МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

УТВЕРЖДАЮ
Директор ИЭиАС
С.И. Лукьянов

26.02.2020 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

ТЕХНОЛОГИИ ТЕСТИРОВАНИЯ ПРОГРАММНЫХ ПРОДУКТОВ ПОСТАВЛЯЕМЫХ РАЗРАБОТЧИКОМ НА СТОРОНЕ ПОЛЬЗОВАТЕЛЯ

Направление подготовки (специальность)
09.04.01 Информатика и вычислительная техника

Направленность (профиль/специализация) программы
Программное обеспечение средств вычислительной техники и автоматизированных систем

Уровень высшего образования - магистратура

Форма обучения
очная

Институт/ факультет	Институт энергетики и автоматизированных систем
Кафедра	Вычислительной техники и программирования
Курс	2
Семестр	3

Магнитогорск
2020 год

Рабочая программа составлена на основе ФГОС ВО - магистратура по направлению подготовки 09.04.01 Информатика и вычислительная техника (приказ Минобрнауки России от 19.09.2017 г. № 918)

Рабочая программа рассмотрена и одобрена на заседании кафедры вычислительной техники и программирования
19.02.2020 г. протокол № 5

Зав. кафедрой _____ О.С. Логунова

Рабочая программа одобрена методической комиссией ИЭ и АС
26.02.2020 г. протокол № 5

Председатель _____ С.И. Лукьянов

Рабочая программа составлена:
доцент кафедры ВТ и П, канд. техн. наук _____ Л.Г.Егорова

Рецензент:
Начальник отдела технологических платформ
ООО Компас Плюс , канд. техн. наук _____ Д.С.Сафонов

Лист актуализации рабочей программы

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2021 - 2022 учебном году на заседании кафедры вычислительной техники и программирования

Протокол от __ _____ 20__ г. № __
Зав. кафедрой _____ О.С. Логунова

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2022 - 2023 учебном году на заседании кафедры вычислительной техники и программирования

Протокол от __ _____ 20__ г. № __
Зав. кафедрой _____ О.С. Логунова

1 Цели освоения дисциплины (модуля)

Цель изучения курса «Технологии тестирования программных продуктов поставляемых разработчиком» - ознакомление студентов с основными видами и методами тестирования программного обеспечения на стороне клиента. В курсе изучаются способы обеспечения качества пользовательского программного обеспечения. В ходе изучения дисциплины необходимо решить основные задачи:

1. Проверка работоспособности программного обеспечения в реальных ситуациях так, как задумывалось при его создании.

2. Проверка работоспособности всех доступных функций.

3. Проверка работоспособности программного обеспечения на наличие ошибок и сбоев, которые мешают ему выполнять свои основные функции.

2 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина Технологии тестирования программных продуктов поставляемых разработчиком на стороне пользователя входит в обязательную часть учебного плана образовательной программы.

Для изучения дисциплины необходимы знания (умения, владения), сформированные в результате изучения дисциплин/ практик:

Информационно-управляющие системы

Конфигурирование на платформе TranzAxis

Автоматизированная система TranzWare для розничных банковских процессов

Oracle Database: продвинутые аспекты программирования и настройки производительности

Проектирование и тестирование сложных пользовательских интерфейсов

Современные розничные финансовые платформы на примере TranzAxis

Программное обеспечение для управления проектами в финансовой индустрии

Стандарты PA/PCI DSS в финансовой индустрии

Технологии Pl/SQL

Знания (умения, владения), полученные при изучении данной дисциплины будут необходимы для изучения дисциплин/практик:

Автоматизированная система TranzWare для розничных банковских процессов

Выполнение и защита выпускной квалификационной работы

Конфигурирование на платформе TranzAxis

Производственная - технологическая (проектно-технологическая) практика

Производственная-преддипломная практика

3 Компетенции обучающегося, формируемые в результате освоения дисциплины (модуля) и планируемые результаты обучения

В результате освоения дисциплины (модуля) «Технологии тестирования программных продуктов поставляемых разработчиком на стороне пользователя» обучающийся должен обладать следующими компетенциями:

Код индикатора	Индикатор достижения компетенции
ОПК-5 Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем;	
ОПК-5.1	Определяет необходимость и участвует в разработке и модернизации программного и аппаратного обеспечение информационных и автоматизированных систем

4. Структура, объём и содержание дисциплины (модуля)

Общая трудоемкость дисциплины составляет 3 зачетных единиц 108 акад. часов, в том числе:

– контактная работа – 52,95 акад. часов:

– аудиторная – 51 акад. часов;

– внеаудиторная – 1,95 акад. часов

– самостоятельная работа – 55,05 акад. часов;

Форма аттестации - курсовая работа, зачет

Раздел/ тема дисциплины	Семестр	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа студента	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код компетенции
		Лек.	лаб. зан.	практ. зан.				
1. Основные понятия пользовательского приемочного тестирования								
1.1 Типы приемочного тестирования. Альфа/бета-тестирование. Контрактное приемочное тестирование. Законодательное приемочное тестирование. Операционное приемочное тестирование. Тестирование по стратегии черного ящика.	3	2	4/4И		8	1. Подготовка к лабораторным занятиям 2. Выполнение лабораторных работ 3. Самостоятельное изучение учебной и научной литературы	1. Беседа - обсуждение 2. Проверка индивидуальных заданий 3. Устный опрос.	ОПК-5.1
Итого по разделу		2	4/4И		8			
2. Методология тестирования на стороне клиента								
2.1 Тестирование с помощью разведки. Сканирование портов. Отображение видимого контента. Поиск скрытого контента. Поиск параметров отладки и разработки. Определение точек ввода данных. Определение используемых технологий. Отображение возможных векторов атаки.	3	2	5/2И		6	1. Подготовка к лабораторным занятиям 2. Выполнение лабораторных работ 3. Самостоятельное изучение учебной и научной литературы	1. Беседа - обсуждение 2. Проверка индивидуальных заданий 3. Устный опрос.	ОПК-5.1
2.2 Тестирование контроля доступа. Аутентификация. Управление сессиями. Контроль доступа.		2	5/2И		6	1. Подготовка к лабораторным занятиям 2. Выполнение лабораторных работ 3. Самостоятельное изучение учебной и научной литературы	1. Беседа - обсуждение 2. Проверка индивидуальных заданий 3. Устный опрос.	ОПК-5.1

2.3 Проверка входных данных. Фаззинг параметров. Тестирование SQL-инъекций. Тестирование XSS-уязвимостей. Тестирование инъекций в HTTP заголовках. Тестирование переадресаций. Тестирование инъекций команд ОС. Тестирование уязвимости Pah Traversal. Тестирование HTML/JavaScript-инъекций. Тестирование RFI и LFI. Тестирование SMTP-инъекций. Тестирование SOAP-инъекций. Тестирование LDAP-инъекций. Тестирование XPath-инъекций. Тестирование XXE-инъекций. Тестирование внедрения шаблона.		2	5/2И		6	1. Подготовка к лабораторным занятиям 2. Выполнение лабораторных работ 3. Самостоятельное изучение учебной и научной литературы	1. Беседа - обсуждение 2. Проверка индивидуальных заданий 3. Устный опрос.	ОПК-5.1
2.4 Тестирование логики приложения. Определение векторов атаки. Тестирование передачи данных на стороне клиента. Тестирование валидации данных на стороне клиента. Тестирование компонентов толстых клиентов. Тестирование логики многоступенчатых механизмов. Тестирование обхода аутентификации. Тестирование прав доступа. Тестирование логики транзакций. Тестирование IDOR-уязвимостей.		2	5/2И		8	1. Подготовка к лабораторным занятиям 2. Выполнение лабораторных работ 3. Самостоятельное изучение учебной и научной литературы	1. Беседа - обсуждение 2. Проверка индивидуальных заданий 3. Устный опрос.	ОПК-5.1
2.5 Изучение инфраструктуры приложения. Тестирование разделения в среде виртуального хостинга. Тестирование разделения между ASP-приложениями. Тестирование уязвимостей на сервере. Проверка стандартных учётных записей. Определение стандартного контента на сайте. Определение опасных HTTP-методов. Тестирование прокси.		2	5/2И		6	1. Подготовка к лабораторным занятиям 2. Выполнение лабораторных работ 3. Самостоятельное изучение учебной и научной литературы	1. Беседа - обсуждение 2. Проверка индивидуальных заданий 3. Устный опрос.	ОПК-5.1

2.6 Прочие виды тестирования. Тестирование DOM-модели. Тестирование frame-инъекций. Проверка локальных уязвимостей. Проверка параметров cookies. Определение конфиденциальных данных в URL-параметрах. Проверка наличия слабых SSL-шифров. Анализ HTTP-заголовков		2	5/2И		6	1. Подготовка к лабораторным занятиям 2. Выполнение лабораторных работ 3. Самостоятельное изучение учебной и научной литературы	1. Беседа - обсуждение 2. Проверка индивидуальных заданий 3. Устный опрос.	ОПК-5.1
Итого по разделу		12	30/12И		38			
3. Стандарты регламентирующие процесс тестирования								
3.1 IEEE 12207/ISO/IEC 12207-2008 Software Life Cycle Processes. ISO/IEC 9126-1:2001 Software Engineering – Software Product Quality. IEEE 829-1998 Standard for Software Test Documentation. IEEE 1008-1987 (R1993, R2002) Standard for Software Unit Testing. ISO/IEC 12119:1994 Information Technology. Software Packages – Quality Requirements and Testing. ISO/IEC 20 000:2005. Процессы, сертификация ISO 20 000.	3	3			9,05	1. Самостоятельное изучение учебной и научной литературы	1. Беседа - обсуждение	ОПК-5.1
Итого по разделу		3			9,05			
Итого за семестр		17	34/16И		55,05		зачёт,кр	
Итого по дисциплине		17	34/16И		55,05		курсовая работа, зачет	

5 Образовательные технологии

1. Традиционные образовательные технологии, ориентированные на организацию образовательного процесса и предполагающие прямую трансляцию знаний от преподавателя к студенту.

Формы учебных занятий с использованием традиционных технологий:

Лабораторная работа – организация учебной работы с реальными материальными и информационными объектами, экспериментальная работа с аналоговыми моделями реальных объектов.

2. Технологии проблемного обучения – организация образовательного процесса, которая предполагает постановку проблемных вопросов, создание учебных проблемных ситуаций для стимулирования активной познавательной деятельности студентов.

Формы учебных занятий с использованием технологий проблемного обучения:

Практическое занятие в форме практикума – организация учебной работы, направленная на решение комплексной учебно-познавательной задачи, требующей от студента применения как научно-теоретических знаний, так и практических навыков.

6 Учебно-методическое обеспечение самостоятельной работы обучающихся

Представлено в приложении 1.

7 Оценочные средства для проведения промежуточной аттестации

Представлены в приложении 2.

8 Учебно-методическое и информационное обеспечение дисциплины (модуля)

а) Основная литература:

1. Трояновский, В. М. Программная инженерия информационно-управляющих систем в свете прикладной теории случайных процессов : учеб. пособие / В.М. Трояновский. — Москва : ИНФРА-М, 2019. — 325 с. + Доп. материалы [Электронный ресурс; Режим доступа: http: new.znanium.com]. —(Высшее образование: Магистратура). — www.dx.doi.org/10.12737/textbook_5ad88bf5c35cd8.81685342. - ISBN 978-5-8199-0824-2. - Текст : электронный. - URL:
https://znanium.com/catalog/product/1003316
(дата обращения: 28.10.2020). – Режим доступа: по подписке.

2. Плаксин, М. А. Тестирование и отладка программ для профессионалов будущих и настоящих / М. А. Плаксин. — 4-е изд., электрон. — Москва : Лаборатория знаний, 2020. — 170 с. - ISBN 978-5-00101-810-0. - Текст : электронный. - URL:
https://znanium.com/catalog/product/1093870

б) Дополнительная литература:

1. Антамошкин, О. А. Программная инженерия. Теория и практика [Электронный ресурс] : учебник / О. А. Антамошкин. - Красноярск: Сиб. Федер. ун-т, 2012. - 247 с. - ISBN 978-5-7638-2511-4. - Текст : электронный. - URL:
https://znanium.com/catalog/product/492527
(дата обращения: 28.10.2020). – Режим доступа: по подписке.

в) Методические указания:

Представлены в приложении 1.

г) Программное обеспечение и Интернет-ресурсы:

Программное обеспечение

Наименование ПО	№ договора	Срок действия лицензии
MS Windows 7 Professional(для классов)	Д-1227-18 от 08.10.2018	11.10.2021
MS Office 2007 Professional	№ 135 от 17.09.2007	бессрочно
Kaspersky Endpoint Security для бизнеса-Стандартный	Д-300-18 от 21.03.2018	28.01.2020
MS Visual Studio 2010 Professional(для класса)	Д-1227-18 от 08.10.2018	11.10.2021
SCO OpenServer	свободно распространяемое ПО	бессрочно
Oracle Open JDK	свободно распространяемое ПО	бессрочно
MS Visual Studio 2017 Community Edition	свободно распространяемое ПО	бессрочно
MS Visual Studio Code	свободно распространяемое ПО	бессрочно

MS Visual Studio 2013 Professional(для класса)	Д-1227-18 от 08.10.2018	11.10.2021

Профессиональные базы данных и информационные справочные системы

Название курса	Ссылка
Национальная информационно-аналитическая система – Российский индекс научного цитирования (РИНЦ)	URL: https://elibrary.ru/project_risc.asp
Информационная система - Единое окно доступа к информационным ресурсам	URL: http://window.edu.ru/
Федеральное государственное бюджетное учреждение «Федеральный институт промышленной собственности»	URL: http://www1.fips.ru/

9 Материально-техническое обеспечение дисциплины (модуля)

Материально-техническое обеспечение дисциплины включает:

1. Лекционная аудитория ауд. 282. Мультимедийные средства хранения, передачи и представления информации.

2. Компьютерные классы Центра информационных технологий ФГБОУ ВО «МГТУ». Персональные компьютеры, объединенные в локальные сети с выходом в Internet, оснащенные современными программно-методическими комплексами для решения задач в области информатики и вычислительной техники.

3. Аудитории для самостоятельной работы: компьютерные классы; читальные залы библиотеки. Все классы УИТ и АСУ с персональными компьютерами, выходом в Интернет и с доступом в электронную информационно-образовательную среду университета.

4. Аудиторий для групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации. Ауд. 282 и классы УИТ и АСУ.

5. Помещения для самостоятельной работы обучающихся, оснащенных компьютерной техникой с возможностью подключения к сети «Интернет» и наличием доступа в электронную информационно-образовательную среду организации. Классы УИТ и АСУ.

6. Помещения для хранения и профилактического обслуживания учебного оборудования. Центр информационных технологий – ауд.372.

Приложение 1

Учебно-методическое обеспечение самостоятельной работы обучающихся

Лабораторная работа 1

Анализ бизнес-требований и разработка спецификаций к программному продукту

Цель работы: изучение требований к создаваемому программному продукту, разработка технического задания

Этапы выполнения работы

1 Изучить нормативные документы по разработке технического задания на разработку программного продукта.

2 Разработать техническое задание на программный продукта по заданному варианту.

Техническое задание представляет собой документ, в котором сформулированы основные цели разработки, требования к программному продукту, определены сроки и этапы разработки и регламентирован процесс приемно-сдаточных испытаний. В разработке технического задания участвуют как представители заказчика, так и представители исполнителя. В основе этого документа лежат исходные требования заказчика, анализ передовых достижений техники, результаты выполнения научно-исследовательских работ, предпроектных исследований, научного прогнозирования и т. п.

Основные факторы, определяющие характеристики разрабатываемого программного обеспечения:

- исходные данные и требуемые результаты, которые определяют функции программы или системы;
- среда функционирования (программная и аппаратная); может быть задана, а может выбираться для обеспечения параметров, указанных в техническом задании;
- возможное взаимодействие с другим программным обеспечением или специальными техническими средствами - также может быть определено, а может выбираться исходя из набора выполняемых функций.

Разработка технического задания выполняется в следующей последовательности. Прежде всего, устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных. Затем определяют перечень результатов, их характеристики и способы представления.

Далее уточняют среду функционирования программного обеспечения: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, возможно, версии и параметры другого установленного программного обеспечения, с которым предстоит взаимодействовать будущему программному продукту.

В случаях, когда разрабатываемое программное обеспечение собирает и хранит некоторую информацию или включается в управление каким-либо техническим процессом, необходимо также четко регламентировать действия программы в случае сбоев оборудования и энергоснабжения.

На техническое задание существует стандарт ГОСТ 19.201-78.

«Техническое задание. Требования к содержанию и оформлению». В соответствии с этим стандартом техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;

- стадии и этапы разработки;
- порядок контроля и приемки.

При необходимости допускается в техническое задание включать приложения.

Контрольные вопросы

1. Что понимают под технологичностью программного обеспечения? Почему?
2. Какие типы программных продуктов можно выделить? Чем они различаются?
3. Назовите основные эксплуатационные требования к программным продуктам. Какими средствами и приемами обеспечивается каждый из них? Для каких типов программных систем целесообразно указывать каждый из них?
4. В каких ситуациях необходимы предпроектные исследования? Какие вопросы при этом решают? Что получают в результате таких исследований?
5. Назовите, какой раздел технического задания можно считать основным и почему? Какую информацию должны содержать остальные разделы? В чем основная сложность разработки технического задания?

Варианты заданий

1. Платформа для удаленной идентификации.
2. Платформа быстрых платежей
3. Платформа-маркетплейс для финансовых услуг и продуктов
4. Платформа для регистрации финансовых сделок
5. Система передачи финансовых сообщений
6. Создание сквозного идентификатора клиента
7. Создание платформы для облачных сервисов
8. Создание платформы на основе технологии распределенных реестров
9. Элементы новой цифровой финансовой инфраструктуры
10. Платформа для электронного взаимодействия.

Лабораторная работа 2
Тестовый цикл

Цель работы: изучение цикла исполнения тестов.

Тестовый цикл – это цикл исполнения тестов, включающий фазы 4 и 5 тестового процесса. Тестовый цикл заключается в прогоне разработанных тестов на некотором однозначно определяемом срезе системы (состоянии кода разрабатываемой системы). Обычно такой срез системы называют build.

Этапы выполнения работы

1. Проверка готовности системы и тестов к проведению тестового цикла включающая:

– Проверку того, что все тесты, запланированные для исполнения на данном цикле, разработаны и помещены в систему версионного контроля.

– Проверку того, что все подсистемы, запланированные для тестирования на данном цикле, разработаны и помещены в систему версионного контроля.

– Проверку того, что разработана и задокументирована процедура определения и создания среза системы, или build.

– Проверки некоторых дополнительных критериев.

2. Подготовка тестовой машины в соответствии с требованиями, определенными на этапе планирования (например, полная очистка и переустановка системного программного обеспечения). Конфигурация тестовой машины, так же, как и срез системы, должны быть однозначно воспроизводимыми.

3. Воспроизведение среза системы.
4. Прогон тестов в соответствии с задокументированными процедурами.
5. Сохранение тестовых протоколов (test log). *Test log* может содержать вывод системы в STDOUT, список результатов сравнения полученных при исполнении данных с эталонными или любые другие выходные данные тестов, с помощью которых можно проверить правильность работы системы.
6. Анализ протоколов тестирования и принятие решения о том прошел или не прошел каждый из тестов (Pass/Fail).
7. Анализ и документирование результатов цикла.

Последний перед выпуском продукта тестовый цикл не должен включать изменений кода build или кода продукта тестируемой системы. Этот цикл называется " финальным ". Таким образом обеспечивается ситуация, когда финальный цикл полностью повторяем, а выпускаемый продукт полностью совпадает с продуктом, который прошел тестирование. Финальный цикл необходим для гарантии достоверности результатов тестирования.

Контрольные вопросы

1. Функциональное тестирование.
2. Тестирование инсталляции.
3. Тестирование пользовательской документации
4. Стрессовое тестирование.
5. Тестирование граничных значений.
6. Тестирование производительности.
7. Тестирование на соответствие стандартам.
8. Тестирование совместимости с другими программно-аппаратными комплексами.
9. Тестирование работы с окружением.
10. Тестирование работы на конкретной платформе.

Лабораторная работа 3

Создание тестовых сценариев

Цель работы:

Получение навыков создания тестовых сценариев на основе методики управления требованиями.

Этапы выполнения работы

Процесс создания тестовых сценариев включает в себя четыре шага:

1. Определение переменных для каждого шага сценариев использования.
2. Определение существенно разных вариантов для каждой переменной.
3. Комбинирование вариантов для тестирования в тестовые сценарии.
4. Определение значений переменных.

Первым делом нужно определить все входные переменные во всех шагах в представленном сценарии (алгоритме). Например, если на некотором шаге пользователь вводит свой идентификатор и пароль, это две переменных. Первая переменная – Идентификатор, вторая – Пароль.

Количество переменных может зависеть от введенных на предыдущем шаге значений. На следующем шаге следует определить существенно различные варианты для каждой переменной. Варианты считаются «существенно разными», если они могут вызывать различное поведение системы. Например, если выбрать Идентификатор,

который предполагается быть длиной до 10-ти символов, следующие приведенные значения будут существенно разными:

Alex – слишком короткий и мы ожидаем появления сообщения об ошибке.

Alexandra – верный Идентификатор.

Alexandrena – слишком длинный, и мы ожидаем, что система не позволит нам вводить слишком длинный Идентификатор.

Однако, Alexandria или JohnGordon различны не существенно, т.к. оба являются верными идентификаторами, вызывающими одно поведение системы.

Вариант может считаться существенно другим, если:

Он вызывает другой ход процесса (обычно альтернативный поток).

Пример: Ввод неверного пароля вызовет Альтернативный Поток 2

Он вызывает другое сообщение об ошибке.

Пример: Если адрес электронной почты слишком длинный, сообщение должно быть: «E-mail должен быть не более 50-ти символов».

Пример: Если адрес электронной почты не содержит символа «@», сообщение должно быть: «Неверный E-mail адрес».

Он вызывает другой вид пользовательского интерфейса.

Пример: Если кредитная карта была выбрана в качестве способа оплаты, система должна отображать экран с полями для ввода номера кредитной карты, даты окончания срока действия и название организации, обслуживающей карту.

Он вызывает различный набор значений списков.

Пример: Экран регистрации пользователя должен содержать список Страна и Штат/Провинция. Список Штат/Провинция будет содержать значения на основе выбранной страны: для США он должен содержать все штаты, для Канады – все провинции, для других стран он должен быть недоступен.

Он является условием ограничения.

Пример: Пароль должен быть не менее 6-ти символов.

В этом случае мы должны протестировать следующее:

– Пароль с пятью символами.

– Пароль с шестью символами

– Что-то должно быть изменено вместо использования значения по умолчанию.

Пример: На экране оплаты кредитной картой название организации, обслуживающей кредитную карту, должно быть заполнено именем лица, размещающего заказ. Пользователь должен иметь возможность хранить это значение по умолчанию или ввести новое.

Это создает два различных варианта:

– Хранить установленное по умолчанию название организации, обслуживающей кредитную карту.

– Изменить установленное по умолчанию название организации на другое.

Формат ввода точно не определен и может быть интерпретирован разными способа-

Пример: поле ввода номера телефона должно принимать текст в свободной форме.

Номера телефонов разными лицами пишутся по-разному:

Использования скобок: (973) 123-4567

Использование тире: 973-123-4567

Использование пробелов: 973 123 4567

Без пробелов: 9731234567

Все доступные варианты должны быть протестированы.

Стандартные варианты могут быть разными для разных стран. Формат даты срока окончания действия кредитной карты может быть разным для США и Европы.

На следующем шаге нужно скомбинировать их в последовательность шагов тестового

сценария. Один из способов это сделать – создать Матрицу Распределения Тестовых Сценариев (Test Case Allocation Matrix). Строки этой матрицы содержат все переменные для всех шагов, требующие ввода данных он пользователя. Первая колонка содержит номер шага, вторая – название переменной, а остальные – тестовые сценарии. Их можно назвать Т1, Т2, и т.д. Нужно оценить, насколько много тестовых сценариев нужно для охвата данного сценария (алгоритма). Жестким вариантом оценки будет максимальное количество существенно различных вариантов, определенное для переменной. Не проблема, если при оценке будет допущена ошибка, т.к. можно добавить или удалить колонку при заполнении матрицы.

Обычно типичный сценарий охватывают от пяти до семи тестовых сценариев. Тем не менее, иногда в особых случаях требуется больше тестовых сценариев.

На четвертом шаге следует заменить неопределенные варианты, такие как «очень длинная фамилия» или «длинный номер телефона с ext. (дополнительным номером)» на действительные значения, например «Georgiamistopolis» и «011-48 (242) 425-3456 ext. 1234» соответственно. На этом шаге также разделяют все тестовые сценарии из Матрицы Распределения Тестовых Сценариев, создавая отдельную таблицу для каждого тестового сценария.

Метод извлечения функциональных тестовых сценариев (test cases) из сценариев использования (use cases) имеет несколько преимуществ:

– Тестовые сценарии получаются с использованием более автоматического подхода.

– Уход от дублирования тестовых сценариев.

– Достигается больший охват тестового пространства.

– Легкость мониторинга тестового процесса.

– Легкость распределения работы между тестерами.

– Легкость регрессионного тестирования.

– Раннее обнаружение пропущенных требований.

Созданные тестовые сценарии могут быть использованы для ручного тестирования, также как и для автоматического тестирования с использованием таких инструментов, как IMB Rational Robot или IBM Rational Functional Tester.

Отчет должен содержать:

– название, цель и задачи лабораторной работы;

– краткую теоретическую часть;

– переменные для основного потока сценария использования;

– значения переменных для тестирования;

– матрицу распределения тестовых сценариев;

– распределение значений переменных по тестовым сценариям;

– экранные формы;

– выводы по результатам работы.

Контрольные вопросы

1. Что такое тестирование?
2. Какие существуют типы тестов по покрытию?
3. Какие существуют типы функциональных тестов?
4. Какие существуют типы нефункциональных тестов?
5. Какие этапы составляют процесс тестирования?
6. Что происходит на этапе планирования тестирования?
7. Что происходит на этапе исполнения тестирования?
8. Какие типы тестов выполняют для первой поставки программного продукта?

9. Какие типы тестов выполняют для последующих поставок программного продукта?

Лабораторная работа 4
Тестирование пользовательского интерфейса клиентского приложения методом «черного ящика»

Цель работы: Отработать навыки составления и тестирования программного обеспечения методом «черного ящика».

Этапы выполнения работы

Известны: функции программы.

Исследуется: работа каждой функции на всей области определения.

Как показано на рисунке, основное место приложения тестов «черного ящика» – интерфейс ПО.

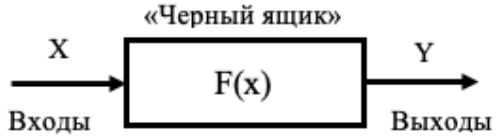

Рисунок 6.1 – Тестирование «черного ящика»

Эти тесты демонстрируют:

- как выполняются функции программ;
- как принимаются исходные данные;
- как вырабатываются результаты;
- как сохраняется целостность внешней информации.

При тестировании «черного ящика» рассматриваются системные характеристики программ, игнорируется их внутренняя логическая структура.

Методы черного ящика

1. Метод эквивалентных разбиений

Его основу составляют 2 положения:

– Каждый тип должен включать столько различных входных и выходных условий, сколько необходимо для того, чтобы минимизировать общее число необходимых тестов.

– Необходимо разбивать входную область программы на конечное число классов эквивалентности

Класс эквивалентности выделяется путем выбора каждого входного условия и разбиением его на две или более групп.

2. Анализ граничных решений (АГР)

Граничные условия – ситуация, возникающая непосредственно на границе, выше или ниже границ входных или выходных элементов класса эквивалентности (КЭ)

АГР предполагает:

– Выбор любого элемента в КЭ в качестве представительного при АГЗ осуществляется таким образом, чтобы проверить тестом каждую границу этого класса.

– При разработке тестов рассматриваются не только входные условия, но и выходные.

3. Метод функциональных диаграмм

Недостатком метод граничных решений и метод эквивалентных разбиений является то, что они не исследуют комбинации входных условий.

Метод функциональных диаграмм помогает создать высоко результирующие тесты.

Функциональная диаграмма представляет собой формальный язык, на который транслируется спецификация, написанная на естественном языке.

Построение тестов осуществляется в несколько этапов:

– Спецификация разбивается на несколько участков
– Спецификация определяется причиной и следствием

Причины и следствия определяются путем последовательного чтения спецификации, каждой причине и следствию присваивается отдельный номер.

Графы причинно-следственных связей

Диаграммы причинно-следственных связей используются для проектирования тестовых вариантов и обеспечивают формальную запись логических условий и соответствующих действий. Данный способ является разновидностью тестирования «черного ящика». Используется автоматный подход к решению задачи.

На первом шаге способа тестирования, основанного на построении диаграмм причинно-следственных связей, для тестируемой программы (или отдельного тестируемого модуля) перечисляются причины (условия ввода или классы эквивалентности условий ввода) и следствия (действия или условия вывода). Каждой причине и следствию присваивается свой идентификатор.

На втором шаге данного способа тестирования разрабатывается **граф причинно-следственных связей**.

Введем нотацию базовых символов для записи графов причин и следствий. Причины будем обозначать символами c_i, а следствия — символами e_i. Каждый узел графа может находиться в состоянии 0 (состояние отсутствует) или 1 (состояние присутствует).

Функция «тождество» (рис. 6.2) устанавливает, что если значение есть 1, то и значение есть 1. В противном случае значение есть 0.

Рисунок 6.2 – Функция «тождество»

Функция «не» (рис. 6.3) устанавливает, что если значение c_1 есть 1, то значение e_1 есть 0. В противном случае значение есть 1.

Рис. 6.3 Функция «не»

Функция «или» (рис. 6.4) устанавливает, что если c_1 или c_2 есть 1, то e_1 есть 1. В противном случае e_1 есть 0.

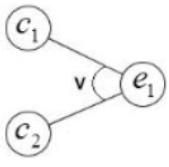

Рис. 6.4 Функция «или»

Функция «и» (рис. 6.5) устанавливает, что если и c_1, и c_2 есть 1, то e_1 есть 1. В противном случае есть 0.

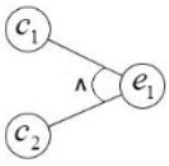

Рис. 6.5 Функция «и»

На третьем шаге рассматриваемого способа тестирования граф преобразуется в таблицу решений. Порядок генерации таблицы решений [1]:

1) Выбирается некоторое следствие, которое должно быть в состоянии «1».

2) Находятся все комбинации причин (с учетом ограничений), которые устанавливают это следствие в состояние «1». Для этого из следствия прокладывается обратная трасса через граф.

3) Для каждой комбинации причин, приводящих следствие в состояние «1», строится один столбец.

4) Для каждой комбинации причин доопределяются состояния всех других следствий. Они помещаются в тот же столбец таблицы решений.

5) Действия 1–4 повторяются для всех следствий графа. На четвертом шаге данного способа тестирования столбцы таблицы решений преобразуются в тестовые варианты.

Задание

Выполнение работы предусматривает следующую последовательность действий:

1. Определение причин (условий ввода) и следствий;
2. Построение графа причинно-следственных связей;
3. Создание таблиц решений;
4. Построение тестовых вариантов;
5. Оформление результатов тестирования.

Лабораторная работа 5

Реализация поддержки проверки данных на стороне клиента для [Equal]

Цель работы:

Получение навыков создания отдельного программного кода, написанного на языке JavaScript для проверки данных на стороне клиента [Equal].

Этапы выполнения работы

1. Интерфейс для атрибутов

Для пересылки своих параметров в JavaScript код атрибут должен поддерживать интерфейс *IClientValidatable*, определенный в пространстве имен System.Web.Mvc. При это необходимо реализовать всего лишь один метод *GetClientValidationRules()*, который возвращает перечень правил. Каждое правило включает список значений для настройки функции проверки и соответствующего сообщения об ошибке.

Откроем исходный код класса *EqualAttribute* и добавим поддержку указанного интерфейса. С помощью редактора Visual Studio создадим заготовку его метода и добавим в неё следующий код:

```
1       namespace MVCDemo.Attributes.Validation
2       {
3           using System;
4           using System.Collections.Generic;
5           using System.ComponentModel.DataAnnotations;
6           using System.Web.Mvc;
7
8           [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = true)]
9           public class EqualAttribute : ValidationAttribute, IClientValidatable
1           {
1               private readonly object _valueToCompare;
1
1               .....
```

```
1
1           #region IClientValidatable Members
1
1           public IEnumerable<ModelClientValidationRule> GetClientValidationRules(
1              ModelMetadata metadata, ControllerContext context)
1           {
2              var rule = new ModelClientValidationRule() {
                  ErrorMessage = this.FormatErrorMessage(metadata.DisplayName),
                  ValidationType = "equal"
               };

               rule.ValidationParameters.Add("valuetocompare", this._valueToCompare);

               yield return rule;
            }

            #endregion
         }
      }
```

Теперь подробно рассмотрим реализацию метода *GetClientValidationRules()*.

Первый параметр данный метод содержит метаданные свойства, для которого назначен атрибут. В частности, из него можно получить отображаемое имя свойства, которое будет использовано для создания текста сообщения об ошибке. Вторым параметром является контекст Контроллера, включающий его данные и значения из исходного запроса.

Поскольку в данном случае только одно правило, то создается перечисление из единственного экземпляра *ModelClientValidationRule*. В нем сохраняются:

– *ErrorMessage* – сообщение об ошибке.

– *ValidationType* – уникальное имя данного правила (запомним его, т.к. оно не раз будет использовано в клиентском коде).

– *ValidationParameters* – коллекция пар "ключ- значение", в которой сохраняются необходимые для передачи в JavaScript параметры (имена ключей также потребуются в коде на JavaScript).

Таким образом, в результате вызова данной реализации *GetClientValidationRules()* будет создано правило "equal", содержащее значение для сравнения "valuetocompare" и строку с текстом сообщения об ошибке.

Каркас ASP.NET MVC 3 сохранит эти значения в HTML теге соответствующего поля в следующем виде:

```
1     <input name="AgreementAccepted" class="check-box"
2     id="AgreementAccepted" type="checkbox"
3     data-val="true"
4     data-val-required="The I have read the EULA and I agree with it field is required."
5     data-val-equal-valuetocompare="True"
6     data-val-equal="You must agree with the EULA."
7     value="true"/>
```

Обратите внимание, имя правила используется для создания названий атрибутов HTML тегов. Сообщение об ошибке при этом располагается в "data-val-[имя-правила]", а значения параметров в "data-val-[имя-правила]-[ключ]". Кроме того, в именах правил и названиях ключей можно использовать только строчные буквы без пробелов и других символов.

2. Создаем JavaScript с алгоритмом проверки.

Все готово для того, чтобы программа на JavaScript смогла получить данные и передать их в функцию проверки. Но перед тем как продолжить, давайте выберем место хранения для файлов кодом проверок на JavaScript. Для этого в папке Scripts создадим папку Validation. Кроме того, называть все создаваемые файлы будем по аналогии с файлами, содержащими исходный код атрибутов.

Чтобы в очередной раз не изобретать колесо, ядро ASP.NET MVC 3 использует библиотеку jQuery для клиентской части веб-приложения. С её помощью реализован функционал проверки в браузере свойств, отмеченных стандартными атрибутами. Не будем углубляться в тонкости её работы и рассмотрим только необходимые для решаемой задачи детали.

Проверка данных на стороне клиента выполняется дополнением jQuery Validation plugin. Его работа сводится к следующему: при изменении поля необходимо вызвать функцию проверки значения и, в случае ошибки, вывести текст полученного сообщение на соответствующе место страницы. Данный механизм доступен через объект *jQuery.validator*. Используя его функцию *addMethod()* необходимо добавить новое правило проверки в их список.

В результате, в папке Scripts\Validation создадим файл EqualAttribute.js, содержащий следующий код:

```
1   /// <reference path="../jquery-1.4.4-vsdoc.js" />
2   /// <reference path="../jquery.validate-vsdoc.js" />
3   /// <reference path="../jquery.validate.unobtrusive.js" />
4
5   jQuery.validator.addMethod("equal", function (value, element, param) {
6       if ($(element).attr("type") == "checkbox") {
7           value = String($(element).attr("checked"));
8           param = param.toLowerCase();
9       }
1
1       return (value == param);
1   });
1
1   jQuery.validator.unobtrusive.adapters.addSingleVal("equal", "valuetocompare");
```

Первые три строчки с комментариями необходимы только поддержки IntelliSense и указывают на используемые файлы с описаниями функций.

Затем, с помощью вызова *addMethod(),* в список проверок добавляется новое правило. Обратите внимание, что указывается тоже самое имя, которое было задано в атрибуте. Это обязательно, за исключением случаев, когда правило содержит не более одного параметра. Причина этого заключена в адаптерах и будет рассмотрена в следующей части.

Последним параметром *addMethod()* является функция, ответственная за проверку значения. При вызове она получает три параметра:

- *value* – текущее значение поля ввода;
- *element* – элемент страницы;
- *param* – список параметров, переданный атрибутом проверки данных.

Код созданной функции проверки сравнивает введенное значение с заданным в *param*. Данная реализация учитывает особенность объектов представляющих checkbox, которые содержат значение не в HTML атрибуте value, а в checked. Метод при успешной проверке возвращает *true*, в противном случае – *false*.

jQuery Validation обработает значение, полученное от функции проверки. Если оно будет равно *false*, то на месте блока , созданного методом *ValidationMessageFor()* и отмеченного при этом специальным HTML атрибутом, будет выведен текст сообщения об ошибке. Оно будет убрано в дальнейшем, после успешного прохождения проверки.

3. Адаптер для jQuery Validate

Рассматривая код функции проверки может возникнуть вопрос: а откуда возьмётся значение для сравнения в переменой *param*? Логично предположить, что оно будет передано из jQuery Validation. Но откуда тогда сама библиотека возьмет это значение и текст сообщения об ошибке?

Для этой цели существуют адаптеры, которые считывают значения из атрибутов HTML тегов и передают их в *jQuery.validator*. Они располагаются в объекте *jQuery.validator.unobtrusive.adapters* и являются дополнением для jQuery Validation, созданным разработчиками ASP.NET MVC 3.

Посмотрим еще раз на последнюю строку созданного JavaScript:

```
jQuery.validator.unobtrusive.adapters.addSingleVal("equal", "valuetocompare");
```

Метод *addSingleVal()* используется для адаптации правил с одним значением. Имя адаптера указано в качестве первого и совпадает с именем правила. Затем следует название параметра (ключ из заданной в C# коде пары), значение которого будет передано как переменная *param* в созданную выше функцию.

Все правила используют только одно значение. Существуют другие вариант добавления адаптеров, которые будет рассмотрены в следующей части.

Подключение JavaScript в Представление

Остался один шаг: необходимо подключить созданный JavaScript на страницу создания профиля. Это можно сделать вручную добавив соответствующую строку в Представление UserProfiles\Create. Или же перетащить в него файл EqualAttribute.js из окна Solution Navigator (Solution Explorer). Для указания пути воспользуемся вызовом метода *Url.Content()*. Результат будет следующим:

```
@model MVCDemo.Models.NewUserProfileModel
@using MVCDemo.Resources.Views.UserProfiles;
@{
   ViewBag.Title = CreateRes.PageTitle;
}

<h2>@ViewBag.Title</h2>

<script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/javascript"></script>
<script
src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/Validation/EqualAttribute.js")" type="text/javascript"></script>

@using (Html.BeginForm()) {
.....
```

Теперь можно запустить веб-приложение посмотреть на новую функциональность в работе. Приступим к разработке аналогичной поддержки проверки на стороне клиента и для оставшегося атрибута.

Контрольные вопросы

1. Тестирование HTML/JavaScript-инъекций
2. Тестирование SQL-инъекций
3. Тестирование XSS-уязвимостей
4. Тестирование инъекций в HTTP заголовках

5. Тестирование переадресаций
6. Тестирование инъекций команд ОС
7. Тестирование уязвимости Pah Traversal

Лабораторная работа 6

Реализация поддержки проверки данных на стороне клиента для атрибута [StringLengthRange]

Цель работы:

Получение навыков создания отдельного программного кода, написанного на языке JavaScript для проверки данных на стороне клиента для атрибута [StringLengthRange].

Этапы выполнения работы

Реализация атрибута без создания адаптера.

Добавим классу *StringLengthRangeAttribute* поддержку интерфейса *IClientValidatable*. Здесь необходимо отметить, что jQuery Validation уже содержит набор часто используемых правил. Для их использования в ASP.NET MVC 3 существуют специальные версии классы, наследники *ModelClientValidationRule()*.

```
1   namespace MVCDemo.Attributes.Validation
2   {
3       using System;
4       using System.Collections.Generic;
5       using System.ComponentModel.DataAnnotations;
6       using System.Web.Mvc;
7
8       [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = true)]
9       public class StringLengthRangeAttribute : ValidationAttribute, IClientValidatable
1       {
1           private readonly int _minLength;
1           private readonly int _maxLength;
1
1           .....
1
            #region IClientValidatable Members

1           public IEnumerable<ModelClientValidationRule> GetClientValidationRules(
1               ModelMetadata metadata, ControllerContext context)
2           {
2               yield return new ModelClientValidationStringLengthRule(
2                   this.FormatErrorMessage(metadata.DisplayName),
2                   this._minLength,
2                   this._maxLength);
2           }
2
2           #endregion
2       }
2   }
```

В данном случае нет необходимости в создании клиентской части на JavaScript и добавлении ссылки на неё в Представление.

Использование *ModelClientValidationStringLengthRule()* позволяет использовать уже готовые адаптер и реализацию алгоритма проверки длины строки.

Использование готовых правил позволяет избежать их двойного применения к объекту. Дело в том, что атрибуты с одинаковыми стандартными правилами будут использовать их одинаковые имена. А это привет к ошибке в момент обращения к Представлению на этапе выполнения.

Классы ASP.NET MVC для стандартных правил

– *ModelClientValidationEqualToRule* – устанавливает равенство значений текущего и указанного поля.

– *ModelClientValidationRangeRule* – определяет минимальное и максимальное значения свойства.

– *ModelClientValidationRegexRule* – проверяет значение на соответствие регулярному выражению. Используется стандартным атрибутом *[RegularExpression]*.

– *ModelClientValidationRemoteRule* – используется для удаленной проверки данных. Используется стандартным ASP.MET MVC 3 атрибутом *[Remote]*.

– *ModelClientValidationRequiredRule* – указывает что данное поле обязательно к заполнению. Используется стандартным атрибутом *[Required]*.

– *ModelClientValidationStringLengthRule* – ограничивает длину заданной строки.

Контрольные вопросы

1. Тестирование RFI и LFI
2. Тестирование SMTP-инъекций
3. Тестирование SOAP-инъекций
4. Тестирование LDAP-инъекций
5. Тестирование XPath-инъекций
6. Тестирование XXE-инъекций
7. Тестирование внедрения шаблона

Задание для выполнения курсовой работы

Провести комплексное тестирование программного продукта по следующему плану:

1. Анализ бизнес-требований.
2. Разработка плана тестирования.
3. Разработка тестовых сценариев и кейсов.
4. Подготовка тестовых данных.
5. Реализация тестов.
6. Анализ результатов тестирования. Достижение бизнес-цели.

Варианты

1. Платформа для удаленной идентификации.
2. Платформа быстрых платежей
3. Платформа-маркетплейс для финансовых услуг и продуктов
4. Платформа для регистрации финансовых сделок
5. Система передачи финансовых сообщений
6. Создание сквозного идентификатора клиента
7. Создание платформы для облачных сервисов
8. Создание платформы на основе технологии распределенных реестров
9. Платформа для электронного взаимодействия.

Приложение 2

Оценочные средства для проведения промежуточной аттестации

а) Планируемые результаты обучения и оценочные средства для проведения промежуточной аттестации:

Код индикатора	Индикатор достижения компетенции	Оценочные средства
ОПК-5 Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем;		
ОПК-5.1	Определяет необходимость и участвует в разработке и модернизации программного и аппаратного обеспечение информационных и автоматизированных систем	***Перечень теоретических вопросов*** Сканирование портов. Отображение видимого контента. Поиск скрытого контента. Поиск параметров отладки и разработки. Определение точек ввода данных. Определение используемых технологий. Отображение возможных векторов атаки. Определение правил стойкости пароля. Тестирование подбора логина. Тестирование подбора пароля. Тестирование восстановления аккаунта. Тестирование функции «Запомнить меня». Тестирование функции идентификации пользователя. Проверка распределения полномочий. Проверка уникальности логина. Тестирование многоступенчатых механизмов. Проверка токенов на предсказуемость. Проверка безопасности передачи токенов. Проверка отображения токенов в логах. Проверка многократного использования токенов. Проверка завершения сеанса. Проверка фиксации сессии. Тестирование уязвимости CSRF. Определение требований контроля доступа. Тестирование эффективности многопользовательского управления. Тестирование незащищённого доступа к методам управления.

Код индикатора	Индикатор достижения компетенции	Оценочные средства
		Фаззинг всех параметров. Тестирование SQL-инъекций. Тестирование XSS-уязвимостей. Тестирование инъекций в HTTP заголовках. Тестирование переадресаций. Тестирование инъекций команд ОС. Тестирование уязвимости Pah Traversal. Тестирование HTML/JavaScript-инъекций. Тестирование RFI и LFI. Тестирование SMTP-инъекций. Тестирование SOAP-инъекций. Тестирование LDAP-инъекций. Тестирование XPath-инъекций. Тестирование XXE-инъекций. Тестирование внедрения шаблона. Определение векторов атаки. Тестирование передачи данных на стороне клиента. Тестирование валидации данных на стороне клиента. Тестирование компонентов толстых клиентов. Тестирование логики многоступенчатых механизмов. Тестирование обхода аутентификации. Тестирование прав доступа. Тестирование логики транзакций. Тестирование IDOR-уязвимостей. Тестирование разделения в среде виртуального хостинга. Тестирование разделения между ASP-приложениями. Тестирование уязвимостей на сервере. Проверка стандартных учётных записей. Определение стандартного контента на сайте. Определение опасных HTTP-методов. Тестирование прокси. Тестирование DOM-модели. Тестирование frame-инъекций.

Код индикатора	Индикатор достижения компетенции	Оценочные средства
		Проверка локальных уязвимостей. Проверка параметров cookies. Определение конфиденциальных данных в URL-параметрах. Проверка наличия слабых SSL-шифров. Анализ HTTP-заголовков. ***Практические задания*** Альфа/бета-тестирование. Контрактное приемочное тестирование. Законодательное приемочное тестирование. Операционное приемочное тестирование. Тестирование по стратегии черного ящика. ***Задания на решение задач из профессиональной области, комплексные задания*** *Провести комплексное тестирование программного продукта по следующему плану:* Анализ бизнес-требований. Разработка плана тестирования. Разработка тестовых сценариев и кейсов. Подготовка тестовых данных. Реализация тестов. Анализ результатов тестирования. Достижение бизнес-цели. *Варианты* Платформа для удаленной идентификации. Платформа быстрых платежей Платформа-маркетплейс для финансовых услуг и продуктов Платформа для регистрации финансовых сделок Система передачи финансовых сообщений Создание сквозного идентификатора клиента Создание платформы для облачных сервисов Создание платформы на основе технологии распределенных реестров Платформа для электронного взаимодействия.

б) Порядок проведения промежуточной аттестации, показатели и критерии оценивания:

Промежуточная аттестация включает теоретические вопросы, позволяющие оценить уровень усвоения обучающимися знаний, и практические задания, выявляющие степень сформированности умений и владений, проводится в форме зачета.

Зачет по дисциплине проводится по результатам отчетности на практических занятиях с опросом в устной форме по этапам выполнения и активного выступления в беседе-обсуждении на лекционных занятиях.

Показатели и критерии для зачета:

– на оценку **«зачтено»** – обучающийся демонстрирует уровень сформированности компетенций, знание учебного материала, свободно выполняет практические задания, свободно оперирует знаниями, умениями, применяет их в различных ситуациях.

– на оценку **«не зачтено» -** обучающийся не может показать знания на уровне воспроизведения и объяснения информации, не может показать интеллектуальные навыки решения простых задач.