КАЗАНСКИЙ КООПЕРАТИВНЫЙ ИНСТИТУТ (ФИЛИАЛ)
АВТОНОМНОЙ НЕКОММЕРЧЕСКОЙ ОБРАЗОВАТЕЛЬНОЙ
ОРГАНИЗАЦИИ ВЫСШЕГО ОБРАЗОВАНИЯ
ЦЕНТРОСОЮЗА РОССИЙСКОЙ ФЕДЕРАЦИИ
«РОССИЙСКИЙ УНИВЕРСИТЕТ КООПЕРАЦИИ»

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ

ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Специальность: 10.05.01 Компьютерная безопасность
Специализация: «Разработка защищенного программного обеспечения»
Формы обучения: очная
Объем дисциплины:
в зачетных единицах: 3 з.е.
в академических часах: 108 ак.ч.

Форма промежуточной аттестации: зачет

2025

Оценочные материалы по дисциплине Технологии проектирования программного обеспечения

обсуждены и рекомендованы к утверждению решением кафедры гуманитарных дисциплин и иностранных языков от 21 марта 2025 г., протокол № 7

одобрены Научно-методическим советом Казанского кооперативного института (филиала) от «2» апреля 2025 г., протокол № 3.

СОДЕРЖАНИЕ

Целью оценочных материалов по дисциплине (модулю) (далее –ОМ) является комплексная оценка результатов обучения по дисциплине Технологии проектирования программного обеспечения и установление уровня сформированности компетенций обучающегося.

ОМ предназначены для проведения текущего контроля успеваемости и промежуточной аттестации.

ОМ включают оценочные средства для проведения текущего контроля успеваемости и промежуточной аттестации.

Оценочные материалы разработаны в соответствии с рабочей программой дисциплины Технологии проектирования программного обеспечения.

1. Паспорт оценочных материалов по дисциплине «Технологии проектирования программного обеспечения»

Формируемые компетенции (код и наименование компетенции)	Индикаторы достижения компетенций	Планируемые результаты обучения	Контролируемые разделы, темы дисциплины	Наименование оценочного средства[1]	
				Текущий контроль	Промежуточная аттестация
1	2	3	4	5	6
ПК-3.2 Способен участвовать в проектировании программных и аппаратных средств защиты информации	ПК-3.2.1 Знает подходы к проектированию программных и аппаратных средств защиты информации	Знать: основы алгоритмизации и программирования Принципы построения и виды архитектуры компьютерного программного обеспечения при разработке мобильных приложений в среде Android Studio / XCode Стандарты оформления кода для используемых языков программирования на языке Java / Kotlin / Swif Уметь: применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов при разработке мобильных приложений в среде Android Studio / XCode Владеть: проектирование программных интерфейсов при разработке мобильных приложений в среде Android Studio / XCode Проектирование структур данных в среде Android Studio / XCode Разработка технической документации на компьютерное программное обеспечение с использованием существующих стандартов при разработке мобильных приложений	Тема 1. Методология DevOps для создания ПО Тема 2. Управление требованиями к ПО Тема 3. Управление дизайном ПО Тема 4. Управление версиями ПО Тема 5. Управление качеством ПО Тема 6. Управление сборками ПО Тема 7. Управление развертыванием ПО	Реферат (Тема 1-8) Тест (тема 1-8)	Тест (П 1-10 тестового задания)

			Тема 8. Совершенствование программного процесса		
	ПК-3.2.2 Умеет разрабатывать предложения по совершенствованию существующих методов и средств, применяемых для контроля и защиты информации и повышения эффективности этой защиты	Знать: основы алгоритмизации и программирования Стандарты оформления кода для используемых языков программирования Java / Kotlin / Swift Типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения при разработке мобильных приложений в среде Android Studio / XCode Уметь: применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов при разработке мобильных приложений в среде Android Studio / XCode Владеть: проектирование программных интерфейсов при разработке мобильных приложений Проектирование структур данных при разработке мобильных приложений в среде Android Studio / XCode	Тема 1. Методология DevOps для создания ПО Тема 2. Управление требованиями к ПО Тема 3. Управление дизайном ПО Тема 4. Управление версиями ПО Тема 5. Управление качеством ПО Тема 6. Управление сборками ПО Тема 7. Управление развертыванием ПО Тема 8. Совершенствование программного процесса	Реферат (Тема 1-8) Тест (Тема 1-8) Практическое задание (Тема 1-8)	Тест (П 11-20 тестового задания)

	ПК-3.2.3 Владеет методами и средствами разработки архитектуры и интерфейсов систем защиты информации, процедуры восстановления работоспособности и средств и систем защиты после сбоев	Знать: стандарты оформления кода для используемых языков программирования Java / Kotlin / Swift Типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения при разработке мобильных приложений в среде Android Studio / XCode Уметь: применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов при разработке мобильных приложений в среде Android Studio / XCode Владеть: проектирование программных интерфейсов при разработке мобильных приложений в среде Android Studio / XCode Разработка, изменение архитектуры компьютерного программного обеспечения и ее согласование с системным аналитиком и архитектором программного обеспечения при разработке мобильных приложений в среде Android Studio / XCode	Тема 1. Методология DevOps для создания ПО Тема 2. Управление требованиями к ПО Тема 3. Управление дизайном ПО Тема 4. Управление версиями ПО Тема 5. Управление качеством ПО Тема 6. Управление сборками ПО Тема 7. Управление развертыванием ПО Тема 8. Совершенствование программного процесса	Реферат (Тема 1-8) Тест (Тема 1-8) Практическое задание (Тема 1-8)	Тест (П 21-30 тестового задания)

2. Оценочные материалы по дисциплине «Технологии проектирования программного обеспечения»

2.1 Оценочные материалы для проведения текущего контроля успеваемости

Тематика рефератов

ТЕМА 1. МЕТОДОЛОГИЯ DEVOPS ДЛЯ СОЗДАНИЯ ПО

1. **Культура DevOps: от противостояния к сотрудничеству.** Роль общих ценностей, доверия и размытия границ между отделами в успешной реализации DevOps.
2. **Инструментальная цепочка DevOps: полный обзор.** Классификация и сравнительный анализ инструментов на каждом этапе (Plan, Code, Build, Test, Release, Deploy, Operate, Monitor).
3. **Три основы (Three Ways) DevOps: философия потоков, обратной связи и непрерывного эксперимента.** Глубокий анализ принципов Джина Кима и их практическая интерпретация.
4. **Непрерывная интеграция (CI) как фундамент DevOps.** Принципы, лучшие практики и влияние на скорость и качество разработки.
5. **Непрерывное развертывание (CD) и доставка (CD): путь к самообновляемому продукту.** Методы автоматизации выпуска ПО и минимизации рисков при развертывании.
6. **DevSecOps: интеграция безопасности в жизненный цикл DevOps.** Как реализовать принцип "безопасность как код" на всех этапах разработки.
7. **Контейнеризация и оркестрация (Docker, Kubernetes) как драйверы DevOps-революции.** Их роль в обеспечении переносимости, масштабируемости и надежности приложений.
8. **Мониторинг и обратная связь в DevOps.** Роль Observability (логи, метрики, трейсы) в создании циклов непрерывного улучшения продукта.
9. **Site Reliability Engineering (SRE) как реализация принципов DevOps в эксплуатации.** Сравнение моделей SRE и DevOps, управление ошибками и надежностью.
10. **Измерение эффективности DevOps: DORA metrics и не только.** Ключевые метрики (Lead Time, Deployment Frequency, MTTR и др.) для оценки успешности трансформации.

ТЕМА 2. УПРАВЛЕНИЕ ТРЕБОВАНИЯМИ К ПО

1. **Классификация требований к ПО: функциональные, нефункциональные и бизнес-требования.** Особенности сбора, анализа и документирования для каждого типа.
2. **Свойства хороших требований (SMART и не только).** Анализ характеристик: однозначность, полнота, непротиворечивость, проверяемость, трассируемость.
3. **Техники сбора и выявления требований: интервью, мозговой штурм, пользовательские истории, Use Cases.** Сравнительный анализ и области применения.
4. **Формализация требований: от пользовательских историй к техническим спецификациям.** Языки моделирования (UML, BPMN) и их роль в документировании.
5. **Жизненный цикл требования: от идеи до реализации и валидации.** Управление изменениями и приоритезация на протяжении всего цикла.
6. **Типичные ошибки при документировании требований и методы их**

предотвращения. Анализ "антипаттернов" и рекомендации по созданию качественной документации.
7. **Роль менеджера по продукту (Product Manager) и бизнес-аналитика в управлении требованиями.** Разграничение ответственности и зон взаимодействия.
8. **Профессиональные и этические требования к инженеру-программисту.** Анализ кодекса этики IEEE/ACM и его применение в повседневной работе.
9. **Управление нефункциональными требованиями (NFR): производительность, безопасность, надежность.** Сложности формализации, тестирования и обеспечения выполнения.
10. **Инструменты для управления требованиями (Jira, Confluence, IBM DOORS и др.).** Обзор возможностей и критерии выбора подходящего инструмента.

ТЕМА 3. УПРАВЛЕНИЕ ДИЗАЙНОМ ПО

1. **Эволюция методов проектирования ПО: от нисходящего проектирования к Agile-моделированию.** Исторический обзор и современные тенденции.
2. **Функционально-ориентированное проектирование: методологии SADT/IDEF0 и структурные схемы.** Принципы, области применения и ограничения.
3. **Объектно-ориентированный анализ и проектирование (OOAD).** Принципы SOLID, GRASP и их роль в создании гибкого и поддерживаемого дизайна.
4. **Архитектурные стили и паттерны: многоуровневая архитектура, MVC, микросервисы, Event-Driven.** Сравнительный анализ и критерии выбора.
5. **Роль UML в управлении дизайном ПО.** Ключевые диаграммы (Use Case, Class, Sequence, Activity) и их практическое применение.
6. **Современные CASE-средства: от визуального моделирования до генерации кода.** Обзор рынка (Enterprise Architect, Visual Paradigm, Lucidchart) и их возможности.
7. **Паттерны проектирования (GoF): порождающие, структурные, поведенческие.** Классификация, примеры применения и влияние на качество кода.
8. **Принципы Domain-Driven Design (DDD) в управлении сложными доменами.** Стратегирующее и тактическое проектирование, ubiquitous language.
9. **Аспектно-ориентированное программирование (AOP) как метод отделения сквозной функциональности.** Роль в управлении дизайном и улучшении модульности.
10. **Документирование архитектуры ПО: решения, контекст и компоненты.** Роль архитектурных решений (ADR) и диаграмм C4 в управлении дизайном.

ТЕМА 4. УПРАВЛЕНИЕ ВЕРСИЯМИ ПО

1. **Эволюция систем контроля версий: от локальных (SCCS) к централизованным (SVN) и распределенным (Git).** Сравнительный анализ архитектур и возможностей.
2. **Git как стандарт индустрии: глубокий анализ внутреннего устройства.** Объекты в Git (blob, tree, commit, tag) и механика работы.
3. **Модели ветвления в Git: GitFlow, GitHub Flow, GitLab Flow.** Сравнительный анализ, преимущества, недостатки и области применения.
4. **Конфигурационное управление (Software Configuration Management - SCM): цели, задачи и основные понятия.** Управление версиями не только кода, но и артефактов.
5. **Понятие Baseline в управлении конфигурациями.** Роль базовых линий в стабилизации проекта, управлении выпусками и оценке изменений.
6. **Управление зависимостями как часть конфигурационного управления.** Инструменты (Maven, Gradle, NuGet) и практики для обеспечения воспроизводимости сборок.
7. **Стратегии тегирования (версионирования) артефактов: Semantic Versioning**

(SemVer). Принципы присвоения версий и их значение для управления зависимостями.
8. **Роль Code Review в распределенных системах контроля версий (на примере Pull/Merge Request в Git).** Как практики контроля версий влияют на качество кода и обмен знаниями.
9. **Инструменты управления версиями для артефактов: бинарные репозитории (Artifactory, Nexus).** Их роль в жизненном цикле DevOps и управлении зависимостями.
10. **Проблемы управления большими двоичными файлами (Large File Storage - LFS) в Git и методы их решения.** Обзор инструментов и практик.

ТЕМА 5. УПРАВЛЕНИЕ КАЧЕСТВОМ ПО

1. **Стандарты качества ПО: ISO 25010, ISO 9000.**
2. **Статический анализ кода (SAST) как метод контроля качества.** Инструменты (SonarQube, Checkstyle, ESLint) и их интеграция в процесс разработки.
3. **Системы отслеживания ошибок (Bug Tracking Systems): Jira, Bugzilla, YouTrack.** Обзор функциональности, workflows и роль в управлении качеством.
4. **Признаки "плохого запаха" кода (Code Smells) и рефакторинг как метод борьбы с ними.** Каталог "запахов" Мартина Фаулера и соответствующие приемы рефакторинга.
5. **Принципы F.I.R.S.T. для написания чистых тестов.** Детальный разбор каждого принципа (Fast, Independent, Repeatable, Self-Validating, Timely) и их важность.
6. **Пирамида тестирования: стратегия распределения усилий по уровням тестирования.**
7. **Модели автоматизированного тестирования: Page Object Model (POM), Screenplay Pattern.** Сравнительный анализ подходов к организации тестового кода.
8. **Нефункциональное тестирование: нагрузочное, стрессовое, тестирование безопасности.** Методы, инструменты и критерии качества.
9. **Test Driven Development (TDD) и Behavior Driven Development (BDD) как методологии управления качеством на этапе разработки.** Сравнение принципов и практик.
10. **Непрерывное тестирование (Continuous Testing) в конвейере CI/CD.** Роль автоматизированных тестов в обеспечении быстрой обратной связи и снижении рисков.

ТЕМА 6. УПРАВЛЕНИЕ СБОРКАМИ ПО

1. **Эволюция систем сборки: от Make к современным инструментам (Maven, Gradle, MSBuild).** Сравнительный анализ подходов и возможностей.
2. **Apache Maven: концепция Convention over Configuration, жизненный цикл сборки и управление зависимостями.**
3. **Gradle как современная система сборки: преимущества Groovy/Kotlin DSL, инкрементальные сборки и производительность.**
4. **Управление артефактами сборки: создание, версионирование и публикация в репозитории.**
5. **Непрерывная интеграция (CI): принципы, практики и инструменты (Jenkins, GitLab CI, GitHub Actions).**
6. **Создание воспроизводимых и переносимых сборок с использованием контейнеризации (Docker).**
7. **Мультиплатформенные сборки: особенности и инструменты для кроссплатформенной разработки.**
8. **Инкрементальные и полные сборки: стратегии оптимизации времени сборки.**
9. **Управление конфигурациями сборки для разных сред (development, staging,**

production).

10. **Мониторинг и метрики процесса сборки: анализ длительности, успешности и "здоровья" CI-пайплайна.**

ТЕМА 7. УПРАВЛЕНИЕ РАЗВЕРТЫВАНИЕМ ПО

1. **Эволюция стратегий развертывания: от "Big Bang" к Canary и Blue-Green.** Сравнительный анализ рисков и времени простоя.
2. **Инфраструктура как код (IaC): инструменты (Terraform, Ansible) и их роль в автоматизации и стандартизации развертывания.**
3. **Контейнеризация и оркестрация (Kubernetes) как стандарт для управления развертыванием микросервисов.**
4. **Непрерывное развертывание (Continuous Deployment): полная автоматизация выпуска ПО от коммита до продакшена.**
5. **Feature Flags (функциональные флаги) как техника управления рисками и проведения A/B тестирования при развертывании.**
6. **Проблемы внедрения ПО: сопротивление пользователей, миграция данных, обратная совместимость.** Методы их преодоления (обучение, фазированный rollout).
7. **Управление конфигурациями runtime-окружения: от переменных окружения до специализированных систем (Apache ZooKeeper, Consul).**
8. **Мониторинг и observability в эксплуатации: использование метрик, логов и трейсов для оперативного управления системой.**
9. **Процедуры отката (Rollback) и аварийного восстановления (Disaster Recovery).** Планирование и автоматизация действий на случай сбоя при развертывании.
10. **Управление рисками эксплуатации ПО: проактивный мониторинг, анализ инцидентов и создание resilient-систем.**

ТЕМА 8. СОВЕРШЕНСТВОВАНИЕ ПРОГРАММНОГО ПРОЦЕССА

1. **Модель зрелости процессов разработки ПО (CMMI): история, структура и пять уровней зрелости.**
2. **Области процессов (Process Areas) в CMMI: детальный разбор на примере уровня 2 (Управляемый).**
3. **Сравнительный анализ моделей CMMI и стандарта ISO 9001: общие черты и различия в подходах к качеству процессов.**
4. **Модель IDEAL (Initiating, Diagnosing, Establishing, Acting, Learning) как дорожная карта для улучшения процессов.**
5. **Метрики и измерения для оценки эффективности программных процессов (Productivity, Cycle Time, Defect Density).**
6. **Agile-модели зрелости и улучшения процессов (например, Scrum Maturity Model).**
7. **Роль ретроспектив в рамках Agile/Scrum как инструмента непрерывного улучшения процессов на уровне команды.**
8. **Фреймворк ITSM и библиотека ITIL: применение лучших практик управления IT-услугами для совершенствования процессов эксплуатации ПО.**
9. **Культура непрерывного улучшения (Kaizen) в программной инженерии.** Роль лидерства и организационной культуры в успешном совершенствовании процессов.
10. **Автоматизация процессов как ключевой фактор повышения их зрелости.** Примеры автоматизации в областях тестирования, сборки, развертывания и мониторинга.

Критерии оценки выполнения рефератов по учебной дисциплине

Оценка «отлично» выставляется студенту, если тема реферата раскрыта в полной мере и все требования по написанию реферата соблюдены.

Оценка «хорошо» выставляется студенту, если недостаточно раскрыта тема реферата, но все требования по написанию реферата соблюдены.

Оценка «удовлетворительно» выставляется студенту, если недостаточно раскрыта тема реферата, не все требования по написанию реферата соблюдены, присутствуют ошибки и неточности в работе.

Оценка «неудовлетворительно» выставляется студенту, если отсутствует реферат.

Рекомендации по написанию реферата:

Реферат — письменная работа, выполняемая обучающимся в течение определенного срока.

Реферат — краткое точное изложение сущности какого-либо вопроса, темы на основе одной или нескольких книг, монографий или других первоисточников. Реферат должен содержать основные фактические сведения и выводы по рассматриваемому вопросу.

Реферат отвечает на вопрос — что содержится в данной публикации (публикациях). Реферат — не механический пересказ работы, а изложение ее существа.

Кроме реферирования прочитанной литературы, от обучающегося требуется аргументированное изложение собственных мыслей по рассматриваемому вопросу. Тему реферата предлагает преподаватель или сам обучающийся, в последнем случае она должна быть согласованна с преподавателем.

В реферате нужны развернутые аргументы, рассуждения, сравнения. Материал подается не столько в развитии, сколько в форме констатации или описания.

Содержание реферируемого произведения излагается объективно от имени автора. Если в первичном документе главная мысль сформулирована недостаточно четко, в реферате она должна быть конкретизирована и выделена.

Структура реферата:

1. Титульный лист.

2. После титульного листа на отдельной странице следует оглавление (план, содержание), в котором указаны названия всех разделов (пунктов плана) реферата и номера страниц, указывающие начало этих разделов в тексте реферата.

3. После оглавления следует введение. Объем введения составляет 1,5-2 страницы.

4. Основная часть реферата может иметь одну или несколько глав, состоящих из 2-3 параграфов (подпунктов, разделов) и предполагает осмысленное и логичное изложение главных положений и идей, содержащихся в изученной литературе. В тексте обязательны ссылки на первоисточники. В том случае если цитируется или используется чья-либо неординарная мысль, идея, вывод, приводится какой-либо цифрой материал, таблицу - обязательно сделайте ссылку на того автора у кого вы взяли данный материал.

5. Заключение содержит главные выводы, и итоги из текста основной части, в нем отмечается, как выполнены задачи и достигнуты ли цели, сформулированные во введении.

6. Приложение может включать графики, таблицы, расчеты.

7. Библиография (список литературы) здесь указывается реально использованная для написания реферата литература. Список составляется согласно правилам библиографического описания.

Этапы работы над рефератом.

Работу над рефератом можно условно подразделить на три этапа:

1. Подготовительный этап, включающий изучение предмета исследования;

2. Изложение результатов изучения в виде связного текста;

3. Устное сообщение по теме реферата.

1. Подготовительный этап работы.

Формулировка темы. Подготовительная работа над рефератом начинается с формулировки темы. Тема в концентрированном виде выражает содержание будущего текста, фиксируя как предмет исследования, так и его ожидаемый результат. Для того чтобы работа над рефератом была успешной, необходимо, чтобы тема заключала в себе проблему, скрытый вопрос (даже если наука уже давно дала ответ на этот вопрос, студент, только знакомящийся с соответствующей областью знаний, будет вынужден искать ответ заново, что даст толчок к развитию проблемного, исследовательского мышления).

Поиск источников. Грамотно сформулированная тема зафиксировала предмет изучения; задача обучающегося — найти информацию, относящуюся к данному предмету и разрешить поставленную проблему. Выполнение этой задачи начинается с поиска источников. На этом этапе необходимо вспомнить, как работать с энциклопедиями и энциклопедическими словарями.

Работа с источниками. Работу с источниками надо начинать с ознакомительного чтения, т. е. просмотреть текст, выделяя его структурные единицы. При ознакомительном чтении закладками отмечаются те страницы, которые требуют более внимательного изучения. В зависимости от результатов ознакомительного чтения выбирается дальнейший способ работы с источником. Если для разрешения поставленной задачи требуется изучение некоторых фрагментов текста, то используется метод выборочного чтения. Если в книге нет подробного оглавления, следует обратить внимание на предметные и именные указатели.

Избранные фрагменты или весь текст (если он целиком имеет отношение к теме) требуют вдумчивого, неторопливого чтения с «мысленной проработкой» материала. Такое чтение предполагает выделение: 1) главного в тексте; 2) основных аргументов; 3) выводов. Особое внимание следует обратить на то, вытекает тезис из аргументов или нет. Необходимо также проанализировать, какие из утверждений автора носят проблематичный, гипотетический характер и уловить скрытые вопросы.

Понятно, что умение таким образом работать с текстом приходит далеко не сразу. Наилучший способ научиться выделять главное в тексте, улавливать проблематичный характер утверждений, давать оценку авторской позиции — это сравнительное чтение, в ходе которого студент знакомится с различными мнениями по одному и тому же вопросу, сравнивает весомость и доказательность аргументов сторон и делает вывод о наибольшей убедительности той или иной позиции.

Создание конспектов для написания реферата.

Подготовительный этап работы завершается созданием конспектов, фиксирующих основные тезисы и аргументы. Здесь важно вспомнить, что конспекты пишутся на одной стороне листа, с полями и достаточным для исправления и ремарок межстрочным расстоянием (эти правила соблюдаются для удобства редактирования). Если в конспектах приводятся цитаты, то непременно должно быть дано указание на источник (автор, название, выходные данные, № страницы).

По завершении предварительного этапа можно переходить непосредственно к созданию текста реферата.

2. Создание текста.

Общие требования к тексту.

Текст реферата должен подчиняться определенным требованиям: он должен раскрывать тему, обладать связностью и цельностью.

Раскрытие темы предполагает, что в тексте реферата излагается относящийся к теме материал и предлагаются пути решения содержащейся в теме проблемы; связность текста предполагает смысловую соотносительность отдельных компонентов, а цельность - смысловую законченность текста.

С точки зрения связности все тексты делятся на тексты-констатации и тексты-рассуждения. Тексты-констатации содержат результаты ознакомления с предметом и фиксируют устойчивые и несомненные суждения. В текстах-рассуждениях одни мысли извлекаются из других, некоторые ставятся под сомнение, дается им оценка, выдвигаются

различные предположения.

План реферата.

Универсальный план реферата – введение, основной текст и заключение.

Требования к введению.

Во введении аргументируется актуальность исследования, - т. е. выявляется практическое и теоретическое значение данного исследования. Далее констатируется, что сделано в данной области предшественниками; перечисляются положения, которые должны быть обоснованы. Введение может также содержать обзор источников или экспериментальных данных, уточнение исходных понятий и терминов, сведения о методах исследования. Во введении обязательно формулируются цель и задачи реферата.

Объем введения - в среднем около 10% от общего объема реферата.

Основная часть реферата.

Основная часть реферата раскрывает содержание темы. Она наиболее значительна по объему, наиболее значима и ответственна. В ней обосновываются основные тезисы реферата, приводятся развернутые аргументы, предполагаются гипотезы, касающиеся существа обсуждаемого вопроса. Важно проследить, чтобы основная часть не имела форму монолога. Аргументируя собственную позицию, можно и должно анализировать, и оценивать позиции различных исследователей, с чем-то соглашаться, чему-то возражать, кого-то опровергать. Текст основной части делится на главы, параграфы, пункты. План основной части может быть составлен с использованием различных методов группировки материала: классификации (эмпирические исследования), типологии (теоретические исследования), периодизации (исторические исследования).

Заключение.

Заключение — последняя часть научного текста. В ней краткой и сжатой форме излагаются полученные результаты, представляющие собой ответ на главный вопрос исследования. Здесь же могут намечаться и дальнейшие перспективы развития темы. Небольшое по объему сообщение также не может обойтись без заключительной части - пусть это будут две-три фразы. Но в них должен подводиться итог проделанной работы.

Список использованной литературы.

Реферат любого уровня сложности обязательно сопровождается списком используемой литературы. Названия книг в списке располагают по алфавиту с указанием выходных данных использованных книг.

Требования, предъявляемыек оформлению реферата

Объемы рефератов – от 10-18 машинописных страниц. Работа выполняется на одной стороне листа стандартного формата. По обеим сторонам листа оставляются поля размером 35 мм. слева и 15 мм. справа, рекомендуется шрифт 12-14, интервал - 1,5. Все листы реферата должны быть пронумерованы. Каждый вопрос в тексте должен иметь заголовок в точном соответствии с наименованием в плане-оглавлении. При написании и оформлении реферата следует избегать типичных ошибок, например, таких:

– поверхностное изложение основных теоретических вопросов выбранной темы, когда автор не понимает, какие проблемы в тексте являются главными, а какие второстепенными,

– в некоторых случаях проблемы, рассматриваемые в разделах, не раскрывают основных аспектов выбранной для реферата темы,

– дословное переписывание книг, статей, заимствования рефератов из интернета и т. д.

При проверке реферата преподавателем оцениваются:

1. Знания и умения на уровне требований стандарта конкретной дисциплины: знание фактического материала, усвоение общих представлений, понятий, идей.

2. Характеристика реализации цели и задач исследования (новизна и актуальность поставленных в реферате проблем, правильность формулирования цели, определения задач исследования, правильность выбора методов решения задач и реализации цели; соответствие

выводов решаемым задачам, поставленной цели, убедительность выводов).

3. Степень обоснованности аргументов и обобщений (полнота, глубина, всесторонность раскрытия темы, логичность и последовательность изложения материала, корректность аргументации и системы доказательств, характер и достоверность примеров, иллюстративного материала, широта кругозора автора, наличие знаний интегрированного характера, способность к обобщению).

4. Качество и ценность полученных результатов (степень завершенности реферативного исследования, спорность или однозначность выводов).

5. Использование литературных источников.

6. Культура письменного изложения материала.

7. Культура оформления материалов работы.

Комплект типовых практических заданий

Тема 1. Методология DevOps для создания ПО

Задание 1.1: Разработка стратегии DevOps-трансформации для унаследованной системы

- **Постановка задачи:** Вам необходимо спланировать DevOps-трансформацию для крупного монолитного банковского приложения, которое сейчас развертывается раз в полгода с ручными процессами и низким уровнем автоматизированного тестирования.
- **Требования:**
    1. Проведите анализ разрыва (Gap Analysis) между текущим и целевым состоянием процесса.
    2. Разработайте поэтапный дорожный план (Roadmap) трансформации на 12 месяцев, выделив ключевые вехи.
    3. Обоснуйте выбор инструментов для CI/CD, мониторинга и инфраструктуры, учитывая требования безопасности (комплаенс).
    4. Предложите конкретные метрики (DORA и др.), по которым будет оцениваться успешность трансформации.
    5. Опишите план работы с культурным сопротивлением в командах разработки и эксплуатации.

Задание 1.2: Проектирование отказоустойчивого CI/CD-пайплайна для микросервисной архитектуры

- **Постановка задачи:** Спроектируйте CI/CD-пайплайн для системы, состоящей из 20+ взаимозависимых микросервисов. Пайплайн должен быть устойчив к сбоям, эффективно использовать ресурсы и предотвращать развертывание нестабильных версий.
- **Требования:**
    1. Разработайте стратегию ветвления и управления зависимостями между микросервисами.
    2. Опишите механизм "кандидатских сборок" (build candidates) и их продвижения по стадиям (dev -> staging -> prod).
    3. Реализуйте механизм автоматического отката (rollback) для случая, если один из сервисов causes инцидент после развертывания.
    4. Интегрируйте в пайплайн шаги проверки безопасности (SAST, DAST) и лицензий зависимостей.
    5. Предложите решение для "горячего" обновления конфигураций без пересборки артефактов (например, с помощью Feature Flags или конфигурационных серверов).

Тема 2. Управление требованиями к ПО

Задание 2.1: Разработка подхода к управлению нефункциональными требованиями (NFR) для высоконагруженной системы

- **Постановка задачи:** Для разрабатываемой системы онлайн-трейдинга с пиковой нагрузкой в 10 000 транзакций в секунду необходимо формализовать и контролировать выполнение нефункциональных требований.
- **Требования:**
    1. Разработайте классификацию NFR для данного проекта (производительность, безопасность, надежность, масштабируемость и т.д.).
    2. Для каждого класса предложите конкретные, измеримые критерии приемки (например, "время отклика 95-го перцентиля < 100 мс", "доступность 99.95%").
    3. Создайте план, описывающий, как эти требования будут проверяться на каждом этапе жизненного цикла (архитектурный обзор, тестирование, эксплуатация).
    4. Предложите механизм трассируемости NFR от бизнес-требований до метрик в продакшн-окружении.
    5. Опишите, как будет решаться конфликт между функциональными требованиями и NFR (например, между скоростью разработки новой функции и требованиями к безопасности).

Задание 2.2: Разрешение конфликта требований в условиях неопределенности

- **Постановка задачи:** В рамках проекта по созданию медицинского ПО возник конфликт: клиенты-врачи требуют максимально простой и быстрый интерфейс для ввода данных, в то время как регуляторы предъявляют строгие требования к аудиту и ведению полной истории изменений, что усложняет UI.
- **Требования:**
    1. Проведите анализ стейкхолдеров и их интересов.
    2. Примените технику анализа компромиссов (trade-off analysis) для поиска возможных решений.
    3. Предложите 2-3 альтернативных сценария реализации, оценив плюсы и минусы каждого с точки зрения разных групп стейкхолдеров.
    4. Разработайте протокол принятия решения по выбору финального варианта.
    5. Сформулируйте итоговое решение, обосновав его с точки зрения профессиональной этики (опираясь на кодекс IEEE/ACM).

Тема 3. Управление дизайном ПО

Задание 3.1: Миграция монолита на микросервисную архитектуру

- **Постановка задачи:** Существует успешное монолитное e-commerce приложение, которое стало трудно поддерживать и масштабировать. Вам необходимо спроектировать процесс его декомпозиции на микросервисы.
- **Требования:**
    1. Выберите и обоснуйте стратегию декомпозиции (например, по доменам в стиле DDD).
    2. Спроектируйте границы сервисов, определите контракты API между ними.
    3. Опишите стратегию управления данными: разделение БД, обеспечение консистентности (сага, eventual consistency).
    4. Спроектируйте architecture runway: какие системные сервисы необходимо разработать в первую очередь (логирование, мониторинг, service discovery, API Gateway).
    5. Предложите поэтапный план миграции, позволяющий постепенно переносить функциональность без полной остановки системы (strangler fig pattern).

Задание 3.2: Проектирование системы с учетом требований к устойчивости (Resilience)

- **Постановка задачи:** Спроектируйте архитектуру системы онлайн-бронирования, которая должна оставаться работоспособной при отказах внешних сервисов (сервис платежей, сервис нотификаций, партнерские API).
- **Требования:**
    1. Примените паттерны устойчивости (Circuit Breaker, Retry, Fallback, Bulkhead) и укажите, в каких именно точках системы они будут использоваться.
    2. Спроектируйте асинхронные взаимодействия, где это необходимо, с использованием message broker (например, RabbitMQ/Kafka).
    3. Опишите, как система будет компенсировать действия (compensating transactions), если длинная транзакция не может быть завершена.
    4. Разработайте сценарии "деградации функциональности": что будет делать система, если критический сервис недоступен.
    5. Представьте диаграмму последовательности для сценария "оформление бронирования при временной недоступности платежного шлюза".

Тема 4. Управление версиями ПО

Задание 4.1: Разработка политики управления версиями для сложного продукта с открытым исходным кодом

- **Постановка задачи:** Вы являетесь мейнтейнером популярной open-source библиотеки. Необходимо формализовать процесс управления версиями, ветвления, приема pull request'ов и выпуска релизов.
- **Требования:**
    1. Разработайте и задокументируйте соглашение по именованию версий (SemVer).
    2. Определите модель ветвления (GitFlow, Trunk-Based Development и т.д.), которая подходит для проекта с большой базой контрибьюторов. Обоснуйте выбор.
    3. Опишите workflow приема pull request: какие проверки (CI, ревью) обязательны, кто имеет права на мерж, как обрабатываются breaking changes.
    4. Создайте политику поддержки legacy-версий: сколько предыдущих мажорных версий получают исправления безопасности.
    5. Автоматизируйте процесс с помощью GitHub Actions/GitLab CI: создайте конфигурацию, которая автоматически собирает и публикует релиз при создании тега, генерирует changelog.

Задание 4.2: Проектирование системы управления конфигурацией для гибридного облачного окружения

- **Постановка задачи:** Разработайте систему для управления конфигурациями (код, параметры, секреты) для приложения, развернутого в гибридной среде (часть сервисов в Kubernetes, часть на bare-metal серверах).
- **Требования:**
    1. Выберите и обоснуйте инструменты для хранения конфигураций (например, Helm Charts для K8s, Ansible для bare-metal) и их версионирования.
    2. Предложите стратегию управления секретами (secrets) для всей среды.
    3. Разработайте механизм промоутинга конфигураций между средами (dev -> stage -> prod), обеспечивающий их идентичность и контроль изменений.
    4. Опишите, как будет решаться проблема конфигурационного дрейфа (configuration drift).
    5. Спроектируйте схему хранения и версионирования "золотых образов" (Golden Images) для виртуальных машин и Docker-образов.

Тема 5. Управление качеством ПО

Задание 5.1: Построение комплексной системы обеспечения качества для FinTech-приложения

- **Постановка задачи:** Разработайте всеобъемлющую стратегию QA для мобильного банкинга, где ошибки могут привести к финансовым потерям и репутационным рискам.
- **Требования:**
    1. Разработайте многоуровневую стратегию тестирования (Test Pyramid), определив баланс между unit, integration, API, E2E-тестами.
    2. Предложите методы тестирования нефункциональных требований: безопасность (пентесты), производительность (load testing), доступность (a11y).
    3. Интегрируйте в процесс разработки практики "сдвига качества влево" (Shift-Left): статический анализ кода (SAST), проверки зависимостей (SCA), пре-коммит хуки.
    4. Спроектируйте процесс управления дефектами, включая классификацию по критичности, SLA на исправление и анализ первопричин (Root Cause Analysis) для дефектов высокой severity.
    5. Создайте концепцию "качества как общего дела" (Quality Ownership), определяющую роль каждого участника команды (разработчик, тестировщик, продакт-менеджер) в обеспечении качества.

Задание 5.2: Оптимизация набора регрессионных тестов

- **Постановка задачи:** В проекте накоплено 10 000 UI-тестов, которые выполняются 8 часов. Это блокирует частые релизы. Необходимо оптимизировать набор тестов, не жертвуя надежностью.
- **Требования:**
    1. Проведите анализ набора тестов: выделите дубликаты, хрупкие (flaky) и малополезные тесты.
    2. Предложите стратегию сегментации тестов для параллельного выполнения (например, по функциональным модулям, по критичности).
    3. Разработайте подход к определению "минимально необходимого" набора регрессионных тестов для быстрого прогона (Smoke Suite).
    4. Предложите методы для стабилизации flaky-тестов.
    5. Внедрите практику "аналитики покрытия тестами" (Test Coverage Analysis) для принятия обоснованных решений о том, какие тесты писать, а какие — удалять.

Тема 6. Управление сборками ПО

Задание 6.1: Создание системы управления артефактами и зависимостями для крупной распределенной команды

- **Постановка задачи:** В компании 10 команд разрабатывают 50+ компонентов на разных языках (Java, Python, JS). Существуют проблемы с версиями зависимостей, конфликтами и воспроизводимостью сборок.
- **Требования:**
    1. Спроектируйте архитектуру системы хранения артефактов (артефакт-репозитория) с использованием Artifactory/Nexus.
    2. Разработайте единую политику версионирования для всех артефактов.
    3. Реализуйте механизм управления зависимостями, предотвращающий использование неразрешенных или уязвимых библиотек.
    4. Настройте "виртуальные" репозитории, агрегирующие артефакты из разных источников (npm, PyPI, Maven Central).
    5. Обеспечьте возможность воспроизведения любой сборки, выпущенной за последние 3 года, включая все её транзитивные зависимости.

Задание 6.2: Оптимизация и ускорение CI-пайплайна для монолитного приложения

- **Постановка задачи:** CI-пайплайн монолита занимает 2 часа. Большую часть времени занимает сборка и прогон тестов. Необходимо сократить время до 30 минут.
- **Требования:**
    1. Проведите профилирование пайплайна, чтобы найти "узкие места".
    2. Реализуйте стратегию инкрементальных сборок, чтобы пересобирались только измененные модули.
    3. Внедрите кэширование зависимостей и результатов промежуточных шагов сборки.
    4. Реорганизуйте запуск тестов: разделите их на независимые группы для параллельного выполнения на множестве агентов.
    5. Внедрите распределенное кэширование для системы сборки (Gradle Build Cache, Bazel Remote Caching).

Тема 7. Управление развертыванием ПО

Задание 7.1: Проектирование стратегии развертывания с нулевым временем простоя (Zero-Downtime Deployment) для stateful-приложения

- **Постановка задачи:** Необходимо обеспечить бесшовное обновление распределенной системы, которая управляет состояниями пользовательских сессий (например, онлайн-игра или торговая платформа).
- **Требования:**
    1. Сравните стратегии Blue-Green, Canary и Rolling Update применительно к stateful-сервисам. Выберите и обоснуйте наиболее подходящую.
    2. Разработайте механизм миграции данных (Database Migrations), который будет совместим со старой и новой версией приложения во время деплоя.
    3. Опишите процедуру "слияния трафика" (traffic shifting) и управления сессиями пользователей.
    4. Предусмотрите сценарий отката, который не приведет к потере данных пользователей.
    5. Интегрируйте в процесс автоматизированные проверки здоровья (health checks, readiness/liveness probes) для принятия решения о успешности деплоя.

Задание 7.2: Построение GitOps-пайплайна для управления инфраструктурой и приложениями в Kubernetes

- **Постановка задачи:** Реализуйте GitOps-подход для кластера Kubernetes, где желаемое состояние инфраструктуры и приложений описывается в Git-репозитории и автоматически применяется.
- **Требования:**
    1. Выберите и настройте GitOps-оператор (ArgoCD или Flux).
    2. Структурируйте репозиторий с декларативными манифестами (Kustomize или Helm).
    3. Настройте автоматическую синхронизацию: при пуше в определенную ветку манифесты автоматически применяются в кластере.
    4. Реализуйте механизм preview-окружений для pull request'ов.
    5. Обеспечьте безопасность: настройте RBAC, контроль доступа к репозиторию и механизм подписывания манифестов (cosign).

Тема 8. Совершенствование программного процесса

Задание 8.1: Проведение аудита процессов и разработка плана улучшений для достижения CMMI Level 3

- **Постановка задачи:** Проведите оценку текущего состояния процессов разработки в компании и создайте детализированный план улучшений для достижения целевого уровня зрелости CMMI 3.
- **Требования:**
    1. Проведите интервью с ключевыми стейкхолдерами и проанализируйте артефакты проектов.
    2. Сопоставьте текущие практики компании с целями практик (Specific Goals) CMMI для уровня 3 (например, Requirements Management, Technical Solution, Verification, Validation).
    3. Выявите наиболее критические разрывы (gaps) и области для улучшения.
    4. Разработайте план улучшений (Improvement Plan) с приоритизированными инициативами, сроками и ответственными.
    5. Определите метрики, которые будут использоваться для оценки эффективности внедренных улучшений.

Задание 8.2: Внедрение культуры непрерывного улучшения на основе данных и метрик

- **Постановка задачи:** Создайте систему, которая позволит командам разработки самостоятельно выявлять bottlenecks в своем процессе и инициировать улучшения.
- **Требования:**
    1. Определите набор метрик (например, Cycle Time, Lead Time, Deployment Frequency, Change Failure Rate), которые будут собираться автоматически.
    2. Создайте дашборды, визуализирующие эти метрики для каждой команды и в разрезе всей организации.
    3. Разработайте и проведите воркшоп для тимлидов и менеджеров по интерпретации этих данных.
    4. Интегрируйте регулярный анализ метрик в процесс ретроспектив команд.
    5. Создайте lightweight-процесс для предложения и пилотирования улучшений, инициированных командами на основе данных.

Критерии оценивания выполнения практического задания:

Оценка «отлично» выставляется, если: работа выполнена в полном объеме с соблюдением необходимой последовательности действий; работа проведена в условиях, обеспечивающих получение правильных результатов и выводов; соблюдены правила техники безопасности; в ответе правильно и аккуратно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ ошибок.

Оценка «хорошо» выставляется, если: работа выполнена правильно с учетом 1-2 мелких погрешностей или 2-3 недочетов, исправленных самостоятельно по требованию преподавателя.

Оценка «удовлетворительно» выставляется, если: работа выполнена правильно не менее чем наполовину, допущены 1-2 погрешности или одна грубая ошибка.

Оценка «неудовлетворительно» выставляется, если: допущены две (и более) грубые ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя или работа не выполнена полностью.

Типовые тесты

Тема 1. Методология DevOps для создания ПО

П1. Какая из перечисленных практик НЕ является одной из "Трех основ" (Three Ways) DevOps?

а) Ускорение потока работы от разработки к эксплуатации

б) Создание коротких циклов обратной связи

в) Непрерывный эксперимент и обучение
г) Полная автоматизация всех ручных процессов
Ответ: г) Полная автоматизация всех ручных процессов

П2. Что из перечисленного является ключевой культурной ценностью DevOps?
а) Строгое разделение обязанностей между отделами
б) Следование только утвержденным процессам
в) Сотрудничество между командами разработки и эксплуатации
г) Приоритет скорости над качеством
Ответ: в) Сотрудничество между командами разработки и эксплуатации

П3. Какие из перечисленных инструментов относятся к этапу "Сборка" (Build) в DevOps-цепочке?
а) Jenkins, GitLab CI, CircleCI
б) Docker, Kubernetes, Terraform
в) Selenium, JUnit, pytest
г) Prometheus, Grafana, ELK Stack
Ответ: а) Jenkins, GitLab CI, CircleCI

П4. Какой показатель НЕ относится к метрикам DORA?
а) Частота развертывания (Deployment Frequency)
б) Время восстановления службы (Mean Time to Recovery)
в) Общее количество строк кода
г) Частота отказов изменений (Change Failure Rate)
Ответ: в) Общее количество строк кода

П5. Установите соответствие между понятием и его определением:

Понятие	Определение
1. CI	а) Автоматический процесс развертывания кода в продакшен
2. CD	б) Практика частого слияния кода в общую ветку с автоматической сборкой и тестированием
3. IaC	в) Управление инфраструктурой с помощью конфигурационных файлов

Правильный ответ: 1-б, 2-а, 3-в

П6. Установите правильную последовательность этапов CI/CD пайплайна:
а) Сборка приложения
б) Запуск автоматических тестов
в) Мерж кода в основную ветку
г) Развертывание в staging-окружении
д) Развертывание в prod-окружении
Ответ: в-а-б-г-д

П7. Какая практика позволяет быстро откатить проблемное развертывание?
а) Blue-Green развертывание
б) Canary развертывание
в) Feature Flags
г) Все перечисленные
Ответ: г) Все перечисленные

П8. Какой инструмент используется для оркестрации контейнеров?
а) Docker
б) Kubernetes
в) Terraform
г) Ansible
Ответ: б) Kubernetes

П9. Что означает аббревиатура "IaC" в контексте DevOps?
Правильный ответ: Инфраструктура как код (Infrastructure as Code)

П10. Как называется практика включения специалистов по безопасности в процесс разработки?

Правильный ответ: DevSecOps

Тема 2. Управление требованиями к ПО

П1. Какое требование описывает, ЧТО должна делать система?

а) Функциональное требование
б) Нефункциональное требование
в) Бизнес-требование
г) Пользовательское требование

Ответ: а) Функциональное требование

П2. Какое свойство НЕ относится к качественным требованиям?

а) Однозначность
б) Проверяемость
в) Реализуемость
г) Креативность

Ответ: г) Креативность

П3. Какие из перечисленных являются нефункциональными требованиями?

а) Система должна поддерживать 1000 одновременных пользователей
б) Система должна иметь функцию поиска товаров
в) Время отклика системы не должно превышать 2 секунды
г) Система должна интегрироваться с платежным шлюзом

Правильные ответы: а, в

П4. Какой метод сбора требований наиболее эффективен для получения экспертного мнения?

а) Анкетирование
б) Интервью
в) Наблюдение
г) Мозговой штурм

Ответ: б) Интервью

П5. Установите соответствие между видом требования и примером:

Вид требования	Пример
1. Функциональное	а) Система должна сохранять резервные копии ежедневно
2. Производительность	б) Пользователь может добавлять товары в корзину
3. Безопасность	в) Данные должны передаваться по зашифрованному каналу

Правильный ответ: 1-б, 2-а, 3-в

П6. Что такое "User Story" в Agile-разработке?

а) Детальное техническое описание функции
б) Пользовательское требование в формате "Как , я хочу , чтобы "
в) Диаграмма последовательности действий пользователя
г) Прототип пользовательского интерфейса

Ответ: б) Пользовательское требование в формате "Как , я хочу , чтобы "

П7. Какой артефакт содержит полное описание всех требований к системе?

а) Бизнес-требования
б) Техническое задание
в) Пользовательские истории
г) Спецификация требований

Ответ: г) Спецификация требований

П8. Что такое трассируемость требований?

а) Возможность отследить связь между требованием и его реализацией
б) Скорость изменения требований
в) Стоимость реализации требования

г) Важность требования для бизнеса

Ответ: а) Возможность отследить связь между требованием и его реализацией

П9. Как называется документ, описывающий общее видение и границы проекта?

Правильный ответ: Vision and Scope document

П10. Как называется процесс управления изменениями в требованиях?

Правильный ответ: Управление изменениями (Change Management)

Тема 3. Управление дизайном ПО

П1. Какой принцип ООП обеспечивает скрытие внутренней реализации?

а) Наследование
б) Инкапсуляция
в) Полиморфизм
г) Абстракция

Ответ: б) Инкапсуляция

П2. Какой архитектурный паттерн разделяет приложение на Model, View и Controller?

а) MVC
б) MVP
в) MVVM
г) Микросервисы

Ответ: а) MVC

П3. Какие из перечисленных являются принципами SOLID?

а) Single Responsibility Principle
б) Don't Repeat Yourself
в) Open/Closed Principle
г) Keep It Simple, Stupid

Правильные ответы: а, в

П4. Какой инструмент используется для визуального моделирования архитектуры?

а) UML
б) JSON
в) XML
г) SQL

Ответ: а) UML

П5. Установите соответствие между паттерном проектирования и его назначением:

Паттерн	Назначение
1. Singleton	а) Создание объектов без указания точного класса
2. Factory	б) Обеспечение единственного экземпляра класса
3. Observer	в) Определение зависимости "один-ко-многим" между объектами

Правильный ответ: 1-б, 2-а, 3-в

П6. Что такое "архитектурное решение"?

а) Выбор технологии для реализации
б) Важное проектное решение, влияющее на структуру системы
в) Диаграмма компонентов системы
г) Документация к API

Ответ: б) Важное проектное решение, влияющее на структуру системы

П7. Какой подход к проектированию фокусируется на бизнес-домене?

а) Functional Programming
б) Domain-Driven Design
в) Test-Driven Development
г) Behavior-Driven Development

Ответ: б) Domain-Driven Design

П8. Что такое "technical debt"?
а) Долг за использование проприетарных технологий
б) Накопленные проблемы в коде, требующие будущего исправления
в) Стоимость лицензирования ПО
г) Затраты на техническую поддержку
Ответ: б) Накопленные проблемы в коде, требующие будущего исправления

П9. Как называется принцип, при котором класс должен быть открыт для расширения, но закрыт для изменений?
Правильный ответ: Принцип открытости/закрытости (Open/Closed Principle)

П10. Как называется документ, описывающий архитектуру системы?
Правильный ответ: Архитектурное описание (Architecture Description)

Тема 4. Управление версиями ПО

П1. Какая система контроля версий является распределенной?
а) SVN
б) CVS
в) Git
г) TFS
Ответ: в) Git

П2. Что такое "commit" в системе контроля версий?
а) Запрос на слияние изменений
б) Фиксация изменений в репозитории
в) Создание новой ветки
г) Обновление локальной копии
Ответ: б) Фиксация изменений в репозитории

П3. Какие из перечисленных операций относятся к работе с ветками в Git?
а) clone, push, pull
б) branch, merge, rebase
в) add, commit, status
г) fetch, remote, init
Правильные ответы: б

П4. Какой подход к ветвлению предполагает разработку в основной ветке?
а) Git Flow
б) GitHub Flow
в) GitLab Flow
г) Trunk-Based Development
Ответ: г) Trunk-Based Development

П5. Установите соответствие между командой Git и ее назначением:

Команда	Назначение
1. git merge	а) Перезапись истории коммитов
2. git rebase	б) Слияние веток
3. git stash	в) Временное сохранение изменений

Правильный ответ: 1-б, 2-а, 3-в

П6. Что такое "semantic versioning"?
а) Система именования версий на основе даты
б) Система версионирования по количеству коммитов
в) Семантическая система нумерации версий MAJOR.MINOR.PATCH
г) Система версионирования на основе хэшей коммитов
Ответ: в) Семантическая система нумерации версий MAJOR.MINOR.PATCH

П7. Что такое ".gitignore" файл?
а) Файл с настройками Git
б) Файл, определяющий игнорируемые файлы и папки

в) Файл с историей коммитов
г) Файл с информацией о ветках
Ответ: б) Файл, определяющий игнорируемые файлы и папки

П8. Какой протокол используется для безопасного соединения с Git-репозиторием?
а) HTTP
б) HTTPS
в) SSH
г) FTP
Ответ: в) SSH

П9. Как называется сервис для хостинга Git-репозиториев?
Правильный ответ: GitHub, GitLab, Bitbucket

П10. Как называется запрос на слияние изменений в Git?
Правильный ответ: Pull Request или Merge Request

Тема 5. Управление качеством ПО

П1. Какой вид тестирования проверяет отдельные модули кода?
а) Интеграционное тестирование
б) Системное тестирование
в) Модульное тестирование
г) Приемочное тестирование
Ответ: в) Модульное тестирование

П2. Что такое "test case"?
а) Инструмент для автоматического тестирования
б) Документация по тестированию
в) Конкретный сценарий тестирования с входными данными и ожидаемым результатом
г) Отчет об ошибках
Ответ: в) Конкретный сценарий тестирования с входными данными и ожидаемым результатом

П3. Какие из перечисленных являются принципами F.I.R.S.T. для unit-тестов?
а) Fast, Independent, Repeatable
б) Functional, Integrated, Reliable
в) Secure, Testable, Simple
г) Automated, Manual, Mixed
Правильные ответы: а

П4. Что такое "regression testing"?
а) Тестирование новых функций
б) Тестирование, обеспечивающее, что исправления не сломали существующий функционал
в) Тестирование производительности
г) Тестирование безопасности
Ответ: б) Тестирование, обеспечивающее, что исправления не сломали существующий функционал

П5. Установите соответствие между видом тестирования и его назначением:

Вид тестирования	Назначение
1. Нагрузочное	а) Проверка работы системы под нагрузкой
2. Безопасности	б) Проверка уязвимостей системы
3. Usability	в) Проверка удобства использования

Правильный ответ: 1-а, 2-б, 3-в

П6. Что такое "code coverage"?
а) Процент кода, покрытого тестами
б) Количество тестов в проекте
в) Скорость выполнения тестов

г) Качество написания тестов

Ответ: а) Процент кода, покрытого тестами

П7. Какой инструмент используется для статического анализа кода?

а) Selenium

б) SonarQube

в) JMeter

г) Postman

Ответ: б) SonarQube

П8. Что такое "bug tracker"?

а) Инструмент для автоматического поиска ошибок

б) Система для отслеживания и управления дефектами

в) Инструмент для профилирования кода

г) Система для управления тестовыми данными

Ответ: б) Система для отслеживания и управления дефектами

П9. Как называется тестирование, выполняемое самими разработчиками?

Правильный ответ: Модульное тестирование (Unit Testing)

П10. Как называется методология, при которой тесты пишутся до кода?

Правильный ответ: Test-Driven Development (TDD)

Тема 6. Управление сборками ПО

П1. Какой инструмент используется для сборки Java-проектов?

а) Maven

б) npm

в) pip

г) make

Ответ: а) Maven

П2. Что такое "dependency management"?

а) Управление версиями библиотек и зависимостей

б) Управление командой разработки

в) Управление конфигурациями серверов

г) Управление задачами в проекте

Ответ: а) Управление версиями библиотек и зависимостей

П3. Какие из перечисленных являются системами сборки?

а) Jenkins, GitLab CI

б) Maven, Gradle, Ant

в) Docker, Kubernetes

г) JUnit, TestNG

Правильные ответы: б

П4. Что такое "continuous integration"?

а) Непрерывное развертывание в продакшен

б) Автоматическая сборка и тестирование при каждом изменении кода

в) Непрерывный мониторинг приложения

г) Автоматическое создание документации

Ответ: б) Автоматическая сборка и тестирование при каждом изменении кода

П5. Установите соответствие между файлом и его назначением:

Файл	Назначение
1. pom.xml	а) Конфигурация сборки для Maven
2. package.json	б) Конфигурация зависимостей для Node.js
3. Dockerfile	в) Инструкции для сборки Docker-образа

Правильный ответ: 1-а, 2-б, 3-в

П6. Что такое "build artifact"?

а) Исходный код проекта

б) Результат процесса сборки (jar, war, exe файлы)
в) Документация проекта
г) Тестовые данные
Ответ: б) Результат процесса сборки (jar, war, exe файлы)

П7. Какой инструмент используется для управления зависимостями в Python?
а) pip
б) npm
в) NuGet
г) Composer
Ответ: а) pip

П8. Что такое "repository" в контексте управления зависимостями?
а) Хранилище исходного кода
б) Хранилище собранных артефактов
в) Хранилище бинарных зависимостей
г) База данных проекта
Ответ: в) Хранилище бинарных зависимостей

П9. Как называется сервер для хранения собранных артефактов?
Правильный ответ: Артефакт-репозиторий (Artifact Repository)

П10. Как называется процесс создания исполняемого файла из исходного кода?
Правильный ответ: Компиляция (Compilation) или сборка (Build)

Тема 7. Управление развертыванием ПО

П1. Какая стратегия развертывания предполагает одновременную работу старой и новой версии?
а) Rolling update
б) Blue-green deployment
в) Canary release
г) Все перечисленные
Ответ: г) Все перечисленные

П2. Что такое "containerization"?
а) Виртуализация на уровне операционной системы
б) Полная виртуализация сервера
в) Разделение приложения на микросервисы
г) Упаковка приложения со всеми зависимостями
Ответ: г) Упаковка приложения со всеми зависимостями

П3. Какие из перечисленных являются инструментами для развертывания?
а) Docker, Kubernetes
б) Jenkins, GitLab CI
в) Maven, Gradle
г) Jira, Confluence
Правильные ответы: а

П4. Что такое "infrastructure as code"?
а) Написание кода для инфраструктуры
б) Управление инфраструктурой через конфигурационные файлы
в) Программирование сетевого оборудования
г) Создание документации по инфраструктуре
Ответ: б) Управление инфраструктурой через конфигурационные файлы

П5. Установите соответствие между понятием и определением:

Понятие	Определение
1. Orchestration	а) Автоматическое управление контейнерами
2. Provisioning	б) Подготовка и настройка инфраструктуры
3. Configuration	в) Настройка установленного программного обеспечения

Правильный ответ: 1-а, 2-б, 3-в

П6. Что такое "rollback"?

а) Откат к предыдущей версии приложения
б) Обновление до новой версии
в) Резервное копирование данных
г) Настройка балансировщика нагрузки

Ответ: а) Откат к предыдущей версии приложения

П7. Какой инструмент используется для оркестрации контейнеров?

а) Docker
б) Kubernetes
в) Terraform
г) Ansible

Ответ: б) Kubernetes

П8. Что такое "feature flag"?

а) Флаг версии приложения
б) Механизм включения/выключения функциональности без переразвертывания
в) Индикатор состояния системы
г) Маркер для тестирования

Ответ: б) Механизм включения/выключения функциональности без переразвертывания

П9. Как называется практика постепенного развертывания новой версии на небольшую группу пользователей?

Правильный ответ: Canary release

П10. Как называется файл с описанием развертывания в Kubernetes?

Правильный ответ: Манифест (Manifest) или YAML-файл

Тема 8. Совершенствование программного процесса

П1. Что означает аббревиатура CMMI?

а) Capability Maturity Model Integration
б) Continuous Management Model Integration
в) Capability Management Model International
г) Continuous Maturity Model International

Ответ: а) Capability Maturity Model Integration

П2. Какой уровень CMMI характеризуется предсказуемыми процессами?

а) Уровень 1 - Initial
б) Уровень 2 - Managed
в) Уровень 3 - Defined
г) Уровень 4 - Quantitatively Managed

Ответ: в) Уровень 3 - Defined

П3. Какие из перечисленных являются моделями совершенствования процессов?

а) CMMI, ISO 9001
б) Scrum, Kanban
в) Waterfall, Agile
г) DevOps, CI/CD

Правильные ответы: а

П4. Что такое "process assessment"?

а) Оценка эффективности процессов
б) Назначение процессов команде
в) Документирование процессов
г) Автоматизация процессов

Ответ: а) Оценка эффективности процессов

П5. Установите соответствие между уровнем CMMI и его характеристикой:

Уровень CMMI	Характеристика
1. Уровень 1	а) Процессы хаотичны и непредсказуемы
2. Уровень 2	б) Процессы определены и стандартизированы
3. Уровень 3	в) Процессы управляются на проектом уровне

Правильный ответ: 1-а, 2-в, 3-б

П6. Что такое "continuous improvement"?

а) Постоянное совершенствование процессов
б) Непрерывная интеграция кода
в) Постоянное обновление технологий
г) Непрерывное обучение сотрудников

Ответ: а) Постоянное совершенствование процессов

П7. Какой метод используется для анализа причин проблем?

а) Root cause analysis
б) SWOT analysis
в) PEST analysis
г) Cost-benefit analysis

Ответ: а) Root cause analysis

П8. Что такое "metrics" в контексте управления процессами?

а) Количественные показатели для измерения эффективности
б) Качественные характеристики процессов
в) Документация по процессам
г) Инструменты для автоматизации

Ответ: а) Количественные показатели для измерения эффективности

П9. Как называется цикл постоянного улучшения "Планируй-Делай-Проверяй-Воздействуй"?

Правильный ответ: Цикл Деминга (PDCA)

П10. Как называется регулярная встреча команды для обсуждения улучшений процесса?

Правильный ответ: Ретроспектива (Retrospective)

Критерии оценивания тестовых заданий для текущего контроля

Максимальное количество баллов за решение одного теста составляет 5 баллов, которые переводятся в академические оценки следующим образом:

- оценка «отлично» (5 баллов) выставляется при условии правильного ответа студента не менее чем на 85% контрольных заданий
- оценка «хорошо» (4 балла) выставляется при условии правильного ответа студента не менее чем на 70% контрольных заданий
- оценка «удовлетворительно» (3 балла) выставляется при условии правильного ответа студента не менее чем на 51% контрольных заданий
- оценка «неудовлетворительно» (1-2 балла) выставляется при условии правильного ответа студента менее чем на 50% контрольных заданий

2.2 Оценочные материалы для проведения промежуточной аттестации

ПК-3.2.1 Знает подходы к проектированию программных и аппаратных средств защиты информации

Эта компетенция проверяет знание фундаментальных принципов, архитектурных решений и технологий.

П1. Какой принцип проектирования систем защиты информации предполагает предоставление пользователям и процессам минимальных привилегий, необходимых для выполнения задач?

а) Принцип полномочий
б) Принцип минимальных привилегий ✓
в) Принцип открытого доступа
г) Принцип эскалации привилегий
Ответ: б) Принцип минимальных привилегий

П2. Какой аппаратный модуль в современных мобильных устройствах и серверах предназначен для безопасного хранения криптографических ключей и выполнения защищенных операций?
а) Центральный процессор (CPU)
б) Графический процессор (GPU)
в) Модуль доверенной среды выполнения (Trusted Execution Environment, TEE) или защищенная оболочка (Secure Enclave) ✓
г) Сетевой контроллер
Ответ: в) Модуль доверенной среды выполнения (Trusted Execution Environment, TEE) или защищенная оболочка (Secure Enclave)

П3. Какие из перечисленных подходов являются ключевыми для проектирования безопасной сетевой архитектуры (выберите несколько)?
а) Сегментация сети (Network Segmentation) ✓
б) Единая плоская сеть для упрощения управления
в) Принцип "ноль доверия" (Zero Trust) ✓
г) Использование только программных средств защиты
Правильные ответы: а, в

П4. Что из перечисленного относится к аппаратным средствам защиты информации на уровне процессора?
а) Система предотвращения вторжений (IPS)
б) Технологии исполнения без права доступа (NX bit) и защиты от переполнения буфера ✓
в) Межсетевой экран (Firewall)
г) Антивирусное программное обеспечение
Ответ: б) Технологии исполнения без права доступа (NX bit) и защиты от переполнения буфера

П5. Установите соответствие между архитектурным принципом безопасности и его описанием:

Принцип	Описание
1. Защита в глубину (Defense in Depth)	а) Реализация нескольких независимых уровней защиты
2. Разделение обязанностей (Separation of Duties)	б) Ни одному субъекту не должно быть предоставлено достаточно прав для злоупотребления системой
3. Наименьшая привилегия (Least Privilege)	в) Предоставление пользователям только тех прав, которые необходимы для выполнения их задач

Правильный ответ: 1-а, 2-б, 3-в

П6. Установите правильную последовательность этапов проектирования системы защиты информации:
а) Классификация активов и определение ценности информации
б) Анализ угроз и оценка рисков
в) Выбор и проектирование контрмер и механизмов защиты
г) Определение политик безопасности и требований
д) Реализация и верификация механизмов защиты
Ответ: А-Б-Г-В-Д

П7. Какой стандарт описывает критерии оценки безопасности информационных технологий?
а) ISO 9001

б) Common Criteria (ISO/IEC 15408) ✓
в) ITIL
г) COBIT
Ответ: б) Common Criteria (ISO/IEC 15408)

П8. Какой подход к проектированию предполагает, что безопасность встраивается в архитектуру системы на начальных этапах, а не добавляется постфактум?
а) Security by Obscurity
б) Security after Design
в) Security by Design ✓
г) Security as Option
Ответ: в) Security by Design

П9. Как называется архитектурный принцип, при котором безопасность системы не должна зависеть от сокрытия деталей ее реализации?
Правильный ответ: Отказ от безопасности через неясность (No Security through Obscurity)

П10. Как называется семейство стандартов, посвященных управлению информационной безопасностью?
Правильный ответ: ISO/IEC 27000 series

ПК-3.2.2 Умеет разрабатывать предложения по совершенствованию существующих методов и средств, применяемых для контроля и защиты информации и повышения эффективности этой защиты

Эта компетенция проверяет умение анализировать текущее состояние и предлагать улучшения.

П11. При аудите системы обнаружено, что пароли пользователей хранятся в базе данных в виде открытого текста. Какое основное предложение по совершенствованию защиты вы разработаете?
а) Увеличить минимальную длину пароля
б) Внедрить хэширование паролей с использованием "соли" и адаптивных алгоритмов (bcrypt, Argon2) ✓
в) Скрывать ввод пароля звездочками в интерфейсе
г) Требовать регулярной смены паролей
Ответ: б) Внедрить хэширование паролей с использованием "соли" и адаптивных алгоритмов (bcrypt, Argon2)

П12. Анализ журналов сетевой активности выявил использование устаревших криптографических протоколов (SSLv2, TLS 1.0). Какое усовершенствование будет наиболее эффективным?
а) Блокировка IP-адресов, использующих устаревшие протоколы
б) Принудительное использование только современных защищенных протоколов (TLS 1.2+) ✓
в) Увеличение длины RSA-ключей
г) Шифрование всего трафика на прикладном уровне
Ответ: б) Принудительное использование только современных защищенных протоколов (TLS 1.2+)

П13. Какие предложения по совершенствованию будут эффективны для повышения безопасность существующей системы аутентификации? (Выберите несколько вариантов)
а) Внедрение многофакторной аутентификации (MFA) ✓
б) Увеличение частоты обязательной смены паролей
в) Реализация механизма обнаружения аномалий и подозрительных входов ✓
г) Использование статических паролей повышенной сложности
Правильные ответы: а, в

П14. В существующей системе отсутствует централизованное управление доступом. Какие улучшения вы предложите? (Выберите несколько вариантов)

а) Внедрение системы управления идентификацией и доступом (IAM) ✓

б) Реализация единого входа (Single Sign-On, SSO) ✓

в) Создание отдельных учетных записей для каждого сервиса

г) Регулярный пересмотр и актуализация прав доступа ✓

Правильные ответы: а, б, г

П15. Установите соответствие между выявленной уязвимостью и предлагаемым усовершенствованием:

Выявленная уязвимость	Предлагаемое усовершенствование
1. Отсутствие шифрования конфиденциальных данных в хранилище	а) Внедрение системы обнаружения и предотвращения вторжений (IDS/IPS)
2. Недостаточный мониторинг сетевой активности	б) Внедрение автоматизированного управления исправлениями (Patch Management)
3. Несвоевременное обновление ПО и устранение уязвимостей	в) Реализация сквозного шифрования данных (E2EE) для данных в покое

Правильный ответ: 1-в, 2-а, 3-б

П16. Установите правильную последовательность действий при разработке предложений по совершенствованию защиты:

а) Проведение аудита безопасности и анализ текущего состояния

б) Ранжирование предложений по критериям эффективности, стоимости и сложности внедрения

в) Выявление конкретных уязвимостей и недостатков

г) Формирование перечня конкретных предложений по устранению каждого недостатка

д) Составление дорожной карты (roadmap) по внедрению улучшений

Ответ: А-В-Г-Б-Д

П17. Для повышения эффективности контроля доступа к критическим системам предлагается внедрить:

а) Систему управления привилегированным доступом (PAM) ✓

б) Единый сложный пароль для всех администраторов

в) Запись всех действий администраторов в общий лог-файл

г) Регулярное резервное копирование без проверки целостности

Ответ: а) Систему управления привилегированным доступом (PAM)

П18. Какая метрика НЕ является показателем эффективности системы защиты информации?

а) Время обнаружения инцидента (MTTD)

б) Время реагирования на инцидент (MTTR)

в) Количество сотрудников в отделе информационной безопасности ✓

г) Процент успешно отраженных атак

Ответ: в) Количество сотрудников в отделе информационной безопасности

П19. Как называется процесс систематического улучшения системы защиты информации на основе анализа инцидентов и изменений в угрозах?

Правильный ответ: Непрерывное улучшение системы безопасности (Security Continuous Improvement)

П20. Как называется документ, описывающий план поэтапного улучшения системы защиты информации?

Правильный ответ: Дорожная карта безопасности (Security Roadmap) или План совершенствования СЗИ

ПК-3.2.3 Владеет методами и средствами разработки архитектуры и

интерфейсов систем защиты информации, процедуры восстановления работоспособности средств и систем защиты после сбоев

Эта компетенция проверяет практические навыки создания и восстановления систем защиты.

П21. Какой метод разработки архитектуры системы защиты предполагает создание нескольких независимых уровней контроля?

а) Единая точка защиты

б) Защита в глубину (Defense in Depth) ✓

в) Безопасность через неясность

г) Минималистский подход

Ответ: б) Защита в глубину (Defense in Depth)

П22. При проектировании интерфейса системы управления средствами защиты информации необходимо обеспечить:

а) Максимальную функциональность на одном экране

б) Разделение ролей и предоставление прав в соответствии с принципом наименьших привилегий ✓

в) Яркие цвета и анимацию для удобства восприятия

г) Открытый доступ ко всем функциям для администраторов

Ответ: б) Разделение ролей и предоставление прав в соответствии с принципом наименьших привилегий

Р23. Какие компоненты должны включаться в архитектуру современной системы защиты информации? (Выберите несколько вариантов)

а) SIEM-система для сбора и корреляции событий ✓

б) Система управления уязвимостями ✓

в) Единый суперадминистратор с полным доступом

г) Средства резервного копирования и восстановления ✓

Правильные ответы: а, б, г

П24. Какие процедуры восстановления работоспособности должны быть предусмотрены на случай сбоя системы шифрования? (Выберите несколько вариантов)

а) Автоматический переход на резервный алгоритм шифрования

б) Использование заранее подготовленных резервных ключей ✓

в) Временное отключение шифрования до устранения неисправности

г) Четкое документирование процедур аварийного восстановления ✓

Правильные ответы: б, г

П25. Установите соответствие между компонентом архитектуры системы защиты и его назначением:

Компонент архитектуры	Назначение
1. Подсистема аутентификации и авторизации	а) Обеспечение конфиденциальности и целостности передаваемых данных
2. Подсистема аудита и мониторинга	б) Проверка подлинности субъектов и контроль их доступа к объектам
3. Подсистема криптографической защиты	в) Фиксация событий безопасности и выявление аномальной активности

Правильный ответ: 1-б, 2-в, 3-а

П26. Установите правильную последовательность действий при восстановлении работоспособности системы защиты после сбоя:

а) Локализация и идентификация сбоя

б) Восстановление функционирования критических компонентов

в) Активация резервных средств защиты

г) Полное восстановление работоспособности и проверка эффективности

д) Анализ причин сбоя и внесение корректирующих действий

Ответ: А-Б-В-Г-Д

П27. Какой интерфейс является стандартным для безопасного взаимодействия между компонентами распределенной системы защиты?

а) Веб-интерфейс с базовой аутентификацией

б) Программный интерфейс (API) с использованием аутентификации на основе сертификатов ✓

в) Интерфейс командной строки без аутентификации

г) Графический интерфейс с автоматическим входом

Ответ: б) Программный интерфейс (API) с использованием аутентификации на основе сертификатов

П28. Что такое "горячее резервирование" в контексте средств защиты информации?

а) Резервная копия, хранящаяся в сейфе

б) Автоматическое переключение на дублирующий модуль без прерывания работы ✓

в) Ручное включение резервной системы после сбоя

г) Периодическое создание копий данных

Ответ: б) Автоматическое переключение на дублирующий модуль без прерывания работы

П29. Как называется документ, описывающий порядок действий по восстановлению работоспособности системы защиты после сбоя?

Правильный ответ: План аварийного восстановления (Disaster Recovery Plan) или Процедура восстановления после сбоев

П30. Как называется принцип, согласно которому отказ одного компонента системы защиты не должен приводить к отказу всей системы?

Правильный ответ: Отказоустойчивость (Fault Tolerance) или Принцип единой точки отказа (Avoidance of Single Point of Failure)

Критерии оценки для проведения зачета по дисциплине (тестирование)

Шкала оценивания результатов промежуточной аттестации и критерии выставления оценок.

Оценка	Уровень сформированности компетенции	Критерии
«Отлично» («зачтено»)	Высокий	Выполнено более 80% заданий предложенного теста
«Хорошо» («зачтено»)	Хороший	Выполнено 60-80% заданий предложенного теста
«Удовлетворительно» («зачтено»)	Достаточный	Выполнено 35-60% заданий предложенного теста
«Неудовлетворительно» («не зачтено»)	Недостаточный	Выполнено менее 35% заданий предложенного теста

Обновление оценочных материалов дисциплины

обсуждено и рекомендовано к утверждению решением кафедры наименование кафедры
от ___ _____ 20_ г., протокол № ___

одобрено Научно-методическим советом _____ от ___ _____ 20__ г., протокол №____