и окна тестируемого приложения как объекты и определяет все атрибуты и свойства каждого окна.

Раздел 2. Документирование

Тема 2.1. Документирование программного обеспечения в соответствии с ЕСПД

Проверочная работа по теме 2.1.

Единая система программной документации (ЕСПД)

Цель работы: Изучение требований к оформлению программной документации.

План работы:

1 Изучение ГОСТов, входящих в ЕСПД (Единую систему программной документации); Изучение основополагающих стандартов: Гост 19.101-77 «ЕСПД. Виды программ и программных документов»; Гост 19.103-77 «ЕСПД. Обозначение программ и программных документов»; Гост 19.104-78 СТ СЭВ 2088-80) «ЕСПД. Основные надписи». Гост 19.106-78 СТ СЭВ 2088-80) «ЕСПД. Требования к программным документам, выполненным печатным способом».

2 Изучение ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению»

3 Знакомство с ГОСТ ИСО/МЭК 12207-95

4 Изучение ГОСТ (СТ СЭВ) 19.101-77 (1626-79). ЕСПД. Виды программ и программных документов.

5 Изучение ГОСТ 19.102-77. ЕСПД. Стадии разработки.

6 Изучение ГОСТ 19.781-90 Обеспечение систем обработки информации программное. Термины и определения.

7 Изучение **ГОСТ Р ИСО/МЭК 9294-93** Информационная технология. Руководство по управлению документированием программного обеспечения.

8 Изучить ГОСТ Р ИСО 9127-94 Системы обработки информации. Документация пользователя и информация на упаковке для потребительских программных пакетов.

Содержание отчета:

В рабочей тетради должны быть:

- 1 Наименование и цель работы.
- 2 Порядок выполнения работы.
- 3 Ответы на контрольные вопросы.

Основу отечественной нормативной базы в области документирования программ составляет комплекс стандартов Единой системы программной документации (ЕСПД). Стандарты ЕСПД упорядочивают процесс документирования программных систем.

Стандарты ЕСПД охватывают ту часть документации, которая создается в процессе разработки программы, и связаны с документированием функциональных характеристик.

Кроме ЕСПД в официальной нормативной базе РФ в области документирования программ и в смежных областях есть ряд перспективных стандартов, например:

- международный стандарт **ISO/IEC 12207: 1995-08-01** на организацию жизненного цикла продуктов программного обеспечения;
- стандарты комплекса ГОСТ 34 на создание и развитие автоматизированных систем.

Стандарты ЕСПД подразделяют на группы, приведенные в таблице:

Код группы	Наименование группы
	Общие положения
	Основополагающие стандарты
	Правила выполнения документации разработки
	Правила выполнения документации изготовления
	Правила выполнения документации сопровождения
	Правила выполнения эксплуатационной документации
	Правила обращения программной документации
	Резервные группы
	Прочие стандарты

Обозначение стандарта ЕСПД строят по классификационному признаку: Обозначение стандарта ЕСПД должно состоять из:

1. числа 19 (присвоенных классу стандартов ЕСПД);
2. одной цифры (после точки), обозначающей код классификационной группы стандартов, указанной таблице;
3. двузначного числа (после тире), указывающего год регистрации стандарта.
4. ОСТ 19.001-77. ЕСПД. Общие положения.
5. ГОСТ 19.003-80. ЕСПД. Схемы алгоритмов и программ. Обозначения условные графические.
6. ГОСТ 19.005-85. ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.
7. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов.

8. ГОСТ 19.102-77. ЕСПД. Стадии разработки.
9. ГОСТ 19.103-77. ЕСПД. Обозначение программ и программных документов.
10. ГОСТ 19.104-78. ЕСПД. Основные надписи.
11. ГОСТ 19.105-78. ЕСПД. Общие требования к программным документам.
12. ГОСТ 19.106-78. ЕСПД. Требования к программным документам, выполненным печатным способом.
13. ГОСТ 19.201-78. ЕСПД. Техническое задание. Требования к содержанию и оформлению.
14. ГОСТ 19.202-78. ЕСПД. Спецификация. Требования к содержанию и оформлению.
15. ГОСТ 19.301-79. ЕСПД. Программа и методика испытаний. Требования к содержанию и оформлению.
16. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.
17. ГОСТ 19.402-78. ЕСПД. Описание программы.
18. ГОСТ 19.403-79. ЕСПД. Ведомость держателей подлинников.
19. ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
20. ГОСТ 19.501-78. ЕСПД. Формуляр. Требования к содержанию и оформлению.
21. ГОСТ 19.502-78. ЕСПД. Описание применения. Требования к содержанию и оформлению.
22. ГОСТ 19.503-79. ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.
23. ГОСТ 19.504-79. ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
24. ГОСТ 19.505-79. ЕСПД. Руководство оператора. Требования к содержанию и оформлению.
25. ГОСТ 19.506-79. ЕСПД. Описание языка. Требования к содержанию и оформлению.
26. ГОСТ 19.507-79. ЕСПД. Ведомость эксплуатационных документов.
27. ГОСТ 19.508-79. ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению.
28. ГОСТ 19.601-78. ЕСПД. Общие правила дублирования, учета и хранения.
29. ГОСТ 19.602-78. ЕСПД. Правила дублирования, учета и хранения программных документов, выполненных печатным способом.
30. ГОСТ 19.603-78. ЕСПД. Общие правила внесения изменений.
31. ГОСТ 19.604-78. ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом.
32. ГОСТ 19.701-90 (ИСО 5807-85). ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
33. ГОСТ 19.781-90. Обеспечение систем обработки информации программное.

Контрольные вопросы:

1 Какие документы относят к программным? Какие сведения они должны содержать в соответствии с ГОСТ 19.103-77 «Обозначение программ и программных документов»?

2 Что представляет собой состав и структура программного документа в соответствии с ГОСТ 19.105-78?

3 В какой последовательности располагают материалы программного документа, выполненным печатным способом, согласно ГОСТ 19.106-78* (СТ СЭВ 2088-80)?

4 Что устанавливает ГОСТ 19.104-78*(СТ СЭВ 2088-80) «ОСНОВНЫЕ НАДПИСИ»?

5 Какие структурные данные входят в состав основных надписей листа утверждения и титульного листа в программных документах в соответствии с ГОСТ 19.104-78* (СТ СЭВ 2088-80)?

6 Что устанавливает ГОСТ (СТ СЭВ) 19.101-77 (1626-79). ЕСПД. Виды программ и программных документов)?

ЗАДАНИЕ 2

2.1 Изучить ГОСТ 19.102-77. ЕСПД. Стадии разработки.

Из ГОСТ 19.102-77 законспектировать Стадии разработки программы, этапы и содержание работ.

2.2 Первым стандартом, который можно использовать при формировании технических заданий на программирование является ГОСТ (СТ СЭВ) 19.201-78 (1626-79). ЕСПД. (Техническое задание. Требования к содержанию и оформлению.).

Ответить на вопрос: Какие разделы должно содержать Техническое задание согласно ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению»?

2.3 Вторым стандартом является ГОСТ (СТ СЭВ) 19.101-77 (1626-79). ЕСПД. Виды программ и программных документов). Изучить этот стандарт и ответить на вопрос:

Что устанавливает этот стандарт для вычислительных машин, комплексов и систем независимо от их назначения и области применения?

ЗАДАНИЕ 3

3.1 Изучить ГОСТ Р ИСО/МЭК 9294-93 Информационная технология. Руководство по управлению документированием программного обеспечения.

Ответить на вопрос: Что является Целью стандарта ГОСТ Р ИСО/МЭК 9294-93 и что он устанавливает?

3.2 Изучить ГОСТ Р ИСО/МЭК 8631-94 Информационная технология . Программные конструктивы и условные обозначения для их представления.

Ответить на вопрос: Что этот ГОСТ описывает?

ЗАДАНИЕ 4

Изучить ГОСТ Р ИСО 9127-94 Системы обработки информации. Документация пользователя и информация на упаковке для потребительских программных пакетов. Ответить на вопрос: Что представляет собой информация на упаковке для потребительских программных пакетов?

ЗАДАНИЕ 5

Выписать в тетрадь, что конкретно устанавливают следующие ГОСТы (Стандарты для наведения порядка в программном хозяйстве)

ГОСТ 19.202-78 ЕСПД. Спецификация

ГОСТ 19.401-78 ЕСПД. Текст программы

ГОСТ 19.403-79 ЕСПД. Ведомость держателей подлинников

ГОСТ 19.501-78 ЕСПД. Формуляр

ГОСТ 19.507-79 ЕСПД. Ведомость эксплуатационных документов

Новые Государственные стандарты РФ (ГОСТ Р)

В РФ действует ряд стандартов в части документирования ПС, разработанных на основе прямого применения международных стандартов ИСО. Это — самые «свежие» по времени принятия стандарты. Некоторые из них напрямую адресованы

руководителям проекта или директорам информационных служб. Вместе с тем они неоправданно мало известны в среде профессионалов. Вот их представление. вспомогательных процессов ЖЦ ПО

7.4 Оценочные средства для проведения промежуточной аттестации

7.4.1. Вопросы к дифференцированному зачету по Разделу 1

Вопрос 1

Является ли программа аналогом математической формулы?

Варианты ответов

- Да
- Нет
- Математические формулы и программы не сводятся друг к другу

Вопрос 2

Какие подходы используются для обоснования истинности программ?

Варианты ответов

- использование аналогий
- эксперимент над программой
- доказательство программы
- формальный и интерпретационный

Вопрос 3

Отметьте верные утверждения

Варианты ответов

- тестирование – процесс поиска ошибок
- в фазу тестирования входят поиски и исправление ошибок
- отладка – процесс локализации и исправления ошибок

Вопрос 4

Зачем нужна спецификация тестирования?

Варианты ответов

- для формирования команды тестировщиков
- для разработки тестового набора
- для понимания смысла программы

Вопрос 5

Какие существуют методы анализа и локализации ошибки?

Варианты ответов

- выполнение программы в уме
- пошаговое выполнение
- метод контрольных точек и анализа трасс

Вопрос 6

Зачем нужен Log-файл?

Варианты ответов

- для изучения результатов тестирования в режиме on-line
- для фиксации результатов прогона test-suite

- для записи комментариев после прогона тестов

Вопрос 7

Какие существуют фазы процесса тестирования?

Варианты ответов

- разработка тестового набора
- прогон программы на тестовом наборе
- доказательство правильности программы
- анализ результатов тестирования

Вопрос 8

Каковы особенности разработки тестового набора?

Варианты ответов

- определение областей эквивалентности входных параметров
- анализ покрытия тестами всех возможных случаев поведения
- проверка граничных значений

Вопрос 9

Что такое управляющий граф программы (УГП)?

Варианты ответов

- множество операторов программы.
- граф, вершины которого кодируют операторы программы, а дуги - управления (порядок исполнения) операторов
- множество операторов управления

Вопрос 10

Что такое путь в УГП(управляющий граф программы)?

Варианты ответов

- множество связанных дуг УГП
- последовательность вершин и дуг УГП с фиксированными начальной и конечной вершиной
- последовательность ветвей УГП с фиксированными начальной вершиной первой ветви и конечной вершиной последней ветви пути

Вопрос 11

Отметьте верные утверждения:

Варианты ответов

- нереализуемый путь недоступен при корректном исполнении программы
- нереализуемый путь недоступен всегда
- нереализуемый путь доступен при сбое
- нереализуемый путь доступен при реализации недопустимых состояний переменных программы

Вопрос 12

Возможно ли тестирование программы на всех допустимых значениях параметров?

Варианты ответов

- да, всегда
- никогда

- возможно в отдельных случаях

Вопрос 13

Какие предъявляются требования к идеальному критерию тестирования?

Варианты ответов

- достаточность
- достижимость
- полнота
- проверяемость

Вопрос 14

Какие классы критериев тестируемости известны

Варианты ответов

- структурные критерии
- мутационные критерии
- функциональные критерии
- сценарные критерии
- стохастические критерии

Вопрос 15

Назовите полный и надежный критерий для нетривиальных классов программ.

Варианты ответов

- сценарный критерий
- такого критерия не существует
- критерий «черного ящика»

Вопрос 16

Какие существуют разновидности структурных критериев?

Варианты ответов

- критерий тестирования команд
- критерий тестирования ветвей
- критерий тестирования циклов
- критерий тестирования путей

Вопрос 17

Назовите недостатки структурных критериев.

Варианты ответов

- не проверяется соответствие со спецификацией
- не проверяется соответствие со спецификацией, не зафиксированное в структуре программы
- не проверяются ошибки в структурах данных

Вопрос 18

Какие существуют разновидности функциональных критериев?

Варианты ответов

- тестирование пунктов спецификации
- тестирование классов входных данных
- тестирование классов выходных данных
- тестирование функций
- тестирование правил

Вопрос 19

Назовите недостатки функциональных критериев.

Варианты ответов

- не проверяется соответствие со спецификацией
- не проверяются ошибки, требования к которым не зафиксированы в спецификации
- не проверяются ошибки в структурах данных, требования к которым не зафиксированы в спецификации

Вопрос 20

Какой подход используется в методе мутационного тестирования?

Варианты ответов

- создание программ-мутантов на основе изменения модульной структуры основной программы
- создание программ-мутантов с функциональными дефектами
- оценка числа ошибок в программе на основе искусственно внесенных мелких ошибок

Вопрос 21

Чем отличается оценка оттестированности проекта от оценки для модуля?

Варианты ответов

- оценка проекта интегрирует оценки оттестированности модулей
- оценка проекта может вычисляться инкрементально
- в результате получаем наихудшую оценку оттестированности
- в результате получаем наилучшую оценку оттестированности

Вопрос 22

Какие существуют разновидности уровней тестирования?

Варианты ответов

- модульное
- интеграционное
- структурное
- системное
- регрессионное

Вопрос 23

Какие задачи у модульного тестирования?

Варианты ответов

- выявление ошибок при вызове модулей
- выявление ошибок взаимодействия модуля с окружением
- выявление локальных ошибок реализации алгоритмов модулей

Вопрос 24

На основе каких принципов строятся тесты для модульного тестирования?

Варианты ответов

- анализ потоков управления модуля
- анализ потоков данных модуля
- анализ покрытия в соответствии с заданными структурными критериями

Вопрос 25

Каковы фазы процесса построения тестовых путей?

Варианты ответов

- построение УГП (управляющего графа программы)
- выбор тестовых путей
- генерация тестов, соответствующих выбранным тестовым путям

Вопрос 26

Какие существуют методы построения тестовых путей?

Варианты ответов

- статические
- динамические
- методы реализуемых путей

Вопрос 27

Какие существуют разновидности интеграционного тестирования?

Варианты ответов

- Регрессионное тестирование
- монолитное тестирование
- нисходящее тестирование
- восходящее тестирование

Вопрос 28

Каковы особенности нисходящего тестирования?

Варианты ответов

- необходимость разработки заглушек
- параллельная разработка эффективных модулей
- необходимость разработки среды управления очередностью вызовов модулей
- необходимость разработки драйверов

Вопрос 29

Каковы особенности системного тестирования?

Варианты ответов

- тесты оперируют пользовательским или другими внешними интерфейсами
- структура проекта тестируется на уровне подсистем
- тестированию подлежит система в целом
- тестирование осуществляется по методу «черного ящика»

Вопрос 30

Какие задачи решаются на этапе системного тестирования

Варианты ответов

- выявление дефектов в функционировании приложения или в работе с ним
- выявление дефектов использования ресурсов
- выявление несовместимости с окружением
- выявление непредусмотренных сценариев применения или использования непредусмотренных комбинаций данных

Вопрос 31

Каковы особенности регрессионного тестирования?

Варианты ответов

- перетестирование предусматривает только контроль частей приложения, связанных с изменениями
- выбор между полным и частичным перетестированием и пополнением тестовых наборов
- регрессионное тестирование является подмножеством системного тестирования

Вопрос 32

Какие типы дефектов выявляются при системном и регрессионном тестировании

Варианты ответов

- отсутствующая или некорректная функциональность
- непредусмотренные данные или неподдерживаемые сценарии использования
- некорректность проектной документации
- ошибки переносимости на другие платформы
- ошибки инсталляции и конфигурирования
- ошибки пользовательской документации

Вопрос 33

Можно ли гарантировать безопасность метода регрессионного тестирования, если отсутствует информация об изменениях в программе?

Варианты ответов

- да
- нет

Вопрос 34

Какие из перечисленных методов тестирования наиболее затратны

Варианты ответов

- статические методы
- модульное тестирование
- интеграционное тестирование
- системное тестирование с моделируемым окружением
- системное тестирование в реальном окружении и реальном времени

Вопрос 35

Каково содержание тестового отчета?

Варианты ответов

- перечень функциональности, запланированной на тестирование
- количество выполненных тестов и время тестирования
- количество найденных и повторно открытых дефектов
- фиксацию отклонений от процедуры тестирования
- заключение о корректировках тестового набора перед следующим циклом тестирования

Вопрос 36

Какие тестовые метрики используются при тестировании?

Варианты ответов

- покрытие функциональных требований и покрытие кода продукта
- покрытие множества сценариев
- количество и плотность найденных дефектов
- скорость нахождения дефектов

Вопрос 37

Каковы особенности документа для описания дефектов?

Варианты ответов

- номер теста, обнаруживавшего дефект
- уровень серьезности дефекта
- поле записи содержит номер build, на котором дефект был найден
- описание дефекта и описание процедуры его воспроизведения

Вопрос 38

Какие бывают состояния дефекта?

Варианты ответов

- New – дефект занесен в базу дефектов
- Open – дефект зафиксирован за разработчиком для исправления
- Resolved – дефект разработчиком исправлен
- Verified – успешное исправление дефекта подтверждено инженером

по качеству

- Postponed – решение о замораживании активности по исправлению дефекта

Вопрос 39

Какую информацию должен содержать тестовый план?

Варианты ответов

- дизайн тестовых наборов
- тестовые ресурсы
- перечень функций и подсистем, подлежащих тестированию
- тестовую стратегию
- расписание тестовых циклов
- тестовые метрики
- тестовую конфигурацию

Вопрос 40

Как определить цели тестирования программного проекта?

Варианты ответов

- какие их свойства и характеристики подлежат тестированию
 - определить части проекта, подлежащие тестированию
 - каков критерий качества тестирования
 - каков график выполнения задач тестирования
- Теоретические вопросы для промежуточной аттестации :
 1. Тестирование как часть процесса верификации программного обеспечения;
 2. Виды ошибок;
 3. Методы отладки;

- 4. Методы тестирования;
- 5. Классификация тестирования по уровням;
- 6. Тестирование производительности;
- 7. Регрессионное тестирование;
- 8. Тестирование «белым ящиком»;
- 9. Тестирование «черным ящиком»;
- 10. Модульное тестирование;
- 11. Регрессионное тестирование; 1
- 2. Оценка сложности алгоритмов сортировки;
- 13. Оценка сложности алгоритмов поиска;
- 14. Оформление документации на программные средства и использованием инструментальных средств
- 15. Средства разработки технической документации;
- 16. Технологии разработки документов;
- 17. Документирование программного обеспечения в соответствии с Единой системой программной документации;
- 18. Автоматизация разработки технической документации;
- 19. Автоматизированные средства оформления документации;
- 20. Оформление документации на программные средства с использованием инструментальных средств

7.4.2. Практические задания к зачёту

Задания для практической части экзамена

1. Дан двумерный массив 5×5 . Найти сумму модулей отрицательных нечетных элементов. Сформулировать требования к программному продукту и выполнить анализ и тестирование программных требований в соответствии со свойствами качественных требований.

2. Дана матрица. Вывести на экран все четные строки, то есть с четными номерами, у которых первый элемент больше последнего. Сформулировать требования к программному продукту и разработать чек – лист.

3. В матрице $m \times n$. Отсортировать по убыванию элементы строк, расположенные после второго отрицательного числа. Сформулировать требования к программному продукту и разработать чек – лист.

4. Дана целочисленная квадратная матрица. Определить: произведение элементов в тех строках, которые не содержат отрицательных элементов. Сформулировать требования к программному продукту и разработать тест – кейс.

5. Для заданной матрицы размером 8 на 8 найти такие k , что k -я строка матрицы совпадает с k -м столбцом. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент. Сформулировать требования к программному продукту и разработать набор тест – кейсов.

6. Две строки матрицы назовем похожими, если совпадают множества чисел, встречающихся в этих строках. Найдите все пары похожих строк в заданной матрице $m \times n$. Сформулировать требования к программному продукту и разработать набор тест – кейсов.

7. В матрице $m \times n$. Отсортировать по убыванию элементы строк, расположенные после второго отрицательного числа. Сформулировать требования к программному продукту. Выполнить тестирование программного продукта по структурным критериям.

8. В матрице $m \times n$. Отсортировать по убыванию элементы строк, расположенные после второго отрицательного числа. Сформулировать требования к программному продукту. Выполнить тестирование программного продукта по методу «белого ящика».

9. Дана матрица. Вывести на экран все четные строки, то есть с четными номерами, у которых первый элемент больше последнего. Сформулировать требования к программному продукту и выполнить тестирование программного продукта по методу «белого ящика».

10. Для заданной матрицы размером 8 на 8 найти такие k , что k -я строка матрицы совпадает с k -м столбцом. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент. Сформулировать требования к программному продукту и выполнить тестирование по методу «белого ящика».

11. Для заданной матрицы размером 8 на 8 найти такие k , что k -я строка матрицы совпадает с k -м столбцом. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент. Сформулировать требования к программному продукту и выполнить тестирование программного продукта по структурным критериям.

12. В матрице $m \times n$. Отсортировать по убыванию элементы строк, расположенные после второго отрицательного числа. Сформулировать требования к программному продукту. Выполнить тестирование программного продукта по функциональным критериям.

13. В матрице $m \times n$. Отсортировать по убыванию элементы строк, расположенные после второго отрицательного числа. Сформулировать требования к программному продукту. Выполнить тестирование программного продукта по методу «черного ящика».

14. Для заданной матрицы размером 8 на 8 найти такие k , что k -я строка матрицы совпадает с k -м столбцом. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент. Сформулировать требования к программному продукту и выполнить тестирование по методу «черного ящика».

15. Описать функцию $NMin(A,N)$ и $NMax(A,N)$ целого типа, находящую номер минимального и максимального элемента массива A (массив состоит из N вещественных чисел). Сформулировать требования к программному продукту и выполнить тестирование по методу «белого ящика».

16. Описать функцию $NMin(A,N)$ и $NMax(A,N)$ целого типа, находящую номер минимального и максимального элемента массива A (массив состоит из N вещественных чисел). Сформулировать требования к программному продукту и выполнить модульное тестирование

17. Описать функцию $NewStr(S)$, удаляющую в строке S начальные и конечные пробелы. В основной программе ввод строки, обращение методу - функции и вывод результата. Сформулировать требования к программному продукту и выполнить unit – тестирование.

18. Описать функцию $NewStr(S)$, удаляющую в строке S начальные и конечные пробелы. В основной программе ввод строки, обращение методу - функции и вывод результата. Предусмотреть использование 2 –х форм. Сформулировать требования к программному продукту и выполнить интеграционное тестирование.

19. Описать функцию $NMin(A,N)$ и $NMax(A,N)$ целого типа, находящую номер минимального и максимального элемента массива A (массив состоит из N вещественных чисел). Предусмотреть использование 2 –х форм. Сформулировать требования к программному продукту и выполнить интеграционное тестирование.

20. Дано натуральное число n и последовательность из 5 чисел. Найти количество чисел, являющихся степенями пятерки. Определить функцию пользователя, позволяющую распознавать степень пятерки. В основной программе ввод чисел, обращение к функции, вычисление количества и вывод результата. Сформулировать требования к программному продукту и выполнить модульное тестирование.

8. ДОПОЛНИТЕЛЬНОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Классификация тестирования

Лекция «Понятие процесса тестирования. Виды тестирования»
Фундаментальная теория тестирования

Тестирование IT-систем*Тестирование веб-сервисов*Тестирование мобильных приложений*Тестирование игр*

В тестировании нет четких определений, как в физике, математике, которые при перефразировании становятся абсолютно неверными. Поэтому важно понимать процессы и подходы. В данной статье разберем основные определения теории тестирования.

Перейдем к основным понятиям

Тестирование программного обеспечения (Software Testing) — проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определенным образом.

Цель тестирования — проверка соответствия ПО предъявляемым требованиям, обеспечение уверенности в качестве ПО, поиск очевидных ошибок в программном обеспечении, которые должны быть выявлены до того, как их обнаружат пользователи программы.

Для чего проводится тестирование ПО?

- Для проверки соответствия требованиям.
- Для обнаружение проблем на более ранних этапах разработки и предотвращение повышения стоимости продукта.
- Обнаружение вариантов использования, которые не были предусмотрены при разработке. А также взгляд на продукт со стороны пользователя.
- Повышение лояльности к компании и продукту, т.к. любой обнаруженный дефект негативно влияет на доверие пользователей.

Принципы тестирования

Принцип 1 — Тестирование демонстрирует наличие дефектов (Testing shows presence of defects).

Тестирование только снижает вероятность наличия дефектов, которые находятся в программном обеспечении, но не гарантирует их отсутствия.

Принцип 2 — Исчерпывающее тестирование невозможно (Exhaustive testing is impossible).

Полное тестирование с использованием всех входных комбинаций данных, результатов и предусловий физически невыполнимо (исключение — тривиальные случаи).

Принцип 3 — Раннее тестирование (Early testing).

Следует начинать тестирование на ранних стадиях жизненного цикла разработки ПО, чтобы найти дефекты как можно раньше.

Принцип 4 — Скопление дефектов (Defects clustering).

Большая часть дефектов находится в ограниченном количестве модулей.

Принцип 5 — Парадокс пестицида (Pesticide paradox).

Если повторять те же тестовые сценарии снова и снова, в какой-то момент этот набор тестов перестанет выявлять новые дефекты.

Принцип 6 — Тестирование зависит от контекста (Testing is context depending). Тестирование проводится по-разному в зависимости от контекста. Например, программное обеспечение, в котором критически важна безопасность, тестируется иначе, чем новостной портал.

Принцип 7 — Заблуждение об отсутствии ошибок (Absence-of-errors fallacy). Отсутствие найденных дефектов при тестировании не всегда означает готовность продукта к релизу. Система должна быть удобна пользователю в использовании и удовлетворять его ожиданиям и потребностям.

Обеспечение качества (QA — Quality Assurance) и контроль качества (QC — Quality Control) — эти термины похожи на взаимозаменяемые, но разница между обеспечением качества и контролем качества все-таки есть, хоть на практике процессы и имеют некоторую схожесть.

QC (Quality Control) — Контроль качества продукта — анализ результатов тестирования и качества новых версий выпускаемого продукта.

К задачам контроля качества относятся:

- проверка готовности ПО к релизу;
- проверка соответствия требований и качества данного проекта.

QA (Quality Assurance) — Обеспечение качества продукта — изучение возможностей по изменению и улучшению процесса разработки, улучшению коммуникаций в команде, где тестирование является только одним из аспектов обеспечения качества.

К задачам обеспечения качества относятся:

- проверка технических характеристик и требований к ПО;
- оценка рисков;
- планирование задач для улучшения качества продукции;
- подготовка документации, тестового окружения и данных;
- тестирование;
- анализ результатов тестирования, а также составление отчетов и других документов.



Верификация и валидация — два понятия тесно связаны с процессами тестирования и обеспечения качества. К сожалению, их часто путают, хотя отличия между ними достаточно существенны.

Верификация (verification) — это процесс оценки системы, чтобы понять, удовлетворяют ли результаты текущего этапа разработки условиям, которые были сформулированы в его начале.

Валидация (validation) — это определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, его требованиям к системе.

Пример: когда разрабатывали аэробус А310, то надо было сделать так, чтобы закрылки вставляли в положение «торможение», когда шасси коснулись земли. Запрограммировали так, что когда шасси начинают крутиться, то закрылки ставим в положение «торможение». Но вот во время испытаний в Варшаве самолет выкатился за пределы полосы, так как была мокрая поверхность. Он проскользил, только потом был крутящий момент и они, закрылки, открылись. С точки зрения «верификации» — программа сработала, с точки зрения «валидации» — нет. Поэтому код изменили так, чтобы в момент изменения давления в шинах открывались закрылки.

Документацию, которая используется на проектах по разработке ПО, можно условно разделить на две группы:

1. Проектная документация — включает в себя всё, что относится к проекту в целом.
2. Продуктовая документация — часть проектной документации, выделяемая отдельно, которая относится непосредственно к разрабатываемому приложению или системе.

Этапы тестирования:

1. Анализ продукта
2. Работа с требованиями
3. Разработка стратегии тестирования и планирование процедур контроля качества
4. Создание тестовой документации
5. Тестирование прототипа
6. Основное тестирование
7. Стабилизация

8. Эксплуатация

Стадии разработки ПО — этапы, которые проходят команды разработчиков ПО, прежде чем программа станет доступной для широкого круга пользователей.

Программный продукт проходит следующие стадии:

1. анализ требований к проекту;
2. проектирование;
3. реализация;
4. тестирование продукта;
5. внедрение и поддержка.

Требования

Требования — это спецификация (описание) того, что должно быть реализовано.

Требования описывают то, что необходимо реализовать, без детализации технической стороны решения.

Атрибуты требований:

1. **Корректность** — точное описание разрабатываемого функционала.
2. **Проверяемость** — формулировка требований таким образом, чтобы можно было выставить однозначный вердикт, выполнено все в соответствии с требованиями или нет.
3. **Полнота** — в требовании должна содержаться вся необходимая для реализации функциональности информация.
4. **Недвусмысленность** — требование должно содержать однозначные формулировки.
5. **Непротиворечивость** — требование не должно содержать внутренних противоречий и противоречий другим требованиям и документам.
6. **Приоритетность** — у каждого требования должен быть приоритет (количественная оценка степени значимости требования). Этот атрибут позволит грамотно управлять ресурсами на проекте.
7. **Атомарность** — требование нельзя разбить на отдельные части без потери деталей.
8. **Модифицируемость** — в каждое требование можно внести изменение.
9. **Прослеживаемость** — каждое требование должно иметь уникальный идентификатор, по которому на него можно сослаться.

Дефект (bug) — отклонение фактического результата от ожидаемого.

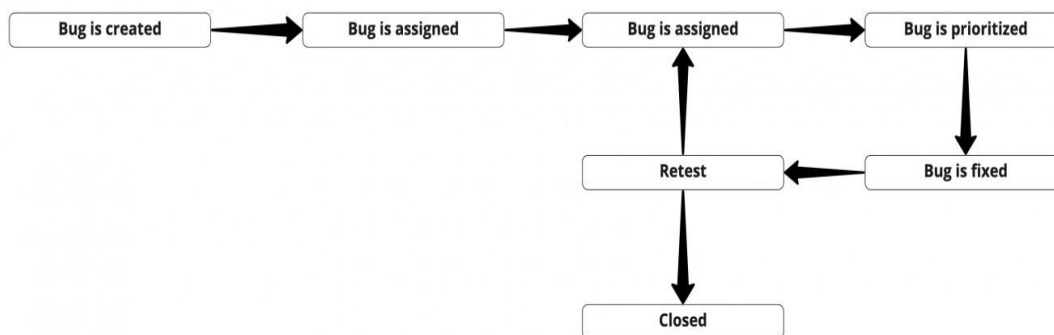
Отчёт о дефекте (bug report) — документ, который содержит отчет о любом недостатке в компоненте или системе, который потенциально может привести компонент или систему к невозможности выполнить требуемую функцию.

Атрибуты отчета о дефекте:

1. Уникальный идентификатор (ID) — присваивается автоматически системой при создании баг-репорта.
2. Тема (краткое описание, Summary) — кратко сформулированный смысл дефекта, отвечающий на вопросы: Что? Где? Когда(при каких условиях)?
3. Подробное описание (Description) — более широкое описание дефекта (указывается опционально).
4. Шаги для воспроизведения (Steps To Reproduce) — описание четкой последовательности действий, которая привела к выявлению дефекта. В шагах воспроизведения должен быть описан каждый шаг, вплоть до конкретных вводимых значений, если они играют роль в воспроизведении дефекта.
5. Фактический результат (Actual result) — описывается поведение системы на момент обнаружения дефекта в ней. чаще всего, содержит краткое описание некорректного поведения(может совпадать с темой отчета о дефекте).
6. Ожидаемый результат (Expected result) — описание того, как именно должна работать система в соответствии с документацией.
7. Вложения (Attachments) — скриншоты, видео или лог-файлы.
8. Серьёзность дефекта (важность, Severity) — характеризует влияние дефекта на работоспособность приложения.
9. Приоритет дефекта (срочность, Priority) — указывает на очерёдность выполнения задачи или устранения дефекта.
10. Статус (Status) — определяет текущее состояние дефекта. Статусы дефектов могут быть разными в разных баг-трекинг-системах.
11. Окружение (Environment) – окружение, на котором воспроизвелся баг.

Жизненный цикл бага

tg: @qa_chillout



Severity vs Priority

Серьёзность (severity) показывает степень ущерба, который наносится проекту существованием дефекта. Severity выставляется тестировщиком.

Градации Серьезности дефекта (Severity):

- **Блокирующий (S1 – Blocker)**
 - тестирование значительной части функциональности вообще недоступно. Блокирующая ошибка, приводящая приложение в нерабочее состояние, в результате которого дальнейшая работа с тестируемой системой или ее ключевыми функциями становится невозможна.
- **Критический (S2 – Critical)**
 - критическая ошибка, неправильно работающая ключевая бизнес-логика, дыра в системе безопасности, проблема, приведшая к временному падению сервера или приводящая в нерабочее состояние некоторую часть системы, то есть не работает важная часть одной какой-либо функции либо не работает значительная часть, но имеется workaround (обходной путь/другие входные точки), позволяющий продолжить тестирование.
- **Значительный (S3 – Major)**
 - не работает важная часть одной какой-либо функции/бизнес-логики, но при выполнении специфических условий, либо есть workaround, позволяющий продолжить ее тестирование либо не работает не очень значительная часть какой-либо функции. Также относится к дефектам с высокими visibility – обычно не сильно влияющие на функциональность дефекты дизайна, которые, однако, сразу бросаются в глаза.
- **Незначительный (S4 – Minor)**
 - часто ошибки GUI, которые не влияют на функциональность, но портят юзабилити или внешний вид. Также незначительные функциональные дефекты, либо которые воспроизводятся на определенном устройстве.
- **Тривиальный (S5 – Trivial)**
 - почти всегда дефекты на GUI — опечатки в тексте, несоответствие шрифта и оттенка и т.п., либо плохо воспроизводимая ошибка, не касающаяся бизнес-логики, проблема сторонних библиотек или сервисов, проблема, не оказывающая никакого влияния на общее качество продукта.

Срочность (priority) показывает, как быстро дефект должен быть устранён. Priority выставляется менеджером, тимлидом или заказчиком

- **P1 Высокий (High)**
 - Критическая для проекта ошибка. Должна быть исправлена как можно быстрее.
- **P2 Средний (Medium)**
 - Не критичная для проекта ошибка, однако требует обязательного решения.
- **P3 Низкий (Low)**
 - Наличие данной ошибки не является критичным и не требует срочного решения. Может быть исправлена, когда у команды появится время на ее устранение.

Существует шесть базовых типов задач:

- **Эпик (epic)** — большая задача, на решение которой команде нужно несколько спринтов.
- **Требование (requirement)** — задача, содержащая в себе описание реализации той или иной фичи.
- **История (story)** — часть большой задачи (эпика), которую команда может решить за 1 спринт.
- **Задача (task)** — техническая задача, которую делает один из членов команды.
- **Под-задача (sub-task)** — часть истории / задачи, которая описывает минимальный объем работы члена команды.
- **Баг (bug)** — задача, которая описывает ошибку в системе.

Тестовые среды

- 1) **Среда разработки (Development Env)** – за данную среду отвечают разработчики, в ней они пишут код, проводят отладку, исправляют ошибки
- 2) **Среда тестирования (Test Env)** – среда, в которой работают тестировщики (проверяют функционал, проводят smoke и регрессионные тесты, воспроизводят).
- 3) **Интеграционная среда (Integration Env)** – среда, в которой проводят тестирование взаимодействующих друг с другом модулей, систем, продуктов.
- 4) **Не надо!!!! - Предпрод (Preprod Env)** – среда, которая максимально приближена к продакшену. Здесь проводится заключительное тестирование функционала.
- 5) **Продакшн среда (Production Env)** – среда, в которой работают пользователи.

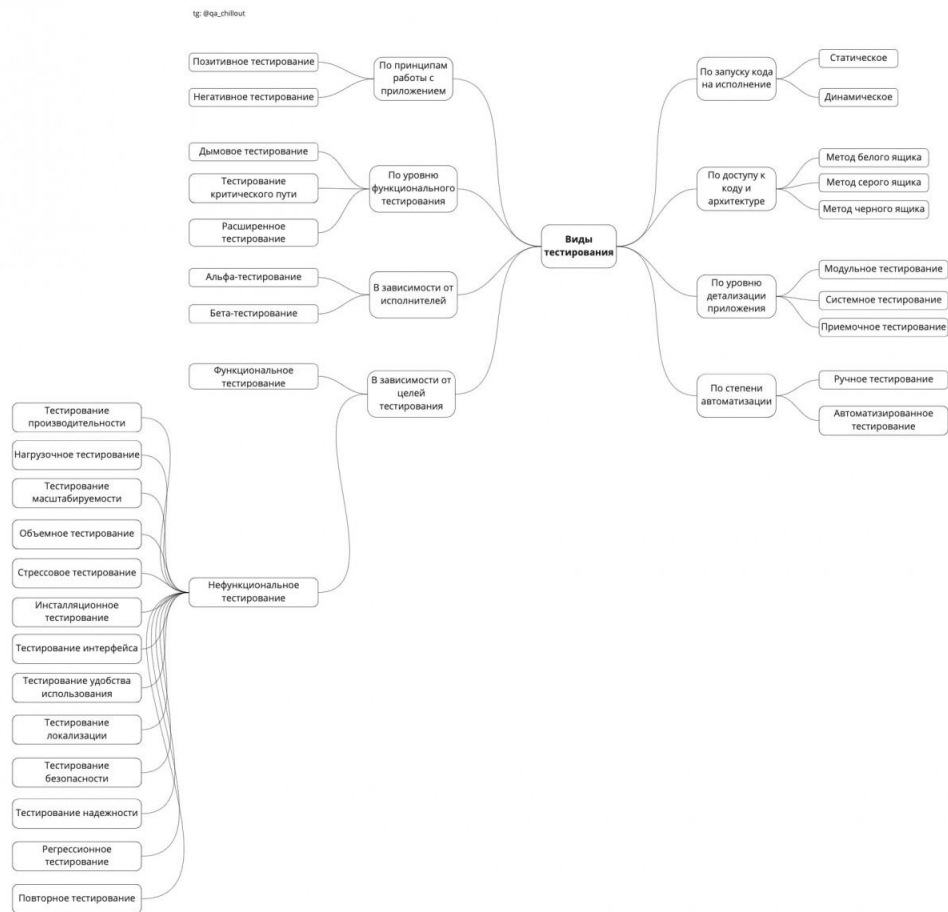
Основные фазы тестирования

Pre-Alpha: прототип, в котором всё ещё присутствует много ошибок и наверняка неполный функционал. Необходим для ознакомления с будущими возможностями программ.

- 1) **Alpha:** является ранней версией программного продукта, тестирование которой проводится внутри фирмы-разработчика.
- 2) **Beta:** практически готовый продукт, который разработан в первую очередь для тестирования конечными пользователями.
- 3) **Release Candidate (RC):** возможные ошибки в каждой из фичей уже устранены и разработчики выпускают версию на которой проводится регрессионное тестирование.
- 4) **Release:** финальная версия программы, которая готова к использованию.

Основные виды тестирования ПО

Вид тестирования — это совокупность активностей, направленных на тестирование заданных характеристик системы или её части, основанная на конкретных целях.



1. Классификация по запуску кода на исполнение:

- **Статическое тестирование** — процесс тестирования, который проводится для верификации практически любого артефакта разработки: программного кода компонент, требований, системных спецификаций, функциональных спецификаций, документов проектирования и архитектуры программных систем и их компонентов.
- **Динамическое тестирование** — тестирование проводится на работающей системе, не может быть осуществлено без запуска программного кода приложения.

2. Классификация по доступу к коду и архитектуре:

- **Тестирование белого ящика** — метод тестирования ПО, который предполагает полный доступ к коду проекта.
- **Тестирование серого ящика** — метод тестирования ПО, который предполагает частичный доступ к коду проекта (комбинация White Box и Black Box методов).

- **Тестирование чёрного ящика** — метод тестирования ПО, который не предполагает доступа (полного или частичного) к системе. Основывается на работе исключительно с внешним интерфейсом тестируемой системы.

3. Классификация по уровню детализации приложения:

- **Модульное тестирование** — проводится для тестирования какого-либо одного логически выделенного и изолированного элемента (модуля) системы в коде. Проводится самими разработчиками, так как предполагает полный доступ к коду.
- **Интеграционное тестирование** — тестирование, направленное на проверку корректности взаимодействия нескольких модулей, объединенных в единое целое.
- **Системное тестирование** — процесс тестирования системы, на котором проводится не только функциональное тестирование, но и оценка характеристик качества системы — ее устойчивости, надежности, безопасности и производительности.
- **Приёмочное тестирование** — проверяет соответствие системы потребностям, требованиям и бизнес-процессам пользователя.

4. Классификация по степени автоматизации

- Ручное тестирование.
- Автоматизированное тестирование.

5. Классификация по принципам работы с приложением

- **Позитивное тестирование** — тестирование, при котором используются только корректные данные.
- **Негативное тестирование** — тестирование приложения, при котором используются некорректные данные и выполняются некорректные операции.

6. Классификация по уровню функционального тестирования:

- **Дымовое тестирование (smoke test)** — тестирование, выполняемое на новой сборке, с целью подтверждения того, что программное обеспечение стартует и выполняет основные для бизнеса функции.
- **Тестирование критического пути (critical path)** — направлено для проверки функциональности, используемой обычными пользователями во время их повседневной деятельности.
- **Расширенное тестирование (extended)** — направлено на исследование всей заявленной в требованиях функциональности.

7. Классификация в зависимости от исполнителей:

- **Альфа-тестирование** — является ранней версией программного продукта. Может выполняться внутри организации-разработчика с возможным частичным привлечением конечных пользователей.
- **Бета-тестирование** — программное обеспечение, выпускаемое для ограниченного количества пользователей. Главная цель — получить отзывы клиентов о продукте и внести соответствующие изменения.

8. Классификация в зависимости от целей тестирования:

- **Функциональное тестирование (functional testing)** — направлено на проверку корректности работы функциональности приложения.
- **Нефункциональное тестирование (non-functional testing)** — тестирование атрибутов компонента или системы, не относящихся к функциональности:
 - 1) **Тестирование производительности (performance testing)** — определение стабильности и потребления ресурсов в условиях различных сценариев использования и нагрузок.
 - 2) **Нагрузочное тестирование (load testing)** — определение или сбор показателей производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству).
 - 3) **Тестирование масштабируемости (scalability testing)** — тестирование, которое измеряет производительность сети или системы, когда количество пользовательских запросов увеличивается или уменьшается.
 - 4) **Объёмное тестирование (volume testing)** — это тип тестирования программного обеспечения, которое проводится для тестирования программного приложения с определенным объемом данных.
 - 5) **Стрессовое тестирование (stress testing)** — тип тестирования направленный для проверки, как система обращается с нарастающей нагрузкой (количеством одновременных пользователей).
 - 6) **Инсталляционное тестирование (installation testing)** — тестирование, направленное на проверку успешной установки и настройки, обновления или удаления приложения.
 - 7) **Тестирование интерфейса (GUI/UI testing)** — проверка требований к пользовательскому интерфейсу.
 - 8) **Тестирование удобства использования (usability testing)** — это метод тестирования, направленный на установление степени удобства использования, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.
 - 9) **Тестирование локализации (localization testing)** — проверка адаптации программного обеспечения для определенной аудитории в соответствии с ее культурными особенностями.

- 10) **Тестирование безопасности (security testing)** — это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.
- 11) **Тестирование надёжности (reliability testing)** — один из видов нефункционального тестирования ПО, целью которого является проверка работоспособности приложения при длительном тестировании с ожидаемым уровнем нагрузки.
- 12) **Регрессионное тестирование (regression testing)** — тестирование уже проверенной ранее функциональности после внесения изменений в код приложения, для уверенности в том, что эти изменения не внесли ошибки в областях, которые не подверглись изменениям.
- 13) **Повторное/подтверждающее тестирование (re-testing/confirmation testing)** — тестирование, во время которого исполняются тестовые сценарии, выявившие ошибки во время последнего запуска, для подтверждения успешности исправления этих ошибок.

Тест-дизайн — это этап тестирования ПО, на котором проектируются и создаются тестовые случаи (тест-кейсы).

Техники тест-дизайна

Автор книги "[A Practitioner's Guide to Software Test Design](#)", Lee Copeland, выделяет следующие техники тест-дизайна:

1. **Тестирование на основе классов эквивалентности (equivalence partitioning)** — это техника, основанная на методе чёрного ящика, при которой мы разделяем функционал (часто диапазон возможных вводимых значений) на группы эквивалентных по своему влиянию на систему значений.
2. **Техника анализа граничных значений (boundary value testing)** — это техника проверки поведения продукта на крайних (граничных) значениях входных данных.
3. **Попарное тестирование (pairwise testing)** — это техника формирования наборов тестовых данных из полного набора входных данных в системе, которая позволяет существенно сократить количество тест-кейсов.
4. **Тестирование на основе состояний и переходов (State-Transition Testing)** — применяется для фиксирования требований и описания дизайна приложения.
5. **Таблицы принятия решений (Decision Table Testing)** — техника тестирования, основанная на методе чёрного ящика, которая применяется для систем со сложной логикой.
6. **Доменный анализ (Domain Analysis Testing)** — это техника основана на разбиении диапазона возможных значений переменной на поддиапазоны, с последующим выбором одного или нескольких значений из каждого домена для тестирования.

7. **Сценарий использования (Use Case Testing)** — Use Case описывает сценарий взаимодействия двух и более участников (как правило — пользователя и системы).

Методы тестирования

@qa_chillout



**Метод
черного ящика**



**Метод
серого ящика**



**Метод
белого ящика**

Тестирование белого ящика — метод тестирования ПО, который предполагает, что внутренняя структура/устройство/реализация системы известны тестирующему.

Согласно ISTQB, тестирование белого ящика — это:

- тестирование, основанное на анализе внутренней структуры компонента или системы;
- тест-дизайн, основанный на технике белого ящика — процедура написания или выбора тест-кейсов на основе анализа внутреннего устройства системы или компонента.
- Почему «белый ящик»? Тестируемая программа для тестирующего — прозрачный ящик, содержимое которого он прекрасно видит.

Тестирование серого ящика — метод тестирования ПО, который предполагает комбинацию White Box и Black Box подходов. То есть, внутреннее устройство программы нам известно лишь частично.

Тестирование чёрного ящика — также известное как тестирование, основанное на спецификации или тестирование поведения — техника тестирования, основанная на работе исключительно с внешними интерфейсами тестируемой системы.

Согласно ISTQB, тестирование черного ящика — это:

- тестирование, как функциональное, так и нефункциональное, не предполагающее знания внутреннего устройства компонента или системы;
- тест-дизайн, основанный на технике черного ящика — процедура написания или выбора тест-кейсов на основе анализа функциональной или нефункциональной спецификации компонента или системы без знания ее внутреннего устройства.

Тестовая документация

Тест план (Test Plan) — это документ, который описывает весь объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков.

Тест план должен отвечать на следующие вопросы:

- Что необходимо протестировать?
- Как будет проводиться тестирование?
- Когда будет проводиться тестирование?
- Критерии начала тестирования.
- Критерии окончания тестирования.

Основные пункты тест плана:

1. Идентификатор тест плана (Test plan identifier);
2. Введение (Introduction);
3. Объект тестирования (Test items);
4. Функции, которые будут протестированы (Features to be tested);
5. Функции, которые не будут протестированы (Features not to be tested);
6. Тестовые подходы (Approach);
7. Критерии прохождения тестирования (Item pass/fail criteria);
8. Критерии приостановления и возобновления тестирования (Suspension criteria and resumption requirements);
9. Результаты тестирования (Test deliverables);
10. Задачи тестирования (Testing tasks);
11. Ресурсы системы (Environmental needs);
12. Обязанности (Responsibilities);
13. Роли и ответственность (Staffing and training needs);
14. Расписание (Schedule);
15. Оценка рисков (Risks and contingencies);
16. Согласования (Approvals).

Чек-лист (check list) — это документ, который описывает что должно быть протестировано. Чек-лист может быть абсолютно разного уровня детализации.

Чаще всего чек-лист содержит только действия, без ожидаемого результата. Чек-лист менее формализован.

Тестовый сценарий (test case) — это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Атрибуты тест кейса:

- **Предусловия (PreConditions)** — список действий, которые приводят систему к состоянию пригодному для проведения основной проверки. Либо список условий, выполнение которых говорит о том, что система находится в пригодном для проведения основного теста состоянии.
- **Шаги (Steps)** — список действий, переводящих систему из одного состояния в другое, для получения результата, на основании которого можно сделать вывод о удовлетворении реализации, поставленным требованиям.
- **Ожидаемый результат (Expected result)** — что по факту должны получить.

Резюме

Старайтесь понять определения, а не зазубривать. Если хотите узнать больше про тестирование, то можете почитать [Библию QA](#). А если возникнет вопрос, всегда можете задать его нам в телеграм-канале [@qa_chillout](#).

Теги:

- [теория](#)
- [теория тестирования](#)
- [основные определения тестирования](#)

Хабы:

- [Тестирование IT-систем](#)
- [Тестирование веб-сервисов](#)
- [Тестирование мобильных приложений](#)
- [Тестирование игр](#)

Типы тестирования

White/Black/Grey Box-тестирование

Для того, чтобы лучше понимать подходы к тестированию программного обеспечения, нужно, конечно же, знать, какие виды и типы тестирования в принципе бывают. Давайте начнем с рассмотрения основных типов тестирования, которые определяют высокоуровневую классификацию тестов.

Самым высоким уровнем в иерархии подходов к тестированию будет понятие **типа**, которое может охватывать сразу несколько смежных техник тестирования. То есть, **одному типу тестирования может соответствовать несколько его видов**. Рассмотрим, для начала, несколько типов тестирования, которые отличаются знанием внутреннего устройства объекта тестирования.

Black Box

Summary: Мы не знаем, как устроена тестируемая система.

Тестирование методом «черного ящика», также известное как тестирование, основанное на спецификации или тестирование поведения – техника тестирования,

основанная на работе исключительно с внешними интерфейсами тестируемой системы.

Согласно ISTQB, **тестирование черного ящика – это:**

- тестирование, как функциональное, так и нефункциональное, не предполагающее знания внутреннего устройства компонента или системы;
- тест-дизайн, основанный на технике черного ящика – процедура написания или выбора тест-кейсов на основе анализа функциональной или нефункциональной спецификации компонента или системы без знания ее внутреннего устройства.

Почему именно «черный ящик»? Тестируемая программа для тестировщика – как черный непрозрачный ящик, содержания которого он не видит. Целью этой техники является поиск ошибок в таких категориях:

- неправильно реализованные или недостающие функции;
- ошибки интерфейса;
- ошибки в структурах данных или организации доступа к внешним базам данных;
- ошибки поведения или недостаточная производительности системы;

Таким образом, мы не имеем представления о структуре и внутреннем устройстве системы. Нужно концентрироваться на том, что программа делает, а не на том, как она это делает.

Пример:

Тестировщик проводит тестирование веб-сайта, не зная особенностей его реализации, используя только предусмотренные разработчиком поля ввода и кнопки. Источник ожидаемого результата – спецификация.

Поскольку это тип тестирования, то он может включать и другие его виды. Тестирование черного ящика может быть как функциональным, так и нефункциональным. Функциональное тестирование предполагает проверку работы функций системы, а нефункциональное – общие характеристики нашей программы.

Техника черного ящика применима на всех уровнях тестирования (от модульного до приемочного), для которых существует спецификация. Например, при осуществлении системного или интеграционного тестирования, требования или функциональная спецификация будут основой для написания тест-кейсов.

Техники тест-дизайна, основанные на использовании черного ящика, включают:

- классы эквивалентности;
- анализ граничных значений;
- таблицы решений;
- диаграммы изменения состояния;
- тестирование всех пар.

Преимущества:

1. тестирование производится с позиции конечного пользователя и может помочь обнаружить неточности и противоречия в спецификации;

2. тестировщику нет необходимости знать языки программирования и углубляться в особенности реализации программы;

3. тестирование может производиться специалистами, независимыми от отдела разработки, что помогает избежать предвзятого отношения;

4. можно начинать писать тест-кейсы, как только готова спецификация.

Недостатки:

1. тестируется только очень ограниченное количество путей выполнения программы;

2. без четкой спецификации (а это скорее реальность на многих проектах) достаточно трудно составить эффективные тест-кейсы;

3. некоторые тесты могут оказаться избыточными, если они уже были проведены разработчиком на уровне модульного тестирования.

Противоположностью техники черного ящика является тестирование методом белого ящика, речь о котором пойдет ниже.

White Box

Summary: Нам известны все детали реализации тестируемой программы.

Тестирование методом белого ящика (также прозрачного, открытого, стеклянного ящика или же основанное на коде или структурное тестирование) – метод тестирования программного обеспечения, который предполагает, что внутренняя структура/устройство/реализация системы известны тестировщику. Мы выбираем входные значения, основываясь на знании кода, который будет их обрабатывать. Точно так же мы знаем, каким должен быть результат этой обработки. Знание всех особенностей тестируемой программы и ее реализации обязательны для этой техники. Тестирование белого ящика – углубление во внутреннее устройство системы за пределы ее внешних интерфейсов.

Согласно ISTQB: **тестирование белого ящика – это:**

- тестирование, основанное на анализе внутренней структуры компонента или системы;
- тест-дизайн, основанный на технике белого ящика – процедура написания или выбора тест-кейсов на основе анализа внутреннего устройства системы или компонента.

Почему «белый ящик»? Тестируемая программа для тестировщика – прозрачный ящик, содержимое которого он прекрасно видит.

Пример:

Тестировщик, который, как правило, является программистом, изучает реализацию кода поля ввода на веб-странице, определяет все предусмотренные (как правильные, так и неправильные) и не предусмотренные пользовательские вводы и сравнивает фактический результат выполнения программы с ожидаемым. При этом ожидаемый результат определяется именно тем, как должен работать код программы.

Тестирование методом белого ящика похоже на работу механика, который изучает двигатель машины, чтобы понять, почему она не заводится.

Техника белого ящика применима на разных уровнях тестирования: от модульного до системного, но, главным образом, применяется именно для реализации модульного тестирования компонента его автором.

Преимущества:

1. тестирование может производиться на ранних этапах: нет необходимости ждать создания пользовательского интерфейса;
2. можно провести более тщательное тестирование с покрытием большого количества путей выполнения программы.

Недостатки:

1. для выполнения тестирования белого ящика необходимо большое количество специальных знаний;
2. при использовании автоматизации тестирования на этом уровне поддержка тестовых скриптов может оказаться достаточно накладной, если программа часто изменяется.

Сравнение Black Box и White Box

Grey Box

Summary: Нам известны только некоторые особенности реализации тестируемой системы.

Тестирование методом серого ящика – метод тестирования программного обеспечения, который предполагает комбинацию White Box и Black Box подходов. То есть внутреннее устройство программы нам известно лишь частично. Предполагается, например, доступ ко внутренней структуре и алгоритмам работы ПО для написания максимально эффективных тест-кейсов, но само тестирование проводится с помощью техники черного ящика, то есть с позиции пользователя.

Эту технику тестирования также называют методом полупрозрачного ящика: что-то мы видим, а что-то – нет.

Пример:

Тестировщик изучает код программы с тем, чтобы лучше понимать принципы ее работы и изучить возможные пути ее выполнения. Такое знание поможет написать тест-кейс, который наверняка будет проверять определенную функциональность.

Техника серого ящика применима на разных уровнях тестирования: от модульного до системного, но, главным образом, применяется на интеграционном уровне для проверки взаимодействия разных модулей программы.

Статическое и динамическое тестирование

По критерию запуска программы (исполняется ли программный код) выделяют еще два типа тестирования: статическое и динамическое.

1. Статическое тестирование

Статистическое тестирование – тип тестирования, который предполагает, что программный код во время тестирования не будет выполняться. При этом, само тестирование может быть как ручным, так и автоматизированным.

Статическое тестирование начинается на ранних этапах жизненного цикла ПО и является, соответственно, частью процесса верификации. Для этого типа

тестирования в некоторых случаях даже не нужен компьютер, например, при проверке требований.

Большинство статических техник могут быть использованы для «тестирования» любых форм документации, включая вычитку кода, инспекцию проектной документации, функциональной спецификации и требований.

Даже статическое тестирование может быть автоматизировано, например, можно использовать автоматические средства проверки синтаксиса программного кода.

Виды статического тестирования:

- вычитка исходного кода программы;
- проверка требований.

2. Динамическое тестирование

Динамическое тестирование – тип тестирования, который предполагает запуск программного кода. Таким образом, анализируется поведение программы во время ее работы.

Для выполнения динамического тестирования необходимо, чтобы тестируемый программный код был написан, скомпилирован и запущен. При этом, может выполняться проверка внешних параметров работы программы: загрузка процессора, использование памяти, время отклика и т.д., то есть ее производительность.

Динамическое тестирование является частью процесса валидации программного обеспечения.

Кроме того, динамическое тестирование может включать разные подвиды, каждый из которых зависит от:

- Доступа к коду (тестирование черным, белым и серым ящиками).
- Уровня тестирования (модульное интеграционное, системное и приемочное тестирование).
- Сферы использования приложения (функциональное, нагрузочное, тестирование безопасности и пр.).

Ручное и автоматизированное

При **ручном тестировании (manual testing)** тестировщики вручную выполняют тесты, не используя никаких средств автоматизации. Ручное тестирование – самый низкоуровневый и простой тип тестирования, не требующий большого количества дополнительных знаний.

Тем не менее, перед тем, как автоматизировать тестирование любого приложения, необходимо сначала выполнить серию тестов вручную. Мануальное тестирование требует значительных усилий, но без него мы не сможем убедиться в том, возможна ли автоматизация в принципе. Один из фундаментальных принципов тестирования гласит: 100% автоматизация невозможна. Поэтому, ручное тестирование – необходимость.

Мифы о ручном тестировании:

- Кто угодно может провести ручное тестирование. Нет, выполнение любого вида тестирования требует специальных знаний и профессиональной подготовки.

- Автоматизированное тестирование мощнее ручного.

- Полная автоматизация невозможна. Необходимо использовать также и ручное тестирование.

- Ручное тестирование – это просто.

Тестирование может быть очень непростым занятием. Проведение тестирования для проверки максимально возможного количества путей выполнения, с использованием минимального числа тест-кейсов, требует серьезных аналитических навыков.

Автоматизированное тестирование (automation testing) предполагает использование специального программного обеспечения (помимо тестируемого) для контроля выполнения тестов и сравнения ожидаемого фактического результата работы программы. Этот тип тестирования помогает автоматизировать часто повторяющиеся, но необходимые для максимизации тестового покрытия, задачи.

Некоторые задачи тестирования, такие как низкоуровневое регрессионное тестирование, могут быть трудозатратными и требующими много времени, если выполнять их вручную. Кроме того, мануальное тестирование может недостаточно эффективно находить некоторые классы ошибок. В таких случаях автоматизация может помочь сэкономить время и усилия проектной команды.

После создания автоматизированных тестов, их можно в любой момент запустить снова, причем, запускаются и выполняются они быстро и точно. Таким образом, если есть необходимость частого повторного прогона тестов, значение автоматизации для упрощения сопровождения проекта и снижения его стоимости трудно переоценить. Ведь даже минимальные патчи и изменения кода могут стать причиной появления новых багов.

Существует несколько основных видов автоматизированного тестирования:

- **Автоматизация тестирования кода (Code-driven testing)** – тестирование на уровне программных модулей, классов и библиотек (фактически, автоматические юнит-тесты).

- **Автоматизация тестирования графического пользовательского интерфейса (Graphical user interface testing)** – специальная программа (фреймворк автоматизации тестирования) позволяет генерировать пользовательские события– нажатия клавиш, клики мышкой, и отслеживание реакции программы на эти действия: соответствует ли она спецификации.

- **Автоматизация тестирования API (Application Programming Interface)** – тестирование программного интерфейса программы. Тестируются интерфейсы, предназначенные для взаимодействия, например, с другими программами или с пользователем. Здесь, опять же, как правило, используются специальные фреймворки.

Для составления автоматизированных тестов QA-специалист должен уметь программировать. Автоматические тесты – это полноценные программы, просто предназначенные для тестирования.

Когда, что и как автоматизировать и автоматизировать ли вообще – очень важные вопросы, ответы на которые должна дать команда разработки. Выбор правильных элементов программы для автоматизации в большой степени будет определять успех автоматизации тестирования в принципе. Нужно избегать автоматизации тестирования участков кода, которые могут часто меняться.

Сравнение ручного и автоматизированного тестирования

Как ручное, так и автоматизированное тестирования могут использоваться на разных уровнях тестирования, а также быть частью других типов и видов тестирования.

Автоматизация сохраняет время, силы и деньги. Автоматизированный тест можно запускать снова и снова, прилагая минимум усилий.

Вручную можно протестировать практически любое приложение, в то время как автоматизировать стоит только стабильные системы. Автоматизированное тестирование используется, главным образом, для регрессии. Кроме того, некоторые виды тестирования, например, ad-hoc или исследовательское тестирование могут быть выполнены только вручную.

Мануальное тестирование может быть повторяющимся и скучным. В то же время, автоматизация может помочь этого избежать – за вас все сделает компьютер.

Таким образом, на реальных проектах зачастую используется комбинация ручного и автоматизированного тестирования, причем уровень автоматизации будет зависеть как от типа проекта, так и от особенностей постановки производственных процессов в компании.

Виды тестирования

Все виды тестирования программного обеспечения, в зависимости от преследуемых целей, можно условно разделить на следующие группы:

1. Функциональные.
2. Нефункциональные.
3. Связанные с изменениями.

Далее мы постараемся более подробно рассказать о каждом отдельном виде тестирования, его назначении и использовании при тестировании программного обеспечения.

Функциональные виды тестирования

Функциональные тесты базируются на функциях и особенностях, а также на взаимодействии с другими системами и могут быть представлены на всех уровнях тестирования: компонентном или модульном (Component/Unit testing), интеграционном (Integration testing), системном (System testing), приемочном (Acceptance testing). Функциональные виды тестирования рассматривают внешнее поведение системы. Далее перечислены одни из самых распространенных видов функциональных тестов.

Функциональное тестирование рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом.

1. Функциональные тесты основываются на функциях, выполняемых системой, и могут проводиться на всех уровнях тестирования (компонентном,

интеграционном, системном, приемочном). Как правило, эти функции описываются в требованиях, функциональных спецификациях или в виде случаев использования системы (**use cases**).

Тестирование функциональности может проводиться в двух аспектах:

- Требования.
- Бизнес-процессы.

Тестирование в аспекте «требования» использует спецификацию функциональных требований к системе, как основу для дизайна тестовых случаев (**Test Cases**). В этом случае необходимо сделать список того, что будет тестироваться, а что нет, приоритезировать требования на основе рисков (если это не сделано в документе с требованиями), а на основе этого приоритезировать тестовые сценарии (**test cases**). Это позволит сфокусироваться и не упустить при тестировании наиболее важный функционал.

Тестирование в аспекте «бизнес-процессы» использует знание бизнес-процессов, которые описывают сценарии ежедневного использования системы. В этом аспекте тестовые сценарии (**test scripts**), как правило, основываются на случаях использования системы (**use cases**).

Преимущества функционального тестирования:

- имитирует фактическое использование системы.

Недостатки функционального тестирования:

- возможность упущения логических ошибок в программном обеспечении;
- вероятность избыточного тестирования.

Достаточно распространенной является автоматизация функционального тестирования.

2. Тестирование безопасности (Security and Access Control Testing)

Тестирование безопасности - это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Принципы безопасности программного обеспечения

Общая стратегия безопасности основывается на трех основных принципах:

1. Конфиденциальность.
2. Целостность.
3. Доступность.

Конфиденциальность

Конфиденциальность - это сокрытие определенных ресурсов или информации. Под конфиденциальностью можно понимать ограничение доступа к ресурсу некоторой категории пользователей или, другими словами, при каких условиях пользователь авторизован получить доступ к данному ресурсу.

Целостность

Существует два основных критерия при определении понятия целостности:

1. **Доверие.** Ожидается, что ресурс будет изменен только соответствующим способом определенной группой пользователей.

2. **Повреждение и восстановление.** В случае, когда данные повреждаются или неправильно меняются авторизованным или не авторизованным пользователем, Вы должны определить, насколько важной является процедура восстановления данных.

Доступность

Доступность представляет собой требования о том, что ресурсы должны быть доступны авторизованному пользователю, внутреннему объекту или устройству. Как правило, чем более критичен ресурс, тем выше уровень доступности должен быть.

3. Тестирование взаимодействия или Interoperability Testing

Тестирование взаимодействия (Interoperability Testing) – это функциональное тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя тестирование совместимости (compatibility testing) и интеграционное тестирование (integration testing).

Программное обеспечение с хорошими характеристиками взаимодействия может быть легко интегрировано с другими системами, не требуя каких-либо серьезных модификаций. В этом случае, количество изменений и время, требуемое на их выполнение, могут быть использованы для измерения возможности взаимодействия.

Нефункциональные виды тестирования

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. В целом, это тестирование того, как система работает.

1. Все виды тестирования производительности

Тестирование производительности (Performance testing).

Задачей тестирования производительности является определение масштабируемости приложения под нагрузкой, при этом происходит:

- Измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций.
- Определение количества пользователей, одновременно работающих с приложением.
- Определение границ приемлемой производительности при увеличении нагрузки (при увеличении интенсивности выполнения этих операций).
- Исследование производительности на высоких, предельных, стрессовых нагрузках.

Стрессовое тестирование (Stress Testing)

Стрессовое тестирование позволяет проверить, насколько приложение и система в целом работоспособны в условиях стресса, а также оценить способность системы к регенерации, т.е. к возвращению к нормальному состоянию, после прекращения воздействия стресса. Стрессом, в данном контексте, может быть повышение интенсивности выполнения операций до очень высоких значений или аварийное изменение конфигурации сервера. Также, одной из задач при стрессовом

тестировании может быть оценка деградации производительности. Таким образом, цели стрессового тестирования могут пересекаться с целями тестирования производительности.

Объемное тестирование (Volume Testing)

Задачей объемного тестирования является получение оценки производительности при увеличении объемов данных в базе данных приложения, при этом происходит:

- Измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций.
- Может производиться определение количества пользователей, одновременно работающих с приложением.

Тестирование стабильности или надежности(Stability / Reliability Testing)

Задачей тестирования стабильности (надежности) является проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки. Время выполнения операций может играть в данном виде тестирования второстепенную роль. При этом на первое место выходит отсутствие утечек памяти, перезапусков серверов под нагрузкой и другие аспекты влияющие именно на стабильность работы.

В англоязычной терминологии вы можете так же найти еще один вид тестирования - Load Testing - тестирование реакции системы на изменение нагрузки (в пределах допустимого). Нам показалось, что Load и Performance преследуют все же одну и ту же цель: проверка производительности (времени отклика) на разных нагрузках. Собственно поэтому мы и не стали разделять их. В то же время кто то может разделить. Главное все таки понимать цели того или иного вида тестирования и постараться их достигнуть.

2. Тестирование Установки (Installation Testing)

Тестирование установки направленно на проверку успешной инсталляции и настройки, а также на обновление или удаление программного обеспечения.

В настоящий момент, наиболее распространена установка ПО при помощи инсталляторов (специальных программ, которые сами по себе так же требуют надлежащего тестирования, описание которого рассмотрено в разделе "Особенности тестирования инсталляторов").

В реальных условиях инсталляторов может не быть. В этом случае придется самостоятельно выполнять установку программного обеспечения, используя документацию в виде инструкций или "read me" файлов, шаг за шагом описывающих все необходимые действия и проверки.

В распределенных системах, где приложение разворачивается на уже работающем окружении, простого набора инструкций может быть мало. Для этого часто пишется план установки (Deployment Plan), включающий не только шаги по инсталляции приложения, но и шаги отката (roll-back) к предыдущей версии (в случае неудачи). Сам по себе план установки также должен пройти процедуру тестирования для избежания проблем при выдаче в реальную эксплуатацию. Особенно это актуально, если установка выполняется на системы, где каждая минута простоя - это потеря репутации и большого количества средств, например: банки, финансовые

компании или даже баннерные сети. Поэтому тестирование установки можно назвать одной из важнейших задач по обеспечению качества программного обеспечения.

3. Тестирование удобства пользования (Usability Testing)

Иногда мы сталкиваемся с непонятными или нелогичными приложениями, многие функции и способы использования которых часто не очевидны. После такой работы редко возникает желание использовать приложение снова, и мы ищем более удобные аналоги. Для того, чтобы приложение было популярным, ему мало быть функциональным – оно должно быть еще и удобным. Если задуматься, интуитивно понятные приложения экономят нервы пользователям и затраты работодателя на обучение. Значит, они более конкурентоспособны! Поэтому тестирование удобства использования, о котором пойдет речь далее, является неотъемлемой частью тестирования любых массовых продуктов.

Тестирование удобства пользования - это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.

Тестирование удобства пользования дает оценку уровня удобства использования приложения по следующим пунктам:

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

Уровни проведения

Проверка удобства использования может проводиться как по отношению к готовому продукту, посредством тестирования черного ящика (black box testing), так и к интерфейсам приложения (API), используемым при разработке - тестирование белого ящика (white box testing). В этом случае проверяется удобство использования внутренних объектов, классов, методов и переменных, а также рассматривается удобство изменения, расширения системы и интеграции ее с другими модулями или системами. Использование удобных интерфейсов (API) может улучшить качество, увеличить скорость написания и поддержки разрабатываемого кода и, как следствие, улучшить качество продукта в целом.

Отсюда становится очевидно, что тестирование удобства пользования может производиться на разных уровнях разработки программного обеспечения: модульном,

интеграционном, системном, приемочном. При этом, оно целиком и полностью будет зависеть от того, кто будет использовать приложение на выделенном конкретном уровне - разработчик, бизнес-пользователь системы и т.д.

Советы по улучшению удобства пользования:

- Для дизайна удобных приложений полезно следовать принципам «пока-йока» или fail-safe. У нас это более известно как «защита от дурака». Простой пример: если поле требует цифровое значение, то логично ограничить пользователю диапазон ввода только цифрами – будет меньше случайных ошибок.

- Для повышения юзабилити существующих приложений можно использовать цикл Демминга (Plan-Do-Check-Act), собирая отзывы о работе и дизайне приложения у существующих пользователей, и, в соответствии с их замечаниями, планируя и проводя улучшения.

Заблуждения о тестировании удобства пользования:

- **Тестирование пользовательского интерфейса = Тестирование удобства пользования**

Тестирование удобства пользования не имеет ничего общего с тестированием функциональности пользовательского интерфейса, оно лишь проводится на пользовательском интерфейсе, равно как и на многих других возможных компонентах продукта. При этом, тип тестирования и тест-кейсы будут совсем другие, так как речь может идти об удобстве использования не визуальных компонентов (если таковые имеются) или процессе администрирования, например, распределенного клиент-серверного продукта и т.д.

- **Тестирование удобства пользования можно провести без участия эксперта**

Не всегда человек не разбирающийся в предметной области способен провести его самостоятельно. Представьте, что тестировщику нужно протестировать удобство пользования стратегического бомбардировщика. Ему придется проверить основные функции: удобство ведения боя, навигации, пилотирования, обслуживания, наземной транспортировки и т.д. Очевидно, что, без привлечения эксперта, это будет весьма проблематично, и, можно даже сказать, невозможно.

4. Тестирование на отказ и восстановление (Failover and Recovery Testing)

Тестирование на отказ и восстановление (Failover and Recovery Testing) проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи (например, отказ сети).

Целью данного вида тестирования является проверка систем восстановления (или дублирующих основной функционал систем), которые, в случае возникновения сбоев, обеспечат сохранность и целостность данных тестируемого продукта.

Тестирование на отказ и восстановление очень важно для систем, работающих по принципу “24x7”. Если Вы создаете продукт, который будет работать, например, в интернете, то без проведения данного вида тестирования Вам просто не обойтись, т.к.

каждая минута простоя или потеря данных, в случае отказа оборудования, может стоить вам денег, потери клиентов и репутации на рынке.

Методика подобного тестирования заключается в симулировании различных условий сбоя и последующем изучении и оценке реакции защитных систем. В процессе подобных проверок выясняется, была ли достигнута требуемая степень восстановления системы после возникновения сбоя.

Для наглядности, рассмотрим некоторые варианты подобного тестирования и общие методы их проведения. Объектом тестирования, в большинстве случаев, являются весьма вероятные эксплуатационные проблемы, такие как:

- Отказ электричества на компьютере-сервере.
- Отказ электричества на компьютере-клиенте.
- Незавершенные циклы обработки данных (прерывание работы фильтров данных, прерывание синхронизации).
- Объявление или внесение в массивы данных невозможных или ошибочных элементов.
- Отказ носителей данных.

Данные ситуации могут быть воспроизведены, как только достигнута некоторая точка в разработке, когда все системы восстановления или дублирования готовы выполнять свои функции. Технически реализовать тесты можно следующими путями:

- Симулировать внезапный отказ электричества на компьютере (обесточить компьютер).
- Симулировать потерю связи с сетью (выключить сетевой кабель, обесточить сетевое устройство).
- Симулировать отказ носителей (обесточить внешний носитель данных).
- Симулировать ситуацию наличия в системе неверных данных (специальный тестовый набор или база данных).

При достижении соответствующих условий сбоя и по результатам работы систем восстановления, можно оценить продукт с точки зрения тестирования на отказ. Во всех вышеперечисленных случаях, по завершении процедур восстановления, должно быть достигнуто определенное требуемое состояние данных продукта:

- Потеря или порча данных в допустимых пределах.
- Отчет или система отчетов, с указанием процессов или транзакций, которые не были завершены в результате сбоя.

Стоит заметить, что тестирование на отказ и восстановление – это весьма специфичное тестирование. Разработка тестовых сценариев должна производиться с учетом всех особенностей тестируемой системы. Принимая во внимание довольно жесткие методы воздействия, стоит также оценить целесообразность проведения данного вида тестирования для конкретного программного продукта.

5. Конфигурационное тестирование (Configuration Testing)

Конфигурационное тестирование(Configuration Testing) — специальный вид тестирования, направленный на проверку работы программного обеспечения при

различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.)

В зависимости от типа проекта конфигурационное тестирование может иметь разные цели:

- Проект по профилированию работы системы.
Цель Тестирования: определить оптимальную конфигурацию оборудования, обеспечивающую требуемые характеристики производительности и времени реакции тестируемой системы.
- Проект по миграции системы с одной платформы на другую.
Цель Тестирования: Проверить объект тестирования на совместимость с объявленным в спецификации оборудованием, операционными системами и программными продуктами третьих фирм.

Примечание: В ISTQB Syllabus вообще не говорится о таком виде тестирования, как конфигурационное. Согласно глоссарию, данный вид тестирования рассматривается там как тестирование портируемости (portability testing: The process of testing to determine the portability of a software product.).

Связанные с изменениями виды тестирования

После проведения необходимых изменений, таких как исправление багов/дефектов, программное обеспечение должно быть перетестировано для подтверждения того факта, что проблема была действительно решена. Ниже перечислены виды тестирования, которые необходимо проводить после установки программного обеспечения, для подтверждения работоспособности приложения или правильности осуществленного исправления дефекта:

1. Дымовое тестирование (Smoke Testing)

Понятие дымовое тестирование пошло из инженерной среды:

"При вводе в эксплуатацию нового оборудования ("железа") считалось, что тестирование прошло удачно, если из установки не пошел дым."

В области же программного обеспечения дымовое тестирование рассматривается как короткий цикл тестов, выполняемый для подтверждения того, что, после сборки кода (нового или исправленного), устанавливаемое приложение стартует и выполняет основные функции.

Вывод о работоспособности основных функций делается на основании результатов поверхностного тестирования наиболее важных модулей приложения на предмет возможности выполнения требуемых задач и наличия быстронаходимых критических и блокирующих дефектов. В случае отсутствия таковых дефектов дымовое тестирование объявляется пройденным и приложение передается для проведения полного цикла тестирования, в противном случае, дымовое тестирование объявляется проваленным и приложение уходит на доработку.

Аналогами дымового тестирования являются Build Verification Testing и Acceptance Testing, выполняемые на функциональном уровне командой тестирования, по результатам которых делается вывод о том, принимается или нет установленная версия программного обеспечения в тестирование, эксплуатацию или на поставку заказчику.

Для облегчения работы, экономии времени и людских ресурсов рекомендуется внедрить автоматизацию тестовых сценариев для дымового тестирования.

2. Регрессионное тестирование (Regression Testing)

Регрессионное тестирование - это вид тестирования, направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб-сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде. Регрессионными могут быть как функциональные, так и нефункциональные тесты.

Как правило, для регрессионного тестирования используются тест-кейсы, написанные на ранних стадиях разработки и тестирования. Это дает гарантию того, что изменения в новой версии приложения не повредили уже существующую функциональность. Рекомендуется делать автоматизацию регрессионных тестов для ускорения последующего процесса тестирования и обнаружения дефектов на ранних стадиях разработки программного обеспечения.

Сам по себе термин "регрессионное тестирование", в зависимости от контекста использования, может иметь разный смысл. Сэм Канер, к примеру, описал 3 основных типа регрессионного тестирования:

- **Регрессия багов (Bug regression)** - попытка доказать, что исправленная ошибка на самом деле не исправлена.
- **Регрессия старых багов (Old bugs regression)** - попытка доказать, что недавнее изменение кода или данных сломало исправление старых ошибок, т.е. старые баги стали снова воспроизводиться.
- **Регрессия побочного эффекта (Side effect regression)** - попытка доказать, что недавнее изменение кода или данных сломало другие части разрабатываемого приложения.

3. Тестирование сборки (Build Verification Test)

Тестирование, направленное на определение соответствия выпущенной версии критериям качества для начала тестирования. По своим целям является аналогом дымового тестирования, направленного на приемку новой версии в дальнейшее тестирование или эксплуатацию. Вглубь оно может проникать дальше, в зависимости от требований к качеству выпущенной версии.

4. Санитарное тестирование или проверка согласованности/исправности (Sanity Testing)

Санитарное тестирование - это узконаправленное тестирование, достаточное для доказательства того, что конкретная функция работает согласно заявленным в спецификации требованиям. Является подмножеством регрессионного тестирования. Используется для определения работоспособности определенной части приложения после изменений произведенных в ней или окружающей среде. Обычно выполняется вручную.

Отличие санитарного тестирования от дымового (Sanity vs Smoke testing)

В некоторых источниках ошибочно полагают, что санитарное и дымовое тестирование - это одно и то же. Мы же полагаем, что эти виды тестирования имеют "векторы движения"- направления в разные стороны. В отличие от дымового (Smoke

testing), санитарное тестирование (Sanity testing) направлено вглубь проверяемой функции, в то время как дымовое - направлено вширь, для покрытия тестами как можно большего функционала в кратчайшие сроки.

Уровни тестирования программного обеспечения

Тестирование на разных уровнях производится на протяжении всего жизненного цикла разработки и сопровождения программного обеспечения. Уровень тестирования определяет то, над чем производятся тесты: над отдельным модулем, группой модулей или системой, в целом. Проведение тестирования на всех уровнях системы - это залог успешной реализации и сдачи проекта.

Уровни Тестирования

1. Компонентное или Модульное тестирование (Component or Unit Testing)

Компонентное (модульное) тестирование проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по-отдельности (модули программ, объекты, классы, функции и т.д.). Обычно компонентное (модульное) тестирование проводится вызывая код, который необходимо проверить и при поддержке сред разработки, таких как фреймворки (frameworks - каркасы) для модульного тестирования или инструменты для отладки. Все найденные дефекты, как правило исправляются в коде без формального их описания в системе менеджмента багов (Bug Tracking System).

Один из наиболее эффективных подходов к компонентному (модульному) тестированию - это подготовка автоматизированных тестов до начала основного кодирования (разработки) программного обеспечения. Это называется разработка от тестирования (test-driven development) или подход тестирования вначале (test first approach). При этом подходе создаются и интегрируются небольшие куски кода, напротив которых запускаются тесты, написанные до начала кодирования. Разработка ведется до тех пор, пока все тесты не будут успешно пройдены.

Разница между компонентным и модульным тестированием:

По-существу, эти уровни тестирования представляют одно и тоже и разница лишь в том, что в компонентном тестировании, в качестве параметров функций, используют реальные объекты и драйверы, а в модульном тестировании - конкретные значения.

2. Интеграционное тестирование (Integration Testing)

Интеграционное тестирование предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами).

Уровни интеграционного тестирования:

- **Компонентный интеграционный уровень** (Component Integration testing) проверяется взаимодействие между компонентами системы после проведения компонентного тестирования.

- **Системный интеграционный уровень** (System Integration Testing) - проверяется взаимодействие между разными системами после проведения системного тестирования.

Подходы к интеграционному тестированию:

- **Снизу вверх (Bottom Up Integration):**

Все низкоуровневые модули, процедуры или функции собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования. Данный подход считается полезным, если все или практически все модули, разрабатываемого уровня, готовы. Также данный подход помогает определить по результатам тестирования уровень готовности приложения.

- **Сверху вниз (Top Down Integration):**

Сначала тестируются все высокоуровневые модули, затем постепенно, один за другим, добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками с аналогичной функциональностью, затем, по мере готовности, они заменяются реальными активными компонентами. Таким образом, мы проводим тестирование сверху вниз.

- **Большой взрыв ("Big Bang" Integration):**

Все (или практически все) разработанные модули собираются вместе в виде законченной системы или ее основной части, а затем проводится интеграционное тестирование. Такой подход очень хорош для сохранения времени. Однако, если тест-кейсы и их результаты записаны неверно, то сам процесс интеграции будет осложнен, что станет преградой для команды тестирования при достижении основной цели интеграционного тестирования.

3. Системное тестирование (System Testing)

Основной задачей системного тестирования является проверка как функциональных, так и не функциональных требований, дефекты в системе в целом. При этом выявляется неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д. Для минимизации рисков, связанных с особенностями поведения в системы в той или иной среде, во время тестирования рекомендуется использовать окружение, максимально приближенное к тому, на которое будет установлен продукт после выдачи.

Можно выделить два подхода к системному тестированию:

- **на базе требований (requirements based)** - для каждого требования пишутся тестовые случаи (test cases), проверяющие выполнение данного требования.

- **на базе случаев использования (use case based)** - на основе представления о способах использования продукта создаются случаи использования системы (Use Cases). По конкретному случаю использования можно определить один или более сценариев. На проверку каждого сценария пишутся тест-кейсы (test cases), которые должны быть протестированы.

4. Приемочное тестирование или прямо-сдаточное испытание (Acceptance Testing)

Приемочное тестирование или прямо-сдаточное испытание (Acceptance Testing) - формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:

- определения удовлетворения системой приемочным критериям;

- вынесения решения заказчиком или другим уполномоченным лицом принятия приложения.

Приемочное тестирование выполняется на основании набора типичных тестовых случаев и сценариев, разработанных на основании требований к данному приложению.

Решение о проведении приемочного тестирования принимается, когда:

- Продукт достиг необходимого уровня качества.
- Заказчик ознакомлен с Планом Приемочных Работ (Product Acceptance Plan) или иным документом, где описан набор действий, связанных с проведением приемочного тестирования, дата проведения, ответственные и т.д.

Фаза приемочного тестирования длится до тех пор, пока заказчик не выносит решение об отправлении приложения на доработку или выдаче приложения.

ЛИСТ

изменений рабочей учебной программы по учебной дисциплине
МДК.01.02 Поддержка и тестирование программных модулей
Дополнения и изменения, вносимые в рабочую программу модуля

Основания внесения дополнений и изменений	Раздел РПД, в который вносятся изменения	Содержание вносимых дополнений, изменений
Предложение работодателя		
Предложение составителя программы	Обновление списка использованных источников	
Приобретение, издание литературы, обновление перечня и содержания ЭБС, баз данных		

Составитель: преподаватель

Е.А. Поддубная

Утверждена на заседании предметной (цикловой) комиссии профессиональных дисциплин программирования в компьютерных системах

Протокол № 10 от 24 мая 2024 г.

Председатель предметной (цикловой) комиссии профессиональных дисциплин специальности **09.02.07 «Информационные системы и программирование»** и **09.02.03 «Программирование в компьютерных системах»**

Л.А. Благова

подпись

Заместитель директора по УР филиала

Т.А. Резуненко

Заведующая сектором библиотеки филиала

Л.Г. Соколова

Инженер-электроник (программно-информационное обеспечение образовательной программы)

А.В. Сметанин

Рецензия
на рабочую программу по междисциплинарному комплексу
МДК.01.02 «Поддержка и тестирование модулей программного обеспечения» для
специальности 09.02.07 «Информационные системы и программирование»

Рабочая программа междисциплинарного комплекса разработана на основании Федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.07 «Информационные системы и программирование», учебного плана и примерной программы

Программа состоит из титульного листа, листа согласования, содержания, паспорта программы, структуры и содержания междисциплинарного комплекса, описания применяемых образовательных технологий, условий реализации программы, перечня основной и дополнительной литературы, методических указаний для обучающихся, оценочных средств для контроля и оценки результатов освоения программы и примера дополнительного обеспечения междисциплинарного комплекса.

В паспорте программы междисциплинарного комплекса изложены область применения программы, место междисциплинарного комплекса в структуре основной профессиональной образовательной программы, цели и задачи междисциплинарного комплекса, количество часов, отведённое на освоение междисциплинарного комплекса.

Структура и содержание программы междисциплинарного комплекса включают в себя объём междисциплинарного комплекса, виды учебной работы, тематический план и содержание учебной дисциплины.

В тематическом плане программы междисциплинарного комплекса отражены наименования разделов и тем, содержание учебного материала, перечислены дидактические единицы, максимальная нагрузка обучающегося, темы практических занятий, самостоятельная работа. Содержание программы рассчитано на 182 часа.

Тематика программы охватывает в достаточном объёме вопросы отладки и тестирования ПО различными способами, а также документирование ПО в соответствии с ЕСКД.

Оформление соответствует всем предъявляемым требованиям. Содержание программы междисциплинарного комплекса включает 8 разделов. Темы и виды самостоятельной работы подобраны с целью закрепления получаемых на аудиторных занятиях навыков. Рабочая программа изложена грамотно, язык и стиль соответствуют общепринятым нормам, профессиональная лексика употребляется правильно.

Программа имеет методическую ценность и может быть использована для обучения в среднем профессиональном образовании.

Рецензент

Директор ООО «Современные
информационные технологии»



А.В.Сметанин

30.05.2024

Рецензия
на рабочую программу по междисциплинарному комплексу
МДК.01.02 «Поддержка и тестирование модулей программного обеспечения» для
специальности 09.02.07 «Информационные системы и программирование»

Рабочая программа междисциплинарного комплекса разработана на основании Федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.07 «Информационные системы и программирование», учебного плана и примерной программы

Программа состоит из титульного листа, листа согласования, содержания, паспорта программы, структуры и содержания междисциплинарного комплекса, описания применяемых образовательных технологий, условий реализации программы, перечня основной и дополнительной литературы, методических указаний для обучающихся, оценочных средств для контроля и оценки результатов освоения программы и примера дополнительного обеспечения междисциплинарного комплекса. В паспорте программы междисциплинарного комплекса указаны область применения программы, место междисциплинарного комплекса в структуре ОПОП по специальности «Информационные системы и программирование», цели и задачи МДК.01.02, количество часов, отведённое на освоение междисциплинарного комплекса.

Структура и содержание программы междисциплинарного комплекса включают в себя объём междисциплинарного комплекса, виды учебной работы, тематический план и содержание учебной дисциплины.

В тематическом плане программы МДК указаны наименования разделов и тем, содержание учебного материала, перечислены дидактические единицы, максимальная нагрузка обучающегося, темы практических занятий, самостоятельная работа. Содержание программы рассчитано на 182 часа.

Рабочая программа МДК рассматривает темы, указанные во ФГОС и примерной программе. Рабочая программа МДК рассматривает темы, «Отладка и тестирование программного обеспечения» и «Документирование» программы по ЕСКД.

Оформление рабочей программы соответствует всем предъявляемым требованиям. Рабочая программа междисциплинарного комплекса содержит 8 разделов. Тематика самостоятельной работы и ее формы соответствуют поставленной цели: закреплению получаемых на аудиторных занятиях навыков. Рабочая программа написана грамотно, в соответствии с научным и деловым стилем изложения, а также общепринятыми нормами. Профессиональная лексика употребляется правильно. Программа может быть использована для обучения в среднем профессиональном образовании по указанной специальности.

Рецензент

Заместитель директора по научной работе филиала ФГБОУ ВО «КубГУ» в г.

Геленджике



Л.В. Галицкая, к.т.н.

30.05.2024 г