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
_____ А.А. Евдокимов
«24» 04 2024

Рабочая программа дисциплины

МДК.01.01 «Разработка программных модулей»

09.02.07 «Информационные системы и программирование»

2024

Рабочая программа МДК.01.01 «Разработка программных модулей» 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.01.01 «Разработка программных модулей»

Форма обучения	очная
Учебный год	2024-2025
2,3 курсы	4, 5, 6 семестр
Лекции	174 час.
Практические занятия	150 час.
Консультации	7 час.
Экзамен	12 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:

Директор ООО «Современные информационные технологии» _____ А.В.Сметанин

Заместитель директора по научной работе филиала ФГБОУ ВО «КубГУ» в г. Геленджике _____ Л.В. Галицкая, к.т.н.

ЛИСТ

согласования рабочей учебной программы по дисциплине

МДК.01.01 «Разработка программных модулей»

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
МДК.01.02 «Поддержка и тестирование программных модулей»

1.1. Область применения программы

Рабочая программа **МДК.01.01 «Разработка 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 дисциплинаМДК.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 - 343 часа, в том числе

- обязательная аудиторная нагрузка - 324часа;
- самостоятельная работа обучающегося - не предусмотрено
- консультации - 7 часа;
- экзамен - 12 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	**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)
		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
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 профессио-нальное 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, Руковод-ством, клиентav и	**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 государствен-ном языке с учетом особенностей социального и	**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.

	культурного контекста.	
ОК 09	Использовать информацион-ные технологии в профессио-нальной 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 1.1-1.5, 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 1.1	Выполнять отладку программных модуле с использованием специализированных программных средств
ПК 1.2	Разрабатывать программные модули в соответствии с техническим заданием
ПК 1.3	Выполнять отладку программных модулей с использованием специализированных программных средств
ПК 1.4	Выполнять тестирование программных модулей
ПК 1.5	Осуществлять рефакторинг и оптимизацию программного кода

В результате освоения МДК 01.01 студент должен:

Иметь практический опыт	Разрабатывать алгоритм решения поставленной задачи и реализовывать его средствами автоматизированного проектирования. Разрабатывать код программного продукта на основе готовой спецификации на уровне модуля. Использовать инструментальные средства на этапе отладки программного продукта. Проводить тестирование программного модуля по определенному сценарию Проводить тестирование программного модуля по определенному сценарию. Использовать инструментальные средства на этапе тестирования программного продукта Анализировать алгоритмы, в том числе с применением инструментальных средств. Осуществлять рефакторинг и оптимизацию программного кода.

уметь	Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием. Оформлять документацию на программные средства. Оценка сложности алгоритма. Создавать программу по разработанному алгоритму как отдельный модуль. Оформлять документацию на программные средства. Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней в том числе для мобильных платформ Выполнять отладку и тестирование программы на уровне модуля. Оформлять документацию на программные средства. Применять инструментальные средства отладки программного обеспечения Выполнять отладку и тестирование программы на уровне модуля. Оформлять документацию на программные средства Анализировать алгоритмы, в том числе с применением инструментальных средств. Осуществлять рефакторинг и оптимизацию программного кода
знать	Основные этапы разработки программного обеспечения. Основные принципы технологии структурного и объектно-ориентированного программирования Основные этапы разработки программного обеспечения. Основные принципы технологии структурного и объектноориентированного программирования.. Знание API современных мо льных операционных систем Основные принципы отладки и тестирования программных продуктов. Инструментарий отладки программных продуктов Основные виды и принципы тестирования программных продуктов. Способы оптимизации и приемы рефакторинга. Инструментальные средства анализа алгоритма. Методы организации рефакторинга и оптимизации кода. Принципы работы с системой контроля версий

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 и практики разработки программных модулей, сред разработки, отладки и тестирования и закрепления их на опыте.

2. СТРУКТУРА И СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ

2.1. Объем учебной дисциплины и виды учебной работы

Вид учебной работы	**Объем в часах**
Объем образовательной программы	**343**
в том числе:	
теоретическое обучение (+ текущая аттестация – зачёт)	174
практические занятия	150
консультации	7
Промежуточная аттестация (экзамен)	12

2.2. Структура дисциплины

Наименование разделов и тем	**Всего**	**Количество аудиторных часов**		**Самостоятельная работа обучающегося (час)**
		Теоретическое обучение	**Практические и лабораторные занятия**	
Раздел1. **Отладка и тестирование программного обеспечения**	**324**	**174**	**150**	-
Тема 1.1. Жизненный цикл ПО	2	2	-	-
Тема 1.2. Структурное программиро-вание	50	24	26	-
Тема 1.3. Объектно-ориентированное программирование	104	54	50	-
Тема 1.4. Паттерны проектирования	22	16	6	-
Тема 1.5. Событийноуправляемое програм-мирование	80	30	50	-
Тема 1.6. Оптимизация и рефакторинг кода	8	6	2	-
Тема 1.7. Разработка пользовательского интерфейса	8	6	2	-
Тема 1.8. Основы ADO.Net	**50**	36	14	-
Всего по дисциплине	**324**	**174**	**150**	-

2.3. Тематический план и содержание учебной дисциплины МДК.01.01 «Разработка программных модулей»

2.2. Тематический план и содержание профессионального модуля (ПМ.01)

Наименование разделов и тем профессионального модуля	**Содержание учебного материала, лабораторные работы и практические занятия, самостоятельная учебная работа обучающихся, курсовая работа (проект)**	Уровень освоения	**ПК**
1	**2**		
Раздел 1. Разработка программных модулей			
МДК.01.01 Разработка программных модулей			
Тема 1.1. Жизненный цикл ПО	**Содержание**		
	Л1. Понятие ЖЦ ПО. Этапы ЖЦ ПО. Модели ЖЦ ПО: каскадная, инкрементная, спиральная модели [Понятие алгоритма. Виды алгоритмов. Способы записи алгоритмов)	2	1
	В том числе, практических занятий и лабораторных		
Тема 1.2. Структурное программиро-вание	**Содержание**		
	Л2. Технология структурного программирования. Основные этапы решения задач на ЭВМ. Структурное программирование и алгоритмизация. Теорема структуризации. [Система программирования: состав и функции]	2	2
	Л3. Разработка сверху вниз и снизу вверх. Основные принципы технологии структурного программирования. Методы и приемы алгоритмизации поставленных задач [Понятие переменной. Виды данных. Алфавит ЯП C#]	2	3
	Л4. Инструментальные средства оформления и документирования алгоритмов программ. Определение процессов предметной области. Процессы управления проектами. Технология быстрой разработки приложений. [Операторы ЯП C#]	2	4
	Л5. Методология, технология и инструментальные средства разработки прикладного программного обеспечения. RAD и CASE-средства. Задачи и функции инструментального программного обеспечения **[**Простые алгоритмы. Массивы]	2	5
	Л6. Простые методы сортировки. [Алгоритмы решения типовых задач, области и способы их применения.]	2	6
	Л7. Сортировка вставками, выбором, быстрая	2	7
	Л8. Алгоритмы сложной сортировки. Сортировка пузырьком. Сортировка перемешиванием (шейкерная сортировка). Сортировка расчёской.	2	8
	Л9. Алгоритмы решения типовых задач, области и способы их применения	2	9
	Л10. Алгоритмы поиска данных. Последовательный поиск. Индексно-последовательный поиск. Бинарный поиск.	2	10

	Л11. Рекурсивные и эвристические алгоритмы. Понятие рекурсии и рекурсивной функции. Эвристика. Применение эвристического алгоритма. Пример оценки эвристического решения	2	11
	Л12. Алгоритмы решения типовых задач области и способы их применения	2	12
	Л13. Оценка сложности алгоритма. Классификация, классы алгоритмов, неразрешимые задачи. Методы и приемы формализации задач	2	13
	В том числе, практических и лабораторных занятий по отработке умений: - использовать методы и приемы алгоритмизации поставленных задач;		
	Пр 1. Применение циклических алгоритмов	2	14
	Пр 2. Оценка сложности циклических алгоритмов	2	15
	Пр 3. Применение вложенных циклов	2	16
	Пр 4. Оценка сложности вложенных циклов	2	17
	Пр 5. Применение алгоритмов сортировки в программе	2	18
	Пр 6. Оценка сложности алгоритмов сортировки	2	19
	Пр 7. Применение алгоритмов поиска данных в программе	2	20
	Пр 8. Разработка алгоритмов с использованием подпрограмм	2	21
	Пр 9. Оценка сложности алгоритмов поиска	2	22
	Пр 10. Применение рекурсивных алгоритмов в программе	2	23
	Пр 11. Оценка сложности рекурсивных алгоритмов	2	24
	Пр 12. Применение эвристических алгоритмов в программе	2	25
	Пр 13. Оценка сложности эвристических алгоритмов	2	26
Тема 1.3. Объектно-ориентированное программирование	Содержание		
	Л14. Основные этапы и принципы процесса разработки программного обеспечения. Процесс разработки. Цикл разработки.	2	27
	Л 15. Языки программирования и среды разработки, средства пакетного выполнения процедур	2	28
	Л16. Основы программирования на объектно-ориентированном языке C#. Вид программы на языке C#. Простые типы данных. Оператор присваивания. Понятие арифметического выражения. Ввод данных с клавиатуры и вывод на экран. Форматный вывод.	2	29
	Л17. Основные структуры данных. Синтаксис языка программирования C# , особенности программирования на этом языке, стандартные библиотеки языка программирования	2	30
	Л18. Особенности использования данных целого и вещественного типа. Правила согласования типов. Проблема переноса программы с одного языка программирования на другой.	2	31

	Л19. Операторы разветвления алгоритма. Логические выражения. Условный оператор if. Условная операция. Оператор выбора (варианта, переключения) switch. Оператор разрыва break. Оператор безусловного перехода goto.	2	32
	Л20. Операторы циклов. Оператор цикла с параметром (for). Оператор цикла с предусловием (while). Оператор цикла с постусловием (do while). Вспомогательные операторы, управляющие работой цикла: оператор разрыва break и оператор продолжения continue.	2	33
	Л21. Одномерные массивы: определение и описание массива, формирование массивов исходных данных, особенности вывода одномерных массивов. Символьный массив	2	34
	Л22. Двумерные (многомерные) массивы: определение двумерных массивов, особенности формирования двумерных массивов, особенности вывода и обработки двумерных массивов.	2	35
	Л23. Использование подпрограмм. Понятие подпрограммы. Структура процедур и функций. Обращение к подпрограмме. Механизм передачи параметров. Рекурсивные процедуры и функции.	2	36
	Л24. Строковые данные. Функция для работы со строками. Перенос символов между строкой и символьным массивом.	2	37
	Л25. Другие способы организации данных. Коллекции. Списки. Понятие коллекции и списка. Способы определения и описания коллекции и списка.	2	38
	Л26. Структурированные данные (структуры, struct).	2	39
	Л27. Запись и чтение данных из файла. Методы организации файловых систем. Запись и чтение строк. Прямой доступ к файлу.	2	40
	Л28. Понятие классов. Классы: основные понятия. Применение классов.	2	41
	Л29. Принципы ООП: наследование, инкапсуляция, полиморфизм	2	42
	Л30. Перегрузка методов. Понятие метода и сигнатуры. Способы применения перегрузки методов	2	43
	Л31. Операции класса. Понятие операции класса. Секции операции. Параметры и выражения. Кванторы видимости	2	44
	Л32. Иерархия классов. Базовые и производные классы. Модификаторы доступа. Конструкторы и деструкторы. Примеры реализации классов	2	45
	Л33. Синтаксис интерфейсов. Определение интерфейса. Применение интерфейсов. Явная реализация и реализация в базовых и производных классах.	2	46
	Экзамен за 4 семестр		

5 семестр		
Л34. Интерфейсы и наследование. Наследование интерфейсов. Ковариантность и контравариантность обобщенных интерфейсов	2	47
Л35. Структуры. Определение структуры. Конструкторы структуры. Применение структур в программе	2	48
Л36. Делегаты. Определение делегатов. Присвоение ссылки на метод. Соответствие методов делегату. Добавление методов в делегат. Объединение делегатов. Вызов делегата. Делегаты как параметры методов. Обобщенные делегаты. Применение делегатов	2	49
Л37. Регулярные выражения. Классы StringBuilder и String. Параметр RegexOptions. Синтаксис регулярных выражений. Проверка на соответствие строки формату. Замена и метод Replace	2	50
Л38. Коллекции. Параметризованные классы. Понятие коллекции. ArrayList. Список List. Двухсвязный список LinkedList. Очередь Queue. Стек Stack. Словарь Dictionary. Класс ObservableCollection. Интерфейсы Enumerable и Enumerator. Итераторы и оператор yield	2	51
Л39. Указатели. Определение указателя. Операции с указателями. Арифметика указателей. Константы и указатели. Указатели и массивы. Указатели в параметрах функции. Массивы в параметрах функции. Указатели на функции. Указатели на функции как параметры. Указатель на функцию как возвращаемое значение. Динамические объекты и массивы	2	52
Л40. Операции со списками. Класс List. Двухсвязный список. Добавление и удаление элементов. Получение индекса элемента и сортировка списков	2	53
В том числе, практических и лабораторных занятий по отработке умений: - создавать программу по разработанному алгоритму как отдельный модуль; - осуществлять разработку кода программного модуля на современных языках программирования; - использовать выбранную среду программирования и средства системы управления базами данных;		
Пр 14. Изучение интерфейса среды разработки программ	2	54
Пр 15. Особенности визуального компонента Форма	2	55
Пр 16. Особенности использования визуальных и	2	56
Пр 17. Проектирование простого консольного приложения с использованием выбранной среды программирования Разработка простого консольного приложения с использованием выбранной среды программирования	2	57
Пр18. Разработка простого консольного приложения с использованием выбранной среды программирования	2	58
Пр 19. Проектирование консольных приложений и применение операторов ветвления с применением стандартных алгоритмов в соответствующих областях	2	59

	Пр 20. Разработка консольных приложений и применение операторов ветвления с применением стандартных алгоритмов в соответствующих областях	2	60
	Пр 21. Разработка консольных приложений и применение циклических операторов	2	61
	Пр 22. Разработка консольных приложений по работе с одномерными массивами	2	62

	ПЗ 23. Разработка консольных приложений по работе с двумерными массивами	2	63
	ПЗ 24. Разработка консольных приложений и применение подпрограмм	2	64
	ПЗ 25. Работа со строками в консольных приложениях с применением стандартных алгоритмов в соответствующих областях	2	65
	ПЗ 26. Работа с коллекциями, списками и структурами в консольных приложениях	2	66
	ПЗ 27. Запись и чтение данных из файла с помощью консольных приложений	2	67
	ПЗ 28. Работа с классами	2	68
	Пр 29. Работа с методами	2	69
	ПЗ 30. Перегрузка методов	2	70
	ПЗ 31. Определение операций в классе	2	71
	ПЗ 32. Создание наследованных классов	2	72
	ПЗ 33. Работа с объектами через интерфейсы	2	73
	ПЗ 34. Использование стандартных интерфейсов	2	74
	ПЗ 35. Работа с типом данных структура	2	75
	ПЗ 36. Коллекции. Параметризованные классы	2	76
	ПЗ 37. Использование регулярных выражений	2	77
	ПЗ 38. Операции со списками	2	78
Тема 1.4. Паттерны проектирования	Содержание		
	Л41. Назначение и виды паттернов. Основы паттернов проектирования. Введение в паттерны проектирования.	2	79
	Л42. Отношения между классами и объектами. Интерфейсы или абстрактные классы	2	80
	Л43. Основные шаблоны. Взаимодействие между классами или объектами. Идиомы. Шаблон делегирования. Шаблон функционального дизайна. Неизменяемый интерфейс. Интерфейс-маркер. Контейнер свойств. Канал событий	2	81
	Л44. Порождающие шаблоны. Фабричный метод (Factory Method). Абстрактная фабрика (Abstract Factory). Одиночка (Singleton). Прототип (Prototype). Строитель (Builder)	2	82
	Л45. Структурные шаблоны. Декоратор (Decorator). Адаптер (Adapter). Фасад (Facade). Компоновщик (Composite). Заместитель (Прокси). Мост (Bridge). Приспособленец (Flyweight)	2	83
	Л46. Поведенческие шаблоны. Стратегия (Strategy). Наблюдатель (Observer). Команда (Command).	2	84
	Л47. Шаблонный метод (Template Method). Итератор (Iterator). Состояние (State). Цепочка Обязанностей (Chain of responsibility). Интерпретатор (Interpreter). Посредник (Mediator). Хранитель (Memento). Посетитель (Visitor)	2	85

	Л48. Принцип единственной обязанности. Принципы SOLID. Принцип единственной обязанности. Принцип открытости/закрытости. Принцип подстановки Лисков. Принцип разделения интерфейсов. Принцип инверсии зависимостей	2	86
	В том числе, практических и лабораторных занятий по отработке умений: - создавать программу по разработанному алгоритму как отдельный модуль; - осуществлять разработку кода программного модуля на современных языках программирования; - использовать выбранную среду программирования и средства системы управления базами данных;		

	- применять выбранные языки программирования для написания программного кода; - применять стандартные алгоритмы в соответствующих областях; применять языки программирования, определенные в техническом задании на разработку системы управления базами данных, для написания программного кода		
	Пр 39. Использование основных и порождающих шаблонов	2	87
	Пр 40. Использование структурных шаблонов	2	88
	Пр 41. Разработка приложения с использованием шаблонов	2	89
Тема 1.5. Событийно-управляемое програм-мирование	**Содержание**		
	Л49. Событийно-управляемое программирование. Основные характеристики языка. Структура программы. Визуальные компоненты .NET Framework.	2	90
	Л50. Методы и средства сборки модулей и компонент программного обеспечения	2	91
	Л51. Технологии разработки приложений Windows Forms. Создание приложений при помощи визуальных компонентов. Создание формы. Компонент Form как основной контейнер для всех визуальных компонентов.	2	92
	Л52. События и свойства. Основные визуальные компоненты программирования: Form, Button, Label, TextBox, ComboBox.	2	93
	Л53. Элементы управления. Диалоговые окна. Обработчики событий. Присваивание и вывод результатов при помощи компонента Label. Ввод данных и вывод результатов. Условный оператор. Условный оператор с двумя ветвями. Условный оператор с операторными скобками. Вложенные условные операторы.	2	94
	6 семестр		
	Л54. Свойства объекта Форма	2	95
	Л55. Способы создания групп компонентов. Создание группы «радиокнопок» (radioButton) и «чек-кнопок» (checkBox). Компонент TrackBar. Компонент ComboBox. Компонент ListBox. Компонент GroupBox. Перезапись строк из разных компонентов. Изменение свойств текста**.** Оператор переключения (выбора). Выбор, содержащий более одного оператора.	2	96
	Л56. Методы формирования меню. Компонент menuStrip. Использование компонента contextMenuStrip. Оформление меню.	2	97

	Л57. Обработка дополнительных форм. Добавление дополнительных форм Additional в проект. Обработка диалоговых окон. Окна сообщений (MessageBox). Вызов диалоговых форм.	2	98
	Л58. Применение вкладок на форме. Компонент tabControl. Разработка программы с помощью вкладок. Отдельная вкладка как индивидуальный контейнер компонентов.	2	99
	Л59. Компоненты диапазона значений. Компонент trackBar. Компонент progressBar. Настройка и применение компонентов trackBar и progressBar.	2	100
	Л60. Работа с компонентом вывода табличных данных. Вывод	2	101
	Л61.Создание базы данных и привязка для последующего вывода и редактирования БД с помощью компонента dataGridView. Компонент DataSet	2	102
	Л62. Введение в графику. Рисование простых графических объектов. Изменение масштаба. Вращение в плоскости. Рисование трехмерных фигур.	2	103
	Л63. Применение в программировании прочих компонентов. Компонент панели (panel). Применение таймера в визуальном программировании (timer). Применение компонента ListView. Ввод «по маске», в том числе паролей, с использованием MaskedTextBox. Использование календаря MonthCalendar. Встраивание компонента WebBrowser.	2	104

	В том числе, практических и лабораторных занятий по отработке умений:36 - создавать программу по разработанному алгоритму как отдельный модуль; - осуществлять разработку кода программного модуля на современных языках программирования; - использовать выбранную среду программирования и средства системы управления базами данных; - применять выбранные языки программирования для		
	Пр 42. Основы языка ZAML и его применение для среды разработки Visual Studio	2	105
	Пр 43. Изучение свойств базовых инструментов технологии Wpf	2	105
	Пр 44. Создание событий для базовых иняттрументов	2	107
	Пр 45. Проектирование приложения с использованием текстовых компонентов	2	108
	Пр 46. Разработка приложения с использованием текстовых компонентов	2	109
	Пр 47. Проектирование приложения с несколькими формами	2	110
	Пр 48. Разработка приложения с несколькими формами	2	111
	Пр 49. Проектирование приложения с не визуальными компонентами	2	112
	Пр 50. Разработка приложения с не визуальными компонентами	2	113
	Пр 51. Разработка алгоритма игрового приложения	2	114
	Пр 52. Разработка интерфейса игрового приложения	2	115
	Пр 53. Разработка кода игрового приложения	2	116
	Пр 54. Разработка алгоритма, интерфейса и кода приложения с анимацией	2	117

	Пр 55. Создание приложения Windows Forms и организация ввода/вывода	2	118
	Пр 56. Работа с радиокнопками, чек-кнопками и компонентом GroupBox	2	119
	Пр 57. Работа с компонентами ComboBox и ListBox	2	120
	Пр 58. Разработка главного меню программы с помощью компонента MenuStrip	2	121
	Пр 59. Работа с контекстным меню (contextMenuStrip)	2	122
	Пр 60. Работа с диалоговыми формами и вывод сообщений	2	123
	Пр 61. Работа с вкладками tabControl	2	124
	Пр 62. Работа с компонентами trackBar и ProgressBar	2	125
	Пр 63. Работа с базами данных на форме с помощью компонента dataGridView	2	126
	Пр 64. Рисование простых графических объектов на форме	2	127
	Пр 65. Рисование трехмерных объектов на форме	2	128
	Пр 66. Применение в программе прочих визуальных компонентов	2	129
Тема 1.6. Оптимизация и рефакторинг кода	**Содержание**		
	Л64. Методы оптимизации программного кода. Основные принципы оптимизации: естественность, производительность, время. Участки кода, которые не оптимизируются. Настройка окружения. Избавление от ненужного функционала. Мемоизация. Кеширование.	2	130
	Л65. Распараллеливание программ. Способы оптимизации	2	131
	Л66. Цели и методы рефакторинга. Понятие рефакторинга (перепроектирования) кода. Причины применения рефакторинга. Признаки плохого кода. Методы и приемы рефакторинга. Проблемы, возникающие при применении рефакторинга. Средства автоматизации рефакторинга.	2	132
	В том числе, практических и лабораторных занятий по отработке умений:		
	Пр 67. Оптимизация и рефакторинг кода		133
Тема 1.7. Разработка пользова-тельского интерфейса	**Содержание**	**8**	
	Л67. Принципы разработки графического интерфейса пользователя. Понятие интерфейса пользователя. GUI. UI-дизайн. Необходимость разработки UI-дизайна. Разработка UI-интерфейсов.	2	134
	Л68. Проектирование. Создание дизайн-макета. Карта экранов (UFD). Утверждение структуры и согласование стиля	2	135
	Л69. Правила разработки интерфейсов пользователя. Главная задача интерфейса пользователя. Ключевое назначение интерфейсов. Акценты и их расстановка. Взаимодействия пользователя с интерфейсом. Основные принципы и правила разработки пользовательских интерфейсов.	2	136
	В том числе, практических и лабораторных занятий по отработке умений: - осуществлять разработку кода программного модуля на современных языках программирования; - использовать выбранную среду программирования и средства системы управления базами данных.	2	
	Пр №68. Разработка интерфейса пользователя		137
Тема 1.8.	**Содержание**		

Основы ADO.Net	**Л70. Основы работы с MS SQL Server.** Установка MS SQL Server. Установка SQL Server Management Studio. Установка LocalDB.	2	138
	Л71. Начало работы с MS SQL Server. Создание базы данных. Создание таблиц. Первый запрос на T-SQL	2	139
	Л72. Работа с базами данных. Понятие и структура базы данных (БД) Основы интерфейса взаимодействия с базами данных в ADO.NET. Провайдеры данных	2	140
	Л73. Доступ к данным. Создание базы данных. Строка подключения. Создание подключения. Пул подключений. Методы и средства миграции и преобразования данных	2	141
	Л74. Операторы DDL. Создание и удаление базы данных. Создание и удаление таблиц. Типы данных T-SQL. Атрибуты и ограничения столбцов и таблиц. Внешние ключи. Изменение таблицы. Пакеты. Команда GO	2	142
	Л75. Создание таблицы, работа с записями. Команды создания и удаления таблиц. Обновление, добавление, изменение и удаление записей в таблицах БД.	2	143
	Л76. Операторы DML. Добавление данных. Команда INSERT. Выборка данных. Команда SELECT.	2	144
	Л 77. Сортировка. ORDER BY. Извлечение диапазона строк. Фильтрация. WHERE. Операторы фильтрации. Обновление данных. Команда UPDATE. Удаление данных. Команда DELETE	2	145
	Л78. Способы создания команд. Выполнение команд и SqlCommand. Чтение результатов запроса и SqlDataReader. Типизация результатов SqlDataReader. Получение скалярных значений.	2	146
	Л79. Параметризация запросов. Выходные параметры	2	147
	Л80. Хранимые процедуры и транзакции. Работа с хранимыми процедурами. Выходные параметры хранимых процедур.	2	148
	Л 81. Транзакции. Сохранение и извлечение файлов из базы данных	2	149
	Л82. Методы группировки и соединения таблиц. Агрегатные функции. Операторы GROUP BY и HAVING. Расширения SQL Server для группировки.	2	150
	Л83. Неявное соединение таблиц. Inner Join. Outer Join. Группировка в соединениях. UNION. EXCEPT. INTERSECT	2	151
	Л84. Принципы работы с компонентами SqlDataAdapter и DataSet. SqlDataAdapter и DataSet. Постраничный просмотр в SqlDataAdapter. SqlCommandBuilder и сохранение изменений DataSet в базе данных.	2	152
	Л85. Обновление БД из DataSet вручную. Все операции с БД в графическом приложении. DataSet и DataTable. Отношения между таблицами в DataSet. LINQ to DataSet. DataSet и XML	2	153

Л86. Методы преобразования LINQ в SQL. Определение контекста данных и моделей. Операции с данными в LINQ to SQL. Изменение объектов в LINQ to SQL. Добавление в LINQ to SQL. Удаление в LINQ to SQL.	2	154
Л 87. Методы ExecuteCommand и ExecuteQuery. Хранимые процедуры	2	155
В том числе, практических и лабораторных занятий по отработке умений:6 - использовать выбранную среду программирования и средства системы управления базами данных - осуществлять разработку кода программного модуля на языках низкого и высокого уровней.	2	
Пр 69. Разработка алгоритма приложения с БД с использованием выбранных средств системы управления базами данных	2	156
Пр 70. Создание интерфейса и кода приложения с БД	2	157
Пр 71. Создание приложения с БД	2	158
ПР 72. Создание простых запросов к БД	2	159
Пр 73. Создание сложных запросов к БД	2	160
Пр 74. Создание отчётов о состоянии БД	2	161
Пр 75. Создание хранимых процедур	2	162

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			
	Тема1.1. 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, 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. Автоматизи-рованное 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
2**Раздел 2. Документирование**			
	Тема 2.1. Документи-рование 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 документирование 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 программ	Т, У
		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**		

	Тема1.1. 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.Перекрёстная проверка результатов тестирования	ПР, У
		10.Тестирование программы методом «чёрного» ящика.	ПР, У
		11.Перекрёстная проверка результатов тестирования	ПР, У
	Тема 1.5. Отчёты о тестировании	13.Составление проекта отчёта о дефектах	ПР, У
		14.Перекрёстный анализ составленных отчётов	ПР, У
		15. Составление тест-плана	ПР, У
		16. Оценка трудозатрат на планирование и тестирование по составленным планам	ПР, У
	Тема 1.6. Методы отладки и тестирования	17.Проверочная работа по теории методов тестирования	ПР, У
		18.Отладка программы всеми известными методами и составление отчёта о результатах	ПР, У
		19.Тестирование одной программы 9-ю методами и составление отчёта о результатах	ПР, У
	Тема 1.7. Автоматизи-рованное тестирование	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	30.Составление руководств системного администратора и пользователя	ПР, У

	вспомогательных процессов ЖЦ ПО		
	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= sgrt(sgr(x2-x1)+sgr(y2-y1)), где (x1,y1)- координаты одной точки, (x2,y2)-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 Mi=i*Pi , если нечетный, то Mi=-Pi. 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 k1 и k2.
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 компетентностного 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	Тема1.1. 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 техноло-гий	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.01.01 Разработка программных модулей»

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 в 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

Основная литература

1. Белугина, С. В. Разработка программных модулей программного обеспечения для компьютерных систем. Прикладное программирование : учебное пособие для спо / С. В. Белугина. — 3-е изд., стер. — Санкт-Петербург : Лань, 2024. — 312 с. — ISBN 978-5-8114-9817-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/200390 — Режим доступа: для авториз. пользователей.
2. Гагарина, Л.Г.Технология разработки программного обеспечения : учебное пособие для студентов вузов, обучающихся по направлениям подготовки "Информатика и вычислительная техника" / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Виснадул ; под ред. Л. Г. Гагариной. - Москва : ФОРУМ : ИНФРА-М, 2017. - 399 с. : ил. - (Высшее образование). - Библиогр.: с. 388-391. - ISBN 978-5-8199-0342-1. - ISBN 978-5-16-003193-4 : Текст непосредственный (50)
3. Гниденко, И. Г. Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2024. — 235 с. — (Профессиональное образование). — ISBN 978-5-534-05047-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/492496
4. 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)

Дополнительная литература

1. Акопов, А. С. Компьютерное моделирование : учебник и практикум для среднего профессионального образования / А. С. Акопов. — Москва : Издательство Юрайт, 2024. — 389 с. — (Профессиональное образование). — ISBN 978-5-534-10712-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/495518
2. Богатырев, В. А. Информационные системы и технологии. Теория надежности : учебное пособие для вузов / В. А. Богатырев. — Москва : Издательство Юрайт, 2024. — 318 с. — (Высшее образование). — ISBN 978-5-534-00475-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/490026
3. Проектирование информационных систем : учебник и практикум для среднего профессионального образования / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук ; под общей редакцией Д. В. Чистова. — Москва : Издательство Юрайт, 2024. — 258 с. — (Профессиональное образование). — ISBN 978-5-534-03173-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/491568
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/489604

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

Прикладная информатика.- URL: http://elibrary.ru/title_about.asp?id=25599

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 «eLabrary.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/

5. КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ МДК.01.01 «Разработка программных модулей»

5.1. Критерии оценивания практических работ

Раздел модуля 2. Технологии тестирования программных модулей		
ПК 1.3 Выполнять отладку програм-мных модулей с использова-нием специализи-рованных програмных средств	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. Оценка «**хорошо**» - выполнена отладка модуля (**Дополнительно для квалифи-каций "Программист" и "Специалист по тестированию в области информационных технологий": с использовани-ем инструментария среды проектиро- вания)**; сохранены и представлены результаты отладки. Оценка «**удовлетворительно**» - выполнена отладка модуля, пояснены ее результаты.	Экзамен/зачет в фор-ме собеседования: практическое заданиепо выполнению от-ладки предложенного программного модуля Защита отчетов по практическим и лабораторным работам Интерпретация ре-зультатов наблюденийза деятельностью обу-чающегося в процессепрактики

ПК 1.4 Выполнять тести рование программны х модулей	Оценка «**отлично**» - выполнено тестиро- вание модуля, в том числе с помощью ин-струментальных средств, и оформлены результаты тестирования в соответствии со стандартами. **Дополнительно для квалификации "Специалист по тести- рованию в области информационных технологий":** выполнено функциональ- ное тестирование, выполнена и представ- лена оценка тестового покрытия, сделан вывод о достаточности тестового пакета. Оценка «**хорошо**» - выполнено тестирова- ние модуля, в том числе с помощью ин- струментальных средств, и оформлены результаты тестирования. **Дополнительно**	Экзамен/зачет в фор- ме собеседования: практическое задание по выполнению за- данных видов тести- рования программно- го модуля. **Дополнительно для квалификации "Специалист по те- стированию в обла- сти информацион- ных технологий":** оценке тестового по-

	для квалификации "Специалист по те-стированию в области информацион- ных технологий": выполнено функцио- нальное тестирование, выполнена и пред-ставлена оценка тестового покрытия. Оценка «**удовлетворительно**» - выполне-но тестирование модуля и оформлены ре-зультаты тестирования. **Дополнительно для квалификации "Специалист по те- стированию в области информацион- ных технологий":** выполнено функцио-нальное тестирование, выполнена и пред-ставлена оценка тестового покрытия с не-которыми погрешностями.	крытия**.** Защита отчетов по практическим и лабо-раторным работам Интерпретация ре-зультатов наблюденийза деятельностью обу-чающегося в процессепрактики
ПК 1.5 Осуществлять рефакторинг и оптими- зацию программного ко-да	Оценка «**отлично**» - определены каче-ственные характеристики программного кода с помощью инструментальных средств; выявлены фрагменты некаче-ственного кода; выполнен рефакторинг науровнях переменных, функций, классов, алгоритмических структур; проведена оп-тимизация и подтверждено повышение качества программного кода. Оценка «**хорошо**» - определены каче-ственные характеристики программного кода с помощью инструментальных средств; выявлены фрагменты некаче-ственного кода; выполнен рефакторинг нанескольких уровнях; проведена оптимиза-ция и выполнена оценка качества полу- ченного программного кода. Оценка «**удовлетворительно**» - опреде-лены качественные характеристики про-граммного кода частично с помощью ин-струментальных средств; выявлено не- сколько фрагментов некачественного ко-да; выполнен рефакторинг на нескольких уровнях;	Экзамен/зачет в фор- ме собеседования: практическое задание по оценке качества кода предложенного программного модуля,поиску некачествен- ного программного кода, его анализу, оп-тимизации методами рефакторинга. Защита отчетов по практическим и лабо-раторным работам Интерпретация ре-зультатов наблюденийза деятельностью обу-чающегося в процессепрактики

	проведена оптимизация и выполнена оценка качества полученного программного кода.	

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
- **5.2. 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.

5.3. Критерии оценивания презентаций

Оценка	**5**	**4**	**3**	**2**
Содержа-ние	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
Грамот-ность	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

6. 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, приводит 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 здания, 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 теста;

– записи лучше делать при прочтении не одного-двух абзацев, а целого параграфа или главы;

– конспектирование ведётся не с целью иметь определённый 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 станица 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.

7. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ КОНТРОЛЯ УСПЕВАЕМОСТИ

7.1. Паспорт фонда оценочных средств

№ п/п	Контролируемые разделы (темы) дисциплины	Код контроли-руемой компетенции (или ее части)	Наименование оценочного средства
1.	Раздел1. **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.	Тема1.1. 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 контроли-руемой 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	

7.2. 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.

7.3. 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, SWEBOK, 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.

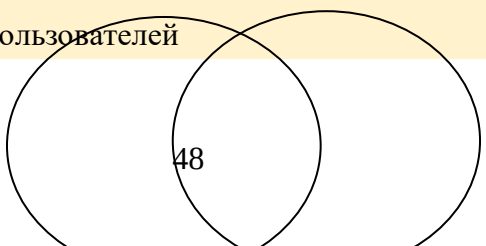
Выделяют 9 критериев качества требований:

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.

Потребности пользователей



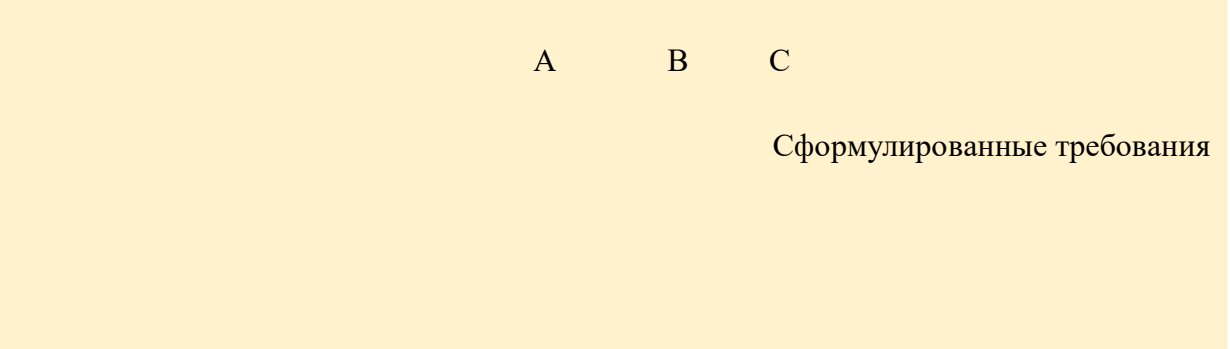


Рисунок 2 - Множества потребностей и требований

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 проверяемы, 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 соответствуе 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.

Титы 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:

J 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;

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

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

Z 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:

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

J функциональный тест программы установки;

J тестирование механизма лицензирования и функций защиты от пиратства;

J 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 багтрэкинговую 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 отрабатывают корректно, тогда качество всего функционала на уровне Smoke может быть оценено, как Примлемое.

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 фуннкции 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 отрабатывают корректно, тогда качество всего функционала на уровне Smoke может быть оценено, как Примлемое.

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 фуннкции 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.
Автоматизи-рованное тестирование Тема 1.7.

- Автоматизи-рованное 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 обслуживания 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.
- Больший 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 приступать 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 заголовок 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.
- Сверните 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 19.102-77законспектировать 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. Требование 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 впрямую 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

- Теориетические 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 двухмерный 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 x 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 x n. Сформулировать требования к программному продукту и разработать набор тест – кейсов.
7. В матрице m x 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 x 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 x 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 x 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

Лекция 1. Понятие модели жизненного цикла программного продукта.

За последние тридцать лет в экономике обозначилась четкая тенденция значительного сокращения жизненного цикла (ЖЦ) продуктов, которая является следствием ускорения научно-технического прогресса. Поэтому умение управлять ЖЦ продукта, способность осуществлять его разработку в кратчайшие сроки становятся важными конкурентными преимуществами. Современные изменения технологий производства и коммуникаций, резко возросшая подвижность капитала радикально меняют отношения на рынке и ставят теорию и практику экономической науки перед новыми проблемами, которые не находят своего решения в рамках равновесного подхода и парадигмы устойчивости неоклассической теории.

В современных экономических условиях их одностороннее использование неправомерно по следующим причинам :

— ускорение эволюции рынка привело к тому, что в ряде отраслей горизонт прогноза сократился до 6 месяцев

— за большее время рынок успевает значительно измениться, а не прийти к равновесию;

— усиление роли положительной обратной связи;

— доказанная нерациональность поведения клиентов.

Теперь на протяжении жизни одного поколения происходят такие изменения, которые раньше требовали столетий. Необходимость принимать решения в быстро изменяющихся условиях требует новой методологии, и

одной из ее составных частей является представление о жизненных циклах наблюдаемых процессов и явлений.

1. Общее представление о моделях жизненного цикла

1.1. Модель полного жизненного цикла. Термин «жизненный цикл» в настоящее время является общеупотребительным как в естественных и технических, так и в гуманитарных науках. В различных предметных словарях имеется достаточное количество определений ЖЦ, схожих друг с другом в основных чертах. Нас прежде всего интересует системотехническое определение, которое в форме модели полного ЖЦ приведено в работе В. Х. Буркова и В. А. Ирикова 1: «Модель полного жизненного цикла отдельного объекта представляет собой описание последовательности всех фаз, этапов его существования от замысла и появления («рождения») до исчезновения («отмирания»)».

Для модели полного ЖЦ характерны следующие два основных **свойства**

1) структура (состав этапов) модели ЖЦ не зависит от того, какой объект описывается, т. е. модель ЖЦ инвариантна по отношению к изменению типа объекта, отрасли и т. д., что делает ее универсальной и широко применимой;

2) в жизни в любой момент времени реально существует только процесс деятельности конкретных агентов по созданию конкретных результатов, и этот исходный первичный реальный процесс полностью описывается потоками моделей ЖЦ. Остальные типы моделей — это вторичное описание удобных для решения задач искусственно сконструированных надстроек над реальным процессом.

Модель полного ЖЦ организационной системы может быть представлена в форме перечня (комплекса) операций и их поэтапного

выполнения (действий). Каждому этапу ЖЦ соответствуют свои управленческие решения, поэтому важно постоянно отслеживать стадии ЖЦ управляемого объекта. Модель ЖЦ может детализироваться на комплексы операций, которые могут описываться моделями различных типов. В настоящее время используются модели ЖЦ различной детализации, некоторые из них включены в международные стандарты (например, стандарт ИСО 9000 по системам управления качеством продукции). Каждый этап модели ЖЦ описывается продолжительностью, сроками начала и окончания, результатами (технико-экономические показатели), объемом, затратами и другими характеристиками. По фиксированному набору ЖЦ с фиксированными сроками легко определить выпуск продукции, мощности, затраты ресурсов и т. д. Каждым этапом занимается соответствующая функциональная служба (снабжения, сбыта, капитального строительства и др.) Поскольку модель полного жизненного цикла объекта всесторонне описывает его развитие, то в рамках каждого аспекта человеческой деятельности возникает своя конкретная модель ЖЦ этого объекта, являющаяся проекцией модели полного ЖЦ на специфику данного вида деятельности.

Примеры моделей ЖЦ в различных сферах человеческой деятельности

Жизненный цикл продукта в маркетинге.. В этом цикле отчетливо выделяются четыре этапа.

1. Этап выведения на рынок — период медленного роста сбыта по мере выхода товара на рынок. В связи с большими затратами по выведению товара прибылей на данном этапе еще нет.

2. Этап роста — период быстрого восприятия товара рынком и быстрого роста прибылей.

3. Этап зрелости — период замедления темпов сбыта в связи с тем, что товар уже добился восприятия большинством потенциальных покупателей. Прибыли стабилизируются или снижаются в связи с ростом затрат на защиту товара от конкурентов.

4. Этап упадка — период, характеризующийся резким падением сбыта и снижением прибылей.

Жизненный цикл инновации.:

— этап 1 — фундаментальные научно-исследовательские работы (НИР). Цель — получение новых знаний;

— этап 2 — прикладные НИР. Цель — решение конкретной технической проблемы;

— этап 3 — опытно-конструкторские работы (ОКР). Цель — создание или усовершенствование образцов новой техники;

— этап 4 — освоение промышленного производства (нового продукта). Цель — испытание новой продукции; техническая и технологическая подготовка производства;

— этап 5 — промышленное производство.

ЖЦ изделия в стандартах серии ISO 9000 Стандартами серии ISO 9000 (ИСО 9000–1-94) по управлению качеством регламентированы следующие типовые стадии ЖЦ изделия:

1) маркетинг;

2) НИОКР (научно-исследовательские и опытно-конструкторские работы);

3) материально-техническое снабжение;

4) подготовка и разработка производственных процессов;

5) собственно производство;

6) контроль и испытание продукции (в процессе производства и на выходе);

7) упаковка и хранение готовой продукции;

8) распределение и реализация;

9) монтаж и эксплуатация;

10) техническая помощь в обслуживании;

11) утилизация после использования. (ИСО 9000-1-94. Общее руководство качеством и стандарты по обеспечению качества. URL:// www.docload.ru/Basesdoc/4/4993/index.htm (дата обращения: 10.12.2012))

2. Жизненный цикл экономических систем.:

1) формирования идей; 2

) отбора идей;

3) разработки замысла и его проверки;

4) разработки стратегии маркетинга;

5) анализа возможностей производства сбыта;

6) разработки товара;

7) испытания в рыночных условиях;

8) развертывания коммерческого производства

++++++++++++++++++++++++++++++++

Каскадная модель (классический жизненный цикл)

Эта модель обязана своим появлением У. Ройсу ([1], 1970 г.). Модель имеет и другое название – водопад (waterfall). Особенность модели – переход на следующую ступень осуществляется только после того, как будет полностью завершена работа на предыдущей стадии; возвратов на пройденные стадии не предусматривается (рис.5.4).

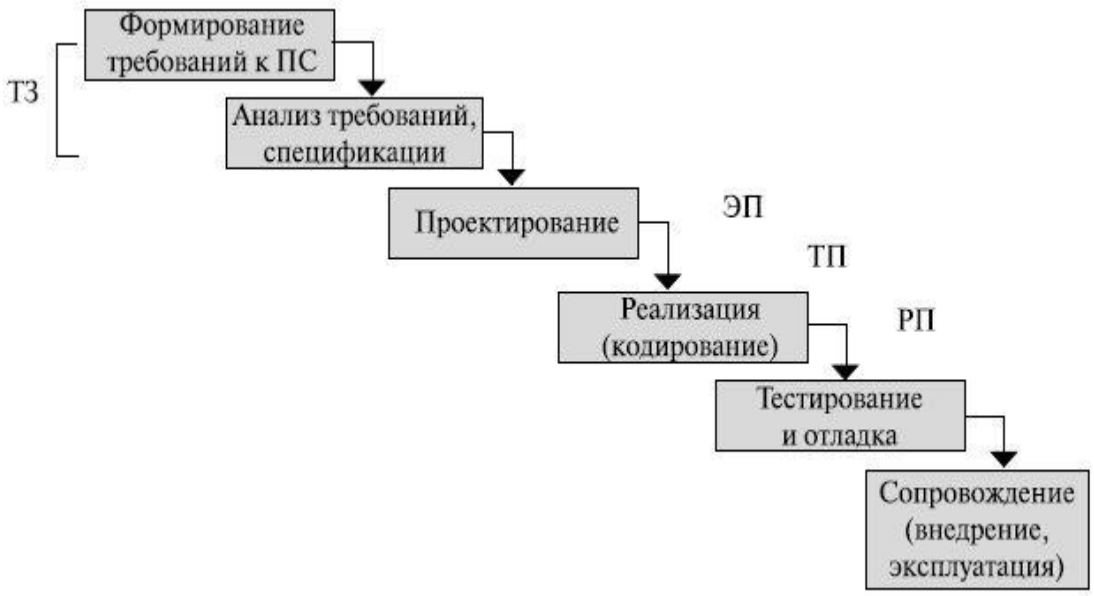


Рис. 5.4. Каскадная модель жизненного цикла программного обеспечения

Требования к разрабатываемой ПС, определенные на стадиях формирования и анализа, строго документируются в виде ТЗ и фиксируются на все время разработки проекта. Каждая стадия завершается выпуском полного комплекта документации (ТЗ, ЭП, ТП, РП), достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков. Критерием качества разработки при таком подходе является точность выполнения спецификаций ТЗ. Основное внимание разработчиков сосредоточивается на достижении оптимальных значений технических характеристик разрабатываемой ПС – производительности, объема занимаемой памяти и др.

Преимущества *каскадной модели*:

- *на каждой стадии формируется законченный набор проектной документации, отвечающей критериям полноты и согласованности;*
- *выполняемые в логической последовательности стадии работ позволяют планировать сроки завершения всех работ и соответствующие затраты.*

Каскадный подход хорошо зарекомендовал себя при построении ПС, для которых в самом начале проекта можно полно и четко сформулировать все требования. *Пока все это контролируется стандартами и различными комиссиями госприемки, схема работает хорошо*.

Недостатки каскадной модели:

- *выявление и устранение ошибок производится только на стадии тестирования, которое может существенно растянуться;*
- *реальные проекты часто требуют отклонения от стандартной последовательности шагов;*
- *цикл основан на точной формулировке исходных требований к ПС, реально в начале проекта требования заказчика определены лишь частично;*
- *результаты работ доступны заказчику только по завершении проекта.*

Итерационная модель ЖЦ ПС

С ростом коммерческих проектов выяснилось, что не всегда удается детально проработать проект будущей системы, поскольку многие аспекты ее функционирования в динамических сферах деятельности (бизнес) меняются, пока система создается. *Потребовалось изменить процесс разработки так, чтобы гарантировать внесение необходимых исправлений после завершения какого-либо этапа разработки. Так появилась итерационная модель ЖЦ ПС, называемая моделью с*

промежуточным контролем или моделью с циклическим повторением фаз.

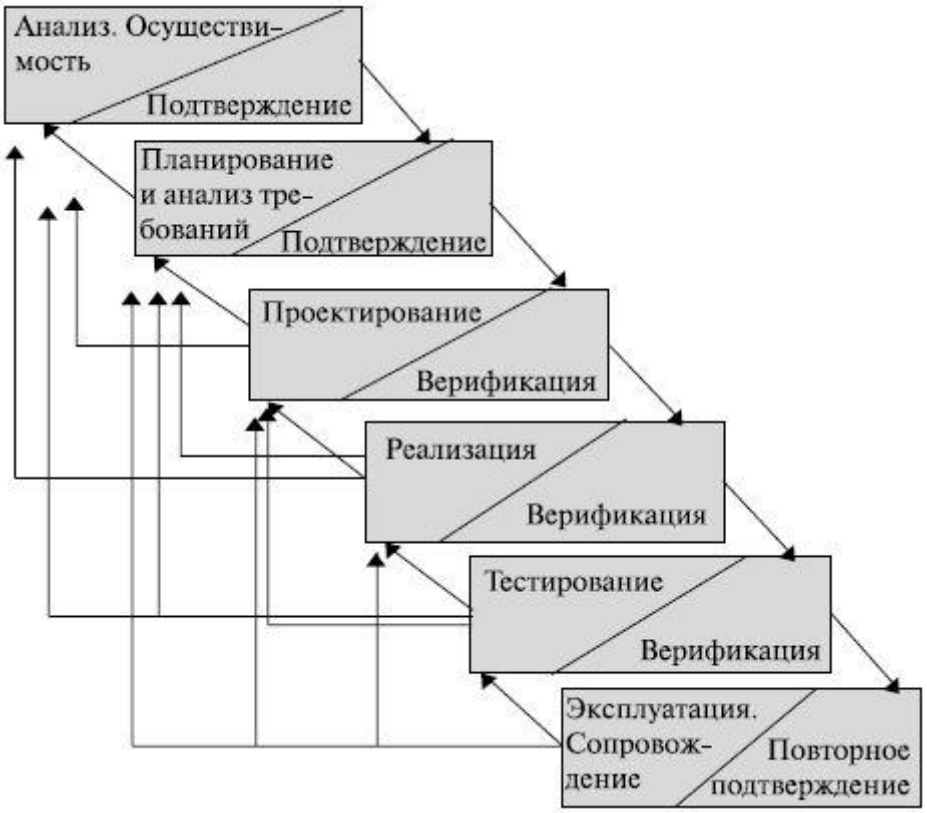


Рис. 5.5. Итерационная модель жизненного цикла программного обеспечения

В итерационной модели (рис.5.5) недостатки проектирования и программирования могут быть устранены позже путем частичного возврата на предыдущую стадию. Чем ниже уровень обнаружения ошибки, тем дороже ее исправление. Если стоимость усилий, необходимых для обнаружения и устранения ошибок на стадии написания кода, принять за единицу, то стоимость выявления и устранения ошибки на стадии выработки требований будет в 5-10 раз меньше, а стоимость выявления и устранения ошибки на стадии сопровождения – в 20 раз больше (рис.5.6).

Рис. 5.6. Зависимость стоимости и устранения ошибки от стадии разработки программного обеспечения

В такой ситуации огромное значение приобретает этап формулирования требований, составление спецификаций и создание плана системы. Программные архитекторы несут личную ответственность за все последующие изменения проектных решений. Объем документации исчисляется тысячами страниц, число утверждающих заседаний огромно. Многие проекты так никогда и не покидают этап планирования, впав в "*паралич* анализа". Одним из возможных путей исключения подобных ситуаций является макетирование ≡ прототипирование.

Прототипирование (макетирование)

Часто заказчик не может сформулировать требования по вводу, обработке или выводу данных для будущего программного продукта. Разработчик может сомневаться в приспособленности продукта к операционной системе, в форме диалога с пользователем или эффективности алгоритма. В таких случаях целесообразно использовать макетирование. *Основная цель макетирования – снять неопределенность в требованиях заказчика. Макетирование (прототипирование) – процесс создания модели требуемого продукта*.

Модель может принимать следующие формы.

1. Бумажный прототип (рисованная схема человеко-машинного диалога) или макет на основе ПК.

2. Работающий прототип, реализующий некоторую часть требуемых функций.

3. Существующая программа, характеристики которой должны быть улучшены.

Как показано на рис.5.7, макетирование основывается на многократном повторении итераций, в которых участвуют заказчик и разработчик.

Рис. 5.7. Итерации прототипирования программного обеспечения

Последовательность действий при макетировании представлена на рис.5.8. Прототипирование начинается со сбора и уточнения требований к создаваемой программной системе. Разработчик и заказчик совместно определяют цели ПО, устанавливают, какие требования известны, а какие предстоит доопределить. Затем выполняется быстрое проектирование. В нем сосредотачиваются на характеристиках, которые должны быть видимыми пользователю. Быстрое проектирование приводит к построению макета. *Макет оценивается заказчиком и используется для уточнения требований к ПО. Итерации продолжаются до тех пор, пока макет не выявит все требования заказчика и даст возможность разработчику понять, что должно быть сделано*.

Достоинства прототипирования – возможность обеспечения определения полных требований к системе.

Недостатки прототипирования:

- *заказчик может принять прототип за продукт;*
- *разработчик может принять прототип за продукт.*

Следует пояснить суть недостатков. Когда заказчик видит работающую версию ПС, он перестает сознавать, что в погоне за работающим вариантом ПС оставлены нерешенными многие вопросы качества и *удобства сопровождения* системы. Когда же заказчику об этом говорит разработчик, то ответом может быть возмущение и требование скорейшего превращения макета в рабочий продукт. Это отрицательно сказывается на управлении разработкой ПО.

Рис. 5.8. Последовательность действий при макетировании программного обеспечения

С другой стороны, для быстрого получения работающего макета разработчик часто идет на определенные компромиссы. Например, могут использоваться не самые подходящие языки программирования или операционная система. Для простой демонстрации может применяться неэффективный (простой) алгоритм. Спустя некоторое время

разработчик забывает о причинах, по которым эти средства не подходят. В результате далеко не идеальный выбранный вариант интегрируется в систему.

Прежде чем рассматривать другие модели ЖЦ ПО, которые пришли на смену *каскадной модели*, следует остановиться на стратегиях конструирования программных систем. Именно стратегия конструирования ПО во многом определяет модель ЖЦ ПО.

Стратегии конструирования ПО

Существует три стратегии конструирования программных систем:

1) однократный проход (каскадная стратегия, рассмотренная выше) – линейная последовательность этапов конструирования;

2) инкрементная стратегия. В начале процесса определяются все пользовательские и системные требования, оставшаяся часть конструирования выполняется в виде последовательности версий. Первая версия реализует часть запланированных возможностей, следующая версия реализует дополнительные возможности и т. д., пока не будет получена полная система;

3) эволюционная стратегия. Система также строится в виде последовательности версий, но в начале процесса определяются не все требования. Требования уточняются в результате разработки версий. Характеристики стратегий конструирования ПО с соответствии с требованиями стандарта IEEE/EIA 12207 приведены в табл. 5.1.

Таблица 5.1.

Стратегия конструирования	**В начале процесса определены все требования?**	**Множество циклов конструирования?**	**Промежуточное ПО распространяется?**
1. Однократный проход	Да	Нет	Нет
2. Инкрементная (запланированное улучшение продукта)	Да	Да	Может быть
3. Эволюционная	Нет	Да	Да

Инкрементная модель

Инкрементная модель является классическим примером инкрементной стратегии конструирования. Она объединяет элементы последовательной водопадной модели с итерационной философией макетирования (предложена Б. Боэмом как усовершенствование *каскадной модели*). Каждая линейная последовательность здесь вырабатывает поставляемый инкремент ПО. Например, ПО для обработки слов в 1-м инкременте (версии) реализует функции базовой обработки файлов, функции редактирования и документирования; во 2-м инкременте – более сложные возможности редактирования и документирования; в 3-м инкременте – проверку орфографии и грамматики; в 4-м инкременте – возможности компоновки страницы.

Первый инкремент приводит к получению базового продукта, реализующего базовые требования (правда, многие вспомогательные

требования остаются нереализованными). План следующего инкремента предусматривает модификацию базового продукта, обеспечивающую дополнительные характеристики и функциональность.

По своей природе инкрементный процесс итеративен, но, в отличие от макетирования, инкрементная модель обеспечивает на каждом инкременте работающий продукт.

Схема такой модели ЖЦ ПО приведена на рис.5.9. Одной из современных реализаций инкрементного подхода является экстремальное программирование (ориентировано на очень малые приращения функциональности) [23].

Рис. 5.9. Инкрементная модель разработки программного обеспечения

Спиральная модель ЖЦ ПО

Спиральная модель – классический пример применения эволюционной стратегии конструирования. Модель (автор Б. Боэм, 1988) базируется на лучших свойствах классического жизненного цикла и макетирования, к которым добавляется новый элемент – анализ риска, отсутствующий в этих парадигмах. Модель определяет четыре действия, представляемые четырьмя квадрантами спирали (рис.5.10).

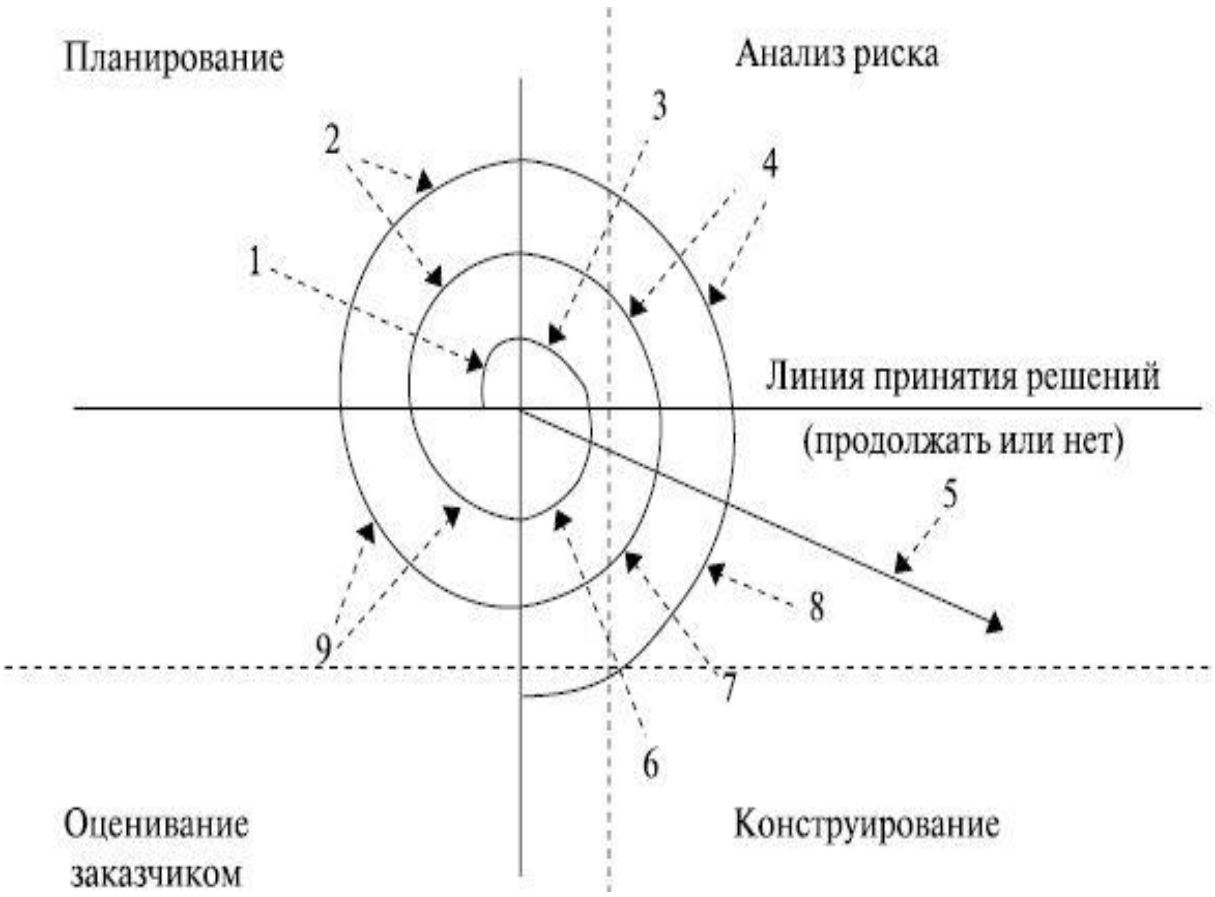


Рис. 5.10. Спиральная модель жизненного цикла программного обеспечения

1. Планирование – определение целей, вариантов и ограничений.
2. Анализ риска – анализ вариантов и распознавание/выбор риска.
3. Конструирование – разработка продукта следующего уровня.
4. Оценивание – оценка заказчиком текущих результатов конструирования.

Интегрирующий аспект *спиральной модели* очевиден при учете радиального измерения спирали. С каждой итерацией по спирали строятся все более полные версии ПС. В первом витке спирали определяются начальные цели, варианты и ограничения, распознается и анализируется риск. Если анализ риска показывает неопределенность требований, на помощь разработчику и заказчику приходит макетирование, используемое в квадранте конструирования.

Для дальнейшего определения проблемных и уточненных требований

может быть использовано моделирование. Заказчик оценивает инженерную (конструкторскую) работу и вносит предложения по модификации (квадрант оценки заказчиком). Следующая фаза планирования и анализа риска базируется на предложениях заказчика. В каждом цикле по спирали результаты анализа риска формируются в виде "продолжать, не продолжать". Если риск слишком велик, проект может быть остановлен.

В большинстве случаев движение по спирали продолжается, с каждым шагом продвигая разработчиков к более общей модели системы. В каждом цикле по спирали требуется конструирование (нижний правый квадрант), которое может быть реализовано классическим жизненным циклом или макетированием. Заметим, что количество действий по разработке (происходящих в правом нижнем квадранте) возрастает по мере продвижения от центра спирали.

Эти действия пронумерованы на рис.5.10 и имеют следующее содержание:

1) – начальный сбор требований и планирование проекта;
2) – та же работа, но на основе рекомендаций заказчика;
3) – анализ риска на основе начальных требований;
4) – анализ риска на основе реакции заказчика;
5) – переход к комплексной системе;
6) – начальный макет системы;
7) – следующий уровень макета;
8) – сконструированная система;
9) – оценивание заказчиком.

Достоинства *спиральной модели*:

1) наиболее реально (в виде эволюции) отображает разработку программного обеспечения;
2) позволяет явно учитывать риск на каждом витке эволюции разработки;

3) включает шаг системного подхода в итерационную структуру разработки;
4) использует моделирование для уменьшения риска и совершенствования программного изделия.

Недостатки *спиральной модели*:

1) сравнительная новизна (отсутствует достаточная статистика эффективности модели);
2) повышенные требования к заказчику;
3) трудности контроля и управления временем разработки.

Модель спирального процесса разработки является наиболее распространенной в настоящее время. Самыми известными ее вариантами являются RUP (Rational Unified Process) от фирмы Rational и MSF (Microsoft Solution Framework). В качестве языка моделирования используется язык UML (Unified Modeling Language). Создание системы предполагается проводить итерационно, двигаясь по спирали и, проходя через одни и те же стадии, на каждом витке уточнять характеристики будущего продукта. Казалось бы, теперь все хорошо: и планируется только то, что можно предвидеть, разрабатывается то, что запланировано, и пользователи начинают знакомиться с продуктом заранее, имея возможность внести необходимые коррективы.

Однако для этого нужны очень большие средства. Действительно, если раньше можно было создавать и распускать группы специалистов по мере необходимости, то теперь все они должны постоянно участвовать в проекте: архитекторы, программисты, тестировщики, инструкторы и т. д. Более того, усилия различных групп должны быть синхронизированы, чтобы своевременно отражать проектные решения и вносить необходимые изменения.

Рациональный унифицированный процесс

Рациональный *унифицированный процесс* (Rational Unified Process,

RUP) – одна из лучших методологий разработки программного обеспечения [32]. Основываясь на опыте многих успешных программных проектов, RUP позволяет создавать сложные программные системы, основываясь на индустриальных методах разработки. Предпосылки для разработки RUP зародились в начале 1980-х гг. в Rational Software corporation. В начале 2003 г. Rational приобрела IBM. Одним из основных столпов, на которые опирается RUP, является процесс создания моделей при помощи унифицированного языка моделирования (UML).

RUP – одна из спиральных методологий разработки программного обеспечения. Методология поддерживается и развивается компанией Rational Software. В качестве языка моделирования в общей базе знаний используется язык Unified Modelling Language (UML). Итерационная и инкрементная разработка программного обеспечения в RUP предполагает разделение проекта на несколько проектов, которые выполняются последовательно, и каждая итерация разработки четко определена набором целей, которые должны быть достигнуты в конце итерации. Конечная итерация предполагает, что набор целей итерации должен в точности совпадать с набором целей, указанных заказчиком продукта, то есть все требования должны быть выполнены.

Процесс предполагает эволюционирование моделей; итерация цикла разработки однозначно соответствует определенной версии модели программного обеспечения. Каждая из итераций содержит элементы управления *жизненным циклом программного обеспечения*: анализ и дизайн (моделирование), реализация, интегрирование, тестирование, внедрение. В этом смысле RUP является реализацией *спиральной модели*, хотя довольно часто изображается в виде графика-таблицы (рис.5.11).

На данном рисунке представлены два измерения: горизонтальная ось представляет время и показывает временные аспекты жизненного цикла процесса; вертикальная ось представляет дисциплины, которые определяют

физическую структуру процесса. На рис.5.11 видно, как с течением времени изменяются акценты в проекте. Например, в ранних итерациях больше времени отводится требованиям; в поздних итерациях больше времени отводится реализации. Горизонтальная ось сформирована из временных отрезков – итераций, каждая из которых является самостоятельным циклом разработки; цель цикла – принести в конечной продукт некоторую заранее определенную осязаемую доработку, полезную с точки зрения заинтересованных лиц.

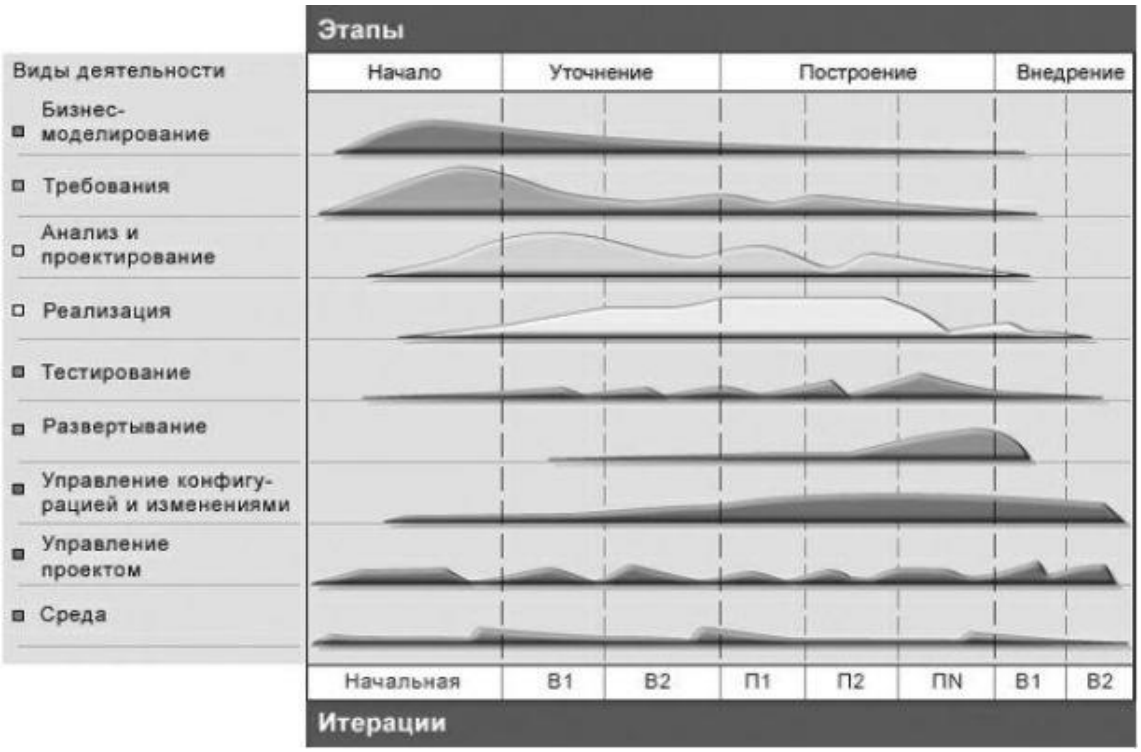


Рис. 5.11. Структура рационального унифицированного процесса (RUP)

По оси времени жизненный цикл делится на четыре основные фазы.

1. Начало (Inception) – формирование концепции проекта, понимание того, что мы создаем, представление о продукте (vision), разработка бизнес-плана (business case), подготовка прототипа программы или частичного решения. Это фаза сбора информации и анализа требований, определение образа проекта в целом. Цель – получить поддержку и

финансирование. В конечной итерации результат этого этапа – техническое задание.

2. Проектирование, разработка (Elaboration) – уточнение плана, понимание того, как мы это создаем, проектирование, планирование необходимых действий и ресурсов, детализация особенностей. Завершается этап исполняемой архитектурой, когда все архитектурные решения приняты и риски учтены. Исполняемая архитектура представляет собой работающее программное обеспечение, которое демонстрирует реализацию основных архитектурных решений. В конечной итерации это – технический проект.

3. Реализация, создание системы (Construction) – этап расширения функциональности системы, заложенной в архитектуре. В конечной итерации это – рабочий проект.

4. Внедрение, развертывание (Transition). Создание конечной версии продукта. Фаза внедрения продукта, поставка продукта конкретному пользователю (тиражирование, доставка и обучение).

Вертикальная ось состоит из дисциплин, каждая из них может быть более детально расписана с точки зрения выполняемых задач, ответственных за них ролей, продуктов, которые подаются задачам на вход и выпускаются в ходе их выполнения и т.д.

По этой оси располагаются ключевые дисциплины жизненного цикла RUP, которые часто на русском языке называют процессами, хотя это не совсем верно с точки зрения данной методологии, поддерживаемые инструментальными средствами IBM (и/или третьих фирм).

1. Бизнес-анализ и моделирование (*Business modeling*) обеспечивает реализацию принципов моделирования с целью изучения бизнеса организации и накопления знаний о нем, оптимизации бизнес-процессов и принятия решения об их частичной или полной автоматизации.

2. Управление требованиями (Requirements) посвящено получению информации от заинтересованных лиц и ее преобразованию в набор

требований, определяющих содержание разрабатываемой системы и подробно описывающих ожидания от того, что система должна делать.

3. Анализ и проектирование (Analysis and design) охватывает процедуры преобразования требований в промежуточные описания (модели), представляющие, как эти требования должны быть реализованы.

4. Реализация (Implementation) охватывает разработку кода, тестирование на уровне разработчиков и интеграцию компонентов, подсистем и всей системы в соответствии с установленными спецификациями.

5. Тестирование (Test) посвящено оценке качества создаваемого продукта.

6. Развертывание (Deployment) охватывает операции, имеющие место при передаче продуктов заказчикам и обеспечении доступности продукта конечным пользователям.

7. Конфигурационное управление и управление изменениями (Configuration management) посвящено синхронизации промежуточных и конечных продуктов и управлению их развитием в ходе проекта и поиском скрытых проблем.

8. Управление проектом (Management) посвящено планированию проекта, управлению рисками, контролю хода его выполнения и непрерывной оценке ключевых показателей.

9. Управление средой (Environment) включает элементы формирования среды разработки информационной системы и поддержки проектной деятельности.

В зависимости от специфики проекта могут быть использованы любые средства IBM Rational, а также третьих фирм. В RUP рекомендовано следовать шести практикам, позволяющим успешно разрабатывать проект: итеративная разработка; управление требованиями; использование модульных архитектур; визуальное моделирование; проверка качества;

отслеживание изменений.

Неотъемлемую часть RUP составляют артефакты (artefact), прецеденты (*precedent*) и роли (role). Артефакты – это некоторые продукты проекта, порождаемые или используемые в нем при работе над окончательным продуктом. Прецеденты – это последовательности действий, выполняемых системой для получения наблюдаемого результата. Фактически любой результат работы индивидуума или группы является артефактом, будь то документ анализа, элемент модели, файл кода, тестовый скрипт, описание ошибки и т. п. За создание того или иного вида артефактов отвечают определенные специалисты. Таким образом, RUP четко определяет обязанности – роли – каждого члена группы разработки на том или ином этапе, то есть когда и кто должен создать тот или иной артефакт. Весь процесс разработки программной системы рассматривается в RUP как процесс создания артефактов – начиная с первоначальных документов анализа и заканчивая исполняемыми модулями, руководствами пользователя и т.п.

Для компьютерной поддержки процессов RUP в IBM разработан широкий набор инструментальных средств:

1) Rational Rose – CASE-*средство визуального моделирования* информационных систем, имеющее возможности генерирования элементов кода. Специальная редакция продукта – Rational Rose RealTime – позволяет на выходе получить исполняемый модуль;
2) Rational *Requisite* Pro – средство управления требованиями, которое позволяет создавать, структурировать, устанавливать приоритеты, отслеживать, контролировать изменения требований, возникающие на любом этапе разработки компонентов приложения;
3) Rational ClearQuest – продукт для управления изменениями и отслеживания дефектов в проекте (bug tracking), тесно интегрирующийся со средствами тестирования и управления

требованиями и представляющий собой единую среду для связывания всех ошибок и документов между собой;

4) Rational SoDA – продукт для автоматического генерирования проектной документации, позволяющий установить корпоративный стандарт на внутрифирменные документы. Возможно также приведение документации к уже существующим стандартам (ISO, CMM);
5) Rational Purify, Rational *Quantify* Rational PureCoverage, – средства тестирования и отладки;
6) Rational Visual *Quantify* – средство измерения характеристик для разработчиков приложений и компонентов, программирующих на C/C++, Visual Basic и Java; помогает определять и устранять узкие места в производительности ПО;
7) Rational Visual PureCoverage – автоматически определяет области кода, которые не подвергаются тестированию;
8) Rational ClearCase – продукт для управления конфигурацией программ (Rational's Software Configuration Management, *SCM*), позволяющий производить версионный контроль всех документов проекта. С его помощью можно поддерживать несколько версий проектов одновременно, быстро переключаясь между ними. Rational *Requisite* Pro поддерживает обновления и отслеживает изменения в требованиях для группы разработчиков;
9) *SQA* TeamTest – средство *автоматизации тестирования*;
10) Rational TestManager – система управления тестированием, которая объединяет все связанные с тестированием инструментальные средства, артефакты, сценарии и данные;
11) Rational Robot – инструмент для создания, модификации и автоматического запуска тестов;

12) SiteLoad, SiteCheck – средства тестирования Web-сайтов на производительность и наличие неработающих ссылок;
13) Rational PerformanceStudio – измерение и предсказание характеристик производительности систем.

Этот набор продуктов постоянно совершенствуется и пополняется. Так, например, недавний продукт IBM Rational *Software Architect* (RSA) является частью IBM Software *Development Platform* – набора инструментов, поддерживающих жизненный цикл разработки программных систем [7, 8]. Продукт IBM Rational *Software Architect* предназначен для построения моделей разрабатываемых программных систем с использованием унифицированного языка моделирования UML 2.0, прежде всего моделей архитектуры разрабатываемого приложения. Тем не менее, RSA объединяет в себе функции таких программных продуктов, как Rational Application Developer, Rational Web Developer и Rational *Software Modeler*, тем самым предоставляя возможность архитекторам и аналитикам создавать различные представления разрабатываемой информационной системы с использованием языка UML 2.0, а разработчикам – выполнять разработку J2EE, XML, веб-сервисов и т.д.

Следуя принципам RUP, Rational *Software Architect* позволяет создавать необходимые модели в рамках рабочих процессов таких дисциплин, как:

1) бизнес-анализ и моделирование (*Business modeling*);
2) управление требованиями (Requirements);
3) анализ и проектирование (Analysis and Design);
4) реализация (Implementation).

Кроме того, Rational *Software Architect* поддерживает технологию разработки, управляемой моделями (model-driven development, MDD), которая позволяет моделировать программное обеспечение на различных уровнях абстракции с возможностью трассируемости

MSF (Microsoft Solution Framework)

В 1994 году, стремясь достичь максимальной отдачи от IT-проектов, Microsoft выпустила в свет пакет руководств по эффективному проектированию, разработке, внедрению и сопровождению решений, построенных на основе своих технологий. Эти знания базируются на опыте, полученном Microsoft при работе над большими проектами по разработке и сопровождению программного обеспечения, опыте консультантов Microsoft и лучшем из того, что накопила на данный момент IT-индустрия. Все это представлено в виде двух взаимосвязанных и хорошо дополняющих друг друга областей знаний: Microsoft Solutions Framework (MSF) и Microsoft Operations Framework (MOF).

Следует отметить, что Microsoft разработала на базе общих методов MSF методики для прикладного и специализированного применения. Причем Microsoft сертифицирует экспертов именно по прикладным знаниям в применении MSF (например, сертификация *MCTS* 74-131 по экспертизе в методике управления проектами). Перед тем как изучать методы MSF, следует сначала определить, какой прикладной вариант MSF имеется в виду.

Наиболее популярные прикладные варианты MSF, разработанные Microsoft:

- методика внедрения решений в области управления проектами;
- методика управления IT-проектами на базе методологий MSF и Agile.

Важность прикладных вариантов MSF подчеркивает тот факт, что в "чистом варианте" саму методику MSF в своих IT-проектах компания Microsoft не использует [36]. В проектах Microsoft Consulting Services применяется гибридная методология MSF и Agile. Несмотря на внешние существенные различия прикладных вариантов MSF, разработанных экспертами Microsoft, база методов MSF для них остается общей и отражает

общие методологические подходы к итеративному ведению проектов [35].

MOF призван обеспечить организации, создающие критически важные (mission-critical) IT решения на базе продуктов и технологий Майкрософт, техническим руководством по достижению их надежности (reliability), доступности (availability), *удобства сопровождения* (supportability) и управляемости (manageability). MOF затрагивает вопросы, связанные с организацией персонала и процессов; технологиями и менеджментом в условиях сложных (complex), распределенных (distributed) и разнородных (heterogeneous) IT-сред. MOF основан на лучших производственных методиках, собранных в IT Infrastructure Library (ITIL), составленной Central Computer and Telecommunications Agency – Агентством правительства Великобритании.

Создание бизнес-решения в рамках отведенных времени и бюджета требует наличия испытанной методологической основы. MSF предлагает проверенные методики для планирования, проектирования, разработки и внедрения успешных IT-решений. Благодаря своей гибкости, масштабируемости и отсутствию жестких инструкций MSF способен удовлетворить нужды организации или проектной группы любого размера. Методология MSF состоит из принципов, моделей и дисциплин по управлению персоналом, процессами, технологическими элементами и связанными со всеми этими факторами вопросами, характерными для большинства проектов. MSF состоит из двух моделей и трех дисциплин. Они подробно описаны в 5 whitepapers. Начинать изучение MSF лучше с моделей (модель проектной группы, модель процессов), а затем перейти к дисциплинам (дисциплина управление проектами, дисциплина управление рисками, дисциплина управление подготовкой).

Модель процессов MSF (MSF process model) представляет общую методологию разработки и внедрения IT-решений. Особенность этой модели состоит в том, что благодаря своей гибкости и отсутствию жестко

навязываемых процедур она может быть применена при разработке весьма широкого круга IT-проектов. Эта модель сочетает в себе свойства двух стандартных производственных моделей: каскадной (waterfall) и спиральной (spiral). Модель процессов в MSF 3.0 была дополнена еще одним инновационным аспектом: она покрывает весь жизненный цикл создания решения, начиная с его отправной точки и заканчивая непосредственно внедрением. Такой подход помогает проектным группам сфокусировать свое внимание на бизнесотдаче (business value) решения, поскольку эта отдача становится реальной лишь после завершения внедрения и начала использования продукта.

Процесс MSF ориентирован на "вехи" (milestones) – ключевые точки проекта, характеризующие достижение в его рамках какого-либо существенного (промежуточного либо конечного) результата. Этот результат может быть оценен и проанализирован, что подразумевает ответы на вопросы: "Пришла ли проектная группа к однозначному пониманию целей и рамок проекта?", "В достаточной ли степени готов план действий?", "Соответствует ли продукт утвержденной спецификации?", "Удовлетворяет ли решение нужды заказчика?" и т. д.

Модель процессов MSF учитывает постоянные изменения проектных требований. Она исходит из того, что разработка решения должна состоять из коротких циклов, создающих поступательное движение от простейших версий решения к его окончательному виду.

Особенностями модели процессов MSF являются:

- подход, основанный на фазах и вехах;
- итеративный подход;
- интегрированный подход к созданию и внедрению решений.

Модель процессов включает такие основные фазы процесса разработки, как:

- выработка концепции (Envisioning);
- планирование (Planning);
- разработка (Developing);
- стабилизация (Stabilizing);
- внедрение (Deploying).

Кроме этого, существует большое количество промежуточных вех, которые показывают достижение в ходе проекта определенного прогресса и расчленяют большие сегменты работы на меньшие, обозримые участки. Для каждой фазы модели процессов MSF определяет:

- что (какие артефакты) является результатом этой фазы;
- над чем работает каждый из ролевых кластеров на этой фазе.

В рамках MSF программный код, документация, дизайн, планы и другие рабочие материалы создаются, как правило, итеративными методами. MSF рекомендует начинать разработку решения с построения, тестирования и внедрения его базовой функциональности. Затем к решению добавляются все новые и новые возможности. Такая стратегия именуется стратегией версионирования. Несмотря на то, что для малых проектов может быть достаточным выпуск одной версии, рекомендуется не упускать возможности создания для одного решения ряда версий. С созданием новых версий эволюционирует функциональность решения.

Итеративный подход к процессу разработки требует использования гибкого способа ведения документации. "Живые" документы (living documents) должны изменяться по мере эволюции проекта вместе с изменениями требований к конечному продукту. В рамках MSF предлагается ряд шаблонов стандартных документов, которые являются артефактами каждой стадии разработки продукта и могут быть использованы для планирования и контроля процесса разработки.

Решение не представляет бизнес-ценности, пока оно не внедрено.

Именно по этой причине модель процессов MSF содержит весь жизненный цикл создания решения, включая его внедрение – вплоть до момента, когда решение начинает давать отдачу.

ЛИСТ
изменений рабочей учебной программы по учебной дисциплине
МДК.01.01 Разработка 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 списков 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.01.01 «Разработка программных модулей» 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.

Программы 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.01.01 «Разработка программных модулей» 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.01.01 «Разработка программных модулей», 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 часов. 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