МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КОСТРОМСКАЯ ГОСУДАРСТВЕННАЯ СЕЛЬСКОХОЗЯЙСТВЕННАЯ АКАДЕМИЯ»

Утверждаю:
И.о. декана электроэнергетического факультета

Николай Александрович Климов Подписано цифровой подписью: Николай Александрович Климов Дата: 2024.09.11 16:11:55 +03'00' /Климов Н.А./
11 сентября 2024 года

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
по дисциплине

ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Специальность 09.02.07 Информационные системы и программирование

Квалификация выпускника программист

Форма обучения очная

Срок освоения ППССЗ 3 года 10 месяцев

На базе основного общего образования

Караваево 2024

Фонд оценочных средств предназначен для оценивания сформированности компетенций по дисциплине «Технология разработки программного обеспечения».

Разработчик:
Преподаватель А.А. Лобачев Андрей Александрович Лобачев
Подписано цифровой подписью: Андрей Александрович Лобачев
Дата: 2024.09.05 14:06:00 +03'00'

Утвержден на заседании кафедры СПО-Тракторы и автомобили, протокол №1 от 05.09.2024

Заведующий кафедрой А.М. Молодов Александр Михайлович Молодов
Подписано цифровой подписью: Александр Михайлович Молодов
Дата: 2024.09.05 14:30:36 +03'00'

Согласовано:
Председатель методической комиссии электроэнергетического факультета

А.С. Яблоков Алексей Сергеевич Яблоков
Подписано цифровой подписью: Алексей Сергеевич Яблоков
Дата: 2024.09.10 15:13:43 +03'00'
протокол № 7 от 10.09.2024

Результаты освоения дисциплины

«Технология разработки программного обеспечения»

ППССЗ (СПО) по специальности:

09.02.07 Информационные системы и программирование

Коды компетенций по ФГОС	Компетенции	Результат освоения
Общие компетенции		
ОК 02	Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности	**Знать** формат оформления результатов поиска информации, порядок применения современных средств и устройств информатизации, как применять программное обеспечение в профессиональной деятельности, в том числе с использованием цифровых средств. **Уметь** оформлять результаты поиска, применять средства информационных технологий для решения профессиональных задач; планировать процесс поиска; структурировать получаемую информацию; оформлять результаты поиска информации, пользоваться современными средствами поиска информатизации. **Владеть** навыками оформления результатов поиска информации; навыками планирования процесса поиска и структурирования полученной информации.
Профессиональные компетенции		
ПК 2.1	Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент	**Знать** современные технологии и инструменты интеграции; основные протоколы доступа к данным; виды и варианты интеграционных решений; методы и способы идентификации сбоев и ошибок при интеграции приложений; основы организации инспектирования и верификации; методы отладочных классов; **Уметь** анализировать проектную и техническую документацию; определять источники и приемники данных; выполнять отладку, используя методы и инструменты условной компиляции (классы debug и trace); использовать специализированные графические средства построения и анализа архитектуры программных продуктов; оценивать размер минимального набора тестов; выявлять ошибки в системных компонентах на основе спецификаций; организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов; проводить сравнительный анализ; разрабатывать тестовые пакеты и тестовые сценарии **Владеть** навыками разработки и оформления требований к программным модулям по предложенной документации; навыками разработки тестовые наборы (пакеты) для программного модуля; разработки тестовые сценарии программного средства; навыками

		инспектирования разработанных программных модулей на предмет соответствия стандартам кодирования
ПК 2.2	Выполнять интеграцию модулей в программное обеспечение	**Знать** модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации программного обеспечения; **Уметь** использовать выбранную систему контроля версий; использовать методы для получения кода с заданной функциональностью и степенью качества; организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов; использовать различные транспортные протоколы и стандарты форматирования сообщений; выполнять тестирование интеграции; организовывать постобработку данных; создавать классы- исключения на основе базовых классов; выполнять ручное и автоматизированное тестирование программного модуля; выявлять ошибки в системных компонентах на основе спецификаций; использовать приемы работы в системах контроля версий **Владеть** навыками интегрирования модулей в программное обеспечение; навыками отладки программных модулей; навыками инспектирования разработанных программных модулей на предмет соответствия стандартам кодирования
ПК 2.3	Выполнять отладку программного модуля с использованием специализированных программных средств	**Знать** стандарты качества программной документации; встроенные и основные специализированные инструменты анализа качества программных продуктов; графические средства проектирования архитектуры программных продуктов; методы организации работы в команде разработчиков **Уметь** использовать выбранную систему контроля версий; использовать методы для получения кода с заданной функциональностью и степенью качества; анализировать проектную и техническую документацию; использовать инструментальные средства отладки программных продуктов; определять источники и приемники данных; выполнять тестирование интеграции; организовывать постобработку данных; использовать приемы работы в системах контроля версий; выполнять отладку, используя методы и инструменты условной компиляции; выявлять ошибки в системных компонентах на основе спецификаций **Владеть** навыками отладки программных модулей; инспектирования разработанных программных модулей на предмет соответствия стандартам кодирования

Паспорт
фонда оценочных средств

Таблица 1

№ п/п	Контролируемые дидактические единицы	Контролируемые компетенции (или их части)	Наименование оценочных средств		
			Тесты, кол-во заданий	Другие оценочные средства	
				вид	кол-во заданий
1	Тема 1. Введение в технологии разработки программных средств	ОК 02 ПК 2.1 ПК 2.2 ПК 2.3	20	Опрос	5
2	Тема 2Стратегии разработки программных средств и систем и реализующие их модели жизненного цикла	ОК 02 ПК 2.1 ПК 2.2 ПК 2.3	20	Опрос	5
3	Тема 3. Выбор модели жизненного цикла для конкретного проекта	ОК 02 ПК 2.1 ПК 2.2 ПК 2.3	20	Опрос	5
4	Тема 4Классические методологии разработки программных средств	ОК 02 ПК 2.1 ПК 2.2 ПК 2.3	20	Опрос	5
5	Тема 5. CASE-технологии структурного анализа и проектирования программных средств	ОК 02 ПК 2.1 ПК 2.2 ПК 2.3	20	Опрос	5
6	Тема 6. Методология объектно-ориентированного анализа и проектирования сложных систем	ОК 02 ПК 2.1 ПК 2.2 ПК 2.3	20	Опрос	5
7	Темы 1-6	ОК 01 ОК 02 ПК 1.1 ПК 1.2 ПК 1.6		Собеседование	20
Всего:			120		50

Методика проведения контроля по проверке базовых знаний по дисциплине «Технология разработки программного обеспечения»

Тема 1 Введение в технологии разработки программных средств

Контролируемые компетенции (знания, умения) ОК 02 ПК 2.1 ПК 2.2 ПК 2.3

Вопросы для устного опроса

1. Объясните концепцию “жизненного цикла ПО” (Software Development Life Cycle, SDLC) и его этапы. Какие модели SDLC вы знаете (например, “водопадная модель”, “итеративная модель”)? Сравните преимущества и недостатки различных моделей SDLC и объясните, какая модель лучше подходит для разработки конкретного типа программного обеспечения.
2. Как можно использовать методологии разработки программного обеспечения (например, Agile, Scrum) для управления проектами? Какие принципы и практики используются в этих методологиях? В каких ситуациях использование Agile или Scrum является более эффективным по сравнению с традиционными методами управления проектами?
3. Опишите концепцию “архитектуры программного обеспечения” (Software Architecture). Какие типы архитектур вы знаете (например, “клиент-серверная архитектура”, “микросервисная архитектура”)? Сравните преимущества и недостатки различных архитектурных стилей и объясните, как выбрать подходящую архитектуру для разработки конкретного типа программного обеспечения.
4. Как можно использовать “инструменты моделирования программного обеспечения” (Software Modeling Tools) для проектирования и разработки программ? Какие типы моделей используются при моделировании программного обеспечения (например, UML диаграммы)? Как можно использовать модели для упрощения разработки и повышения качества программного обеспечения?
5. Объясните, как “тестирование программного обеспечения” (Software Testing) играет важную роль в жизненном цикле ПО. Какие типы тестирования вы знаете (например, “модульное тестирование”, “интеграционное тестирование”, “системное тестирование”)? Как провести тестирование и как оценить результаты тестирования?

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который прочно усвоил программный материал в полном объеме, исчерпывающе, грамотно и логически стройно его излагает, четко формулирует основные понятия, приводит соответствующие примеры, уверенно владеет материалом.

Оценка «хорошо» выставляется обучающемуся, который твердо усвоил программный материал, грамотно и по существу излагает его без существенных ошибок, правильно применяет теоретические положения при решении конкретных задач, с небольшими погрешностями приводит формулировки определений, по ходу изложения допускает небольшие пробелы, не искажающие содержания ответа.

Оценка «удовлетворительно» выставляется обучающемуся, который не совсем твердо владеет программным материалом, знает основные теоретические положения изучаемой темы, при ответах допускает малосущественные погрешности, искажения логической последовательности при изложении материала, неточную аргументацию теоретических положений, испытывает затруднения при ответе на дополнительные вопросы.

Оценка «неудовлетворительно» выставляется обучающемуся, имеющему серьезные пробелы в знании учебного материала, допускающему принципиальные ошибки при ответе на вопросы.

Тестовые задания

Выберите один правильный вариант ответа и нажмите кнопку «Далее»

Определение жизненного цикла программного обеспечения подразумевает:
+ Этапы разработки, эксплуатации и сопровождения ПО
Непрерывный процесс без фиксированных этапов
Только процессы разработки без учета тестирования
Строгую последовательность изменений кода

В стадии проектирования программного обеспечения создается:
+ Архитектура и спецификации системы
Код программы и ее документация
Тестовые планы и сценарии
Полноценный рабочий продукт

Метод Agile отличается от традиционных методологий тем, что:
+ Позволяет гибко реагировать на изменения требований
Всегда завершает проект в заранее установленный срок
Не включает тестирование на каждом этапе
Использует строгое планирование всех этапов

Стадия эксплуатации программного обеспечения включает:
+ Операцию программы пользователями в реальном времени
Проектирование новых функций для улучшений
Обучение пользователей работе с ПО
Полное обновление системы на новой платформе

Документация жизненного цикла ПО должна содержать:
+ Подробные инструкции по использованию и технические требования
Только код и его описание
Личную информацию о разработчиках
Рекомендации по выбору оборудования

Стадия тестирования ПО предназначена для:
+ Обнаружения и исправления ошибок в программе
Записи всех действий пользователей
Запланирования новых этапов разработки
Создания окончательного отчета о проекте

Модель "водопада" предполагает:
+ Последовательное завершение каждой стадии перед началом следующей
Параллельное выполнение всех стадий разработки
Обратный процесс, где тестирование предшествует разработке
Частые изменения требований в процессе разработки

Система контроля версий нужна для:
+ Отслеживания изменений в коде и совместной работы разработчиков
Снижения стоимости разработки ПО
Упрощения процесса обучения пользователей
Ускорения завершения проекта

Понятие "рефакторинг" подразумевает:
+ Улучшение структуры кода без изменения его функциональности
Полную переработку архитектуры системы
Устранение всех ошибок в процессе разработки
Установку новых функций и возможностей

Управление проектом включает в себя:
+ Планирование, организацию и контроль всех этапов разработки
Только техническое управление командой разработчиков
Исключительно финансовые аспекты реализации проекта
Самостоятельное выполнение всех задач без контроля

Использование системы тикетов в разработке ПО позволяет:
+ Упорядочивать запросы и отслеживать их статус
Игнорировать требования пользователей
Увеличивать количество задач без обратной связи
Усложнять коммуникацию внутри команды

Минимально жизнеспособный продукт (MVP) — это:
+ Продукт с базовыми функциями, предназначенный для проверки гипотезы
Полноценный продукт с множеством возможностей
Программа, тестирующая поведение пользователей
Программное обеспечение, разрабатываемое без обратной связи

Тестирование на этапе проектирования необходимо для:
+ Проверки реалистичности требований и функциональности
Полной заведомой готовности к выпуску продукта
Снижения времени разработки без тестирования
Обеспечения письменной документации для пользователей

Управление рисками в проекте предполагает:
+ Идентификацию, анализ и минимизацию потенциальных проблем
Устранение всех ошибок до начала проекта
Обесценивание всех ресурсов проекта
Исключение никаких дополнительных затрат во время разработки

Стадия поддержки включает в себя:
+ Обновление и исправление программы по мере необходимости
Идентификацию новых требований от пользователей
Удаление старых версий ПО
Тестирование новых функций на потребителях

Процесс совершенствования ПО включает:
+ Анализ обратной связи и внедрение улучшений
Полное переписывание кода из-за ошибок
Упрощенное тестирование без документирования
Сокращение функциональности для уменьшения затрат

Жизненный цикл ПО может содержать принципы:
+ Исследования, разработки, тестирования и поддержки
Только разработки и завершения проекта
Лишь этапы дизайна и тестирования

Исключительно маркетинга и продажи

Важность анализа требований заключается в:
+ Понимании нужд пользователей и формировании плана разработки
Быстром запуске продукта на рынок
Игнорировании обратного потока от клиентов
Создании нового программного обеспечения без анализирования

Определение жизненного цикла программного обеспечения включает в себя:
+ Этапы разработки, тестирования, внедрения и поддержки
Процессы, которые никогда не прекращаются
Только алгоритмы и код, без учета пользователя
Исключительно этапы проектирования и маркетинга

Модель "спирали" отличается от традиционных моделей тем, что:
+ Включает итеративное развитие и управление рисками
Является линейной и строго последовательной
Проект не требует обновлений после завершения
Не позволяет вносить изменения в требования на любом этапе

Методика проведения контроля

Параметры методики	Значение параметра
Предел длительности всего контроля	10 минут
Последовательность выбора вопросов	Случайная
Предлагаемое количество вопросов	10

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который правильно выполнил 9-10 тестовых заданий.

Оценка «хорошо» выставляется обучающемуся, который: правильно выполнил 7-8 тестовых заданий.

Оценка «удовлетворительно» выставляется обучающемуся, который правильно выполнил 5-6 тестовых заданий.

Оценка «неудовлетворительно» выставляется обучающемуся, который правильно выполнил менее 4 тестовых заданий.

Тема 2 Стратегии разработки программных средств и систем и реализующие их модели жизненного цикла

Контролируемые компетенции (знания, умения) ОК 02 ПК 2.1 ПК 2.2 ПК 2.3

Вопросы для устного опроса

1. Объясните концепцию “агильной разработки” (Agile Development) и её основные принципы. Как модель “Scrum” реализует агильную стратегию разработки? Сравните Scrum с “Kanban” и объясните, какая модель лучше подходит для разработки проекта с высокой степенью неопределенности и частыми изменениями требований.
2. Опишите “водопадную модель” жизненного цикла ПО (Waterfall Model) и её основные этапы. Какие преимущества и недостатки имеет водопадная модель? В каких ситуациях она является более подходящим решением по сравнению с агильными методами?
3. Как можно интегрировать “DevOps” в жизненный цикл ПО? Какие этапы жизненного цикла необходимо модифицировать для включения DevOps-практик? Какие преимущества и недостатки имеет DevOps в контексте разработки и поддержки ПО?
4. Объясните, как “микросервисная архитектура” (Microservices Architecture) может быть реализована с использованием агильной разработки. Какие преимущества и недостатки имеет микросервисная архитектура в контексте агильных методов? Какие инструменты и технологии могут быть использованы для реализации микросервисной архитектуры?
5. Какие методы и инструменты могут быть использованы для “управления рисками” (Risk Management) в разработке программного обеспечения? Как можно определить, оценить и снизить риски, связанные с разработкой ПО? Какие стратегии управления рисками подходят для различных моделей жизненного цикла ПО?

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который прочно усвоил программный материал в полном объеме, исчерпывающе, грамотно и логически стройно его излагает, четко формулирует основные понятия, приводит соответствующие примеры, уверенно владеет материалом.

Оценка «хорошо» выставляется обучающемуся, который твердо усвоил программный материал, грамотно и по существу излагает его без существенных ошибок, правильно применяет теоретические положения при решении конкретных задач, с небольшими погрешностями приводит формулировки определений, по ходу изложения допускает небольшие пробелы, не искажающие содержания ответа.

Оценка «удовлетворительно» выставляется обучающемуся, который не совсем твердо владеет программным материалом, знает основные теоретические положения изучаемой темы, при ответах допускает малосущественные погрешности, искажения логической последовательности при изложении материала, неточную аргументацию теоретических положений, испытывает затруднения при ответе на дополнительные вопросы.

Оценка «неудовлетворительно» выставляется обучающемуся, имеющему серьезные пробелы в знании учебного материала, допускающему принципиальные ошибки при ответе на вопросы.

Тестовые задания

Выберите один правильный вариант ответа и нажмите кнопку «Далее»

Основной этап в каскадной стратегии разработки программных средств называется:

+ Анализ требований

Системная интеграция

Тестирование и отладка

Подготовка документации

Классическая каскадная модель включает в себя:
Тестирование на этапе разработки
+ Последовательное выполнение этапов
Непредсказуемое изменение требований
Разработку без четких планов

Отличие каскадной модели с обратными связями от классической заключается в :
+ Наличие возможности возврата к предыдущим этапам
Отсутствии четкой документации
Полной независимости этапов
Игнорировании обратной связи

Определение V-образной модели:
Модель, в которой все этапы выполняются параллельно
+ Комбинация каскадной и жизненного цикла тестирования
Подход, игнорирующий заключительные тесты
Метод, основанный на случайных изменениях требований

Основной принцип базовой модели RAD:
Широкое планирование требований
+ Быстрая разработка с использованием прототипов
Фиксация требований на начальном этапе
Поэтапная реализация функций

RAD-модель, основанная на моделировании предметной области, акцентирует внимание на:
Устойчивом изменении требований
+ Динамическом создании моделей
Фиксированном наборе функций
Линейном выполнении задач

Особенности инкрементной модели с уточнением требований:
+ Непрерывное уточнение и дополнение требований
Замораживание требований на старте
Недостаток тестирования
Полное отсутствие рассмотра пользовательского опыта

Основные преимущества инкрементной модели по ГОСТ Р ИСО/МЭК:
Отсутствие необходимости в тестировании
+ Гибкость и адаптация к изменяющимся требованиям
Жесткое планирование этапов
Игнорирование пользовательского мнения

Эволюционная стратегия разработки программного обеспечения предполагает:
Преднамеренные жесткие изменения
+ Пошаговое улучшение продукта
Отсутствие удовлетворения пользователей
Игнорирование возвращающейся обратной связи

Основной элемент структурной эволюционной модели быстрого прототипирования:
Полное игнорирование пользователей
+ Интерактивное создание и тестирование прототипов
Фиксированные этапы разработки
Полная забывчивость об ранних версиях

Спиральная модель Боэма включает в себя:
Линейную последовательность этапов
+ Итеративный процесс с улучшением
Отсутствие оценки рисков
Игнорирование всех процессов тестирования

Упрощенные спиральные модели ориентированы на:
Подробное документирование всех этапов
+ Быстрое внедрение в условиях риска
Полное отсутствие анализа рисков
Прогнозирование по формальным методам

Основной недостаток моделей быстрой разработки ПО:
Низкое качество конечного продукта
+ Ограниченное время на анализ требований
Постоянные изменения в команде разработчиков
Игнорирование проверок и тестов

Задачи инкрементной модели завершаются:
+ Поэтапным добавлением функционала
Полным завершением всех этапов развития
Непредсказуемым завершением
Непрерывной поставкой изменений

Основная цель эволюционных моделей по ГОСТ Р ИСО/МЭК:
Многоуровневая документация
+ Понимание и адаптация к изменениям
Игнорирование полного цикла разработки
Фиксация всех требований на старте

Сущность каскадной модели по ГОСТ Р ИСО/МЭК ТО 15271-2002 заключается в:
Стремлении к постоянным изменениям
+ Четкой последовательности фаз
Тестировании на каждом этапе по желанию
Полной независимости этапов разработки

Неполное исследование требований в RAD-моделях приводит к:
+ Рискам потери данных
Тщательной проработке запросов
Стремительному завершению проекта
Верной реализацией всех функций

Один из основных принципов спиральной модели заключается в:
Оптимизации затрат на разработку
+ Постоянном анализе и оценке рисков
Отсутствии обсуждений с клиентом
Игнорировании требований пользователя

Инкрементная модель экстремального программирования ориентирована на:
Полномасштабные изменения требований
+ Частые релизы и активное сотрудничество с заказчиком
Наиболее низкое качество продукта
Долгосрочное планирование без оценки рисков

Основная характеристика RAD-модели параллельной разработки ПО:
Индивидуальное выполнение всех этапов
+ Синхронизация разработки и тестирования
Четкий контроль всех этапов
Фиксированная структура команды

Методика проведения контроля

Параметры методики	Значение параметра
Предел длительности всего контроля	10 минут
Последовательность выбора вопросов	Случайная
Предлагаемое количество вопросов	10

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который правильно выполнил 9-10 тестовых заданий.

Оценка «хорошо» выставляется обучающемуся, который: правильно выполнил 7-8 тестовых заданий.

Оценка «удовлетворительно» выставляется обучающемуся, который правильно выполнил 5-6 тестовых заданий.

Оценка «неудовлетворительно» выставляется обучающемуся, который правильно выполнил менее 4 тестовых заданий.

Тема 3 Выбор модели жизненного цикла для конкретного проекта

Контролируемые компетенции (знания, умения) ОК 02 ПК 2.1 ПК 2.2 ПК 2.3

Вопросы для устного опроса

1. Опишите, как вы бы определили “уровень неопределенности” (level of uncertainty) проекта разработки программного обеспечения. Какие факторы влияют на уровень неопределенности? Как уровень неопределенности влияет на выбор модели жизненного цикла ПО (например, “водопадная модель”, “агильная модель”)?

2. Сравните и проанализируйте “водопадную модель” (Waterfall Model) и “агильную модель” (Agile Model) с точки зрения “управления рисками” (risk management). Какие риски характерны для каждой модели? Какие методы управления рисками подходят для каждой модели?

3. Как “размер проекта” (project size) влияет на выбор модели жизненного цикла ПО? Какие модели подходят для масштабных проектов с большим количеством участников и сложной архитектурой? Какие модели лучше подходят для небольших проектов с ограниченным бюджетом и сроками?

4. Объясните, как “характер заказчика” (client characteristics) может влиять на выбор модели жизненного цикла. Какие модели подходят для заказчиков с высокой степенью вовлеченности в проект и частыми изменениями требований? Какие модели подходят для заказчиков с низкой степенью вовлеченности и четкими требованиями?

5. Какие “инструменты и технологии” (tools and technologies) доступны для реализации разных моделей жизненного цикла? Как доступные инструменты и технологии могут влиять на выбор модели жизненного цикла? Приведите примеры инструментов и технологий, которые подходят для разных моделей жизненного цикла.

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который прочно усвоил программный материал в полном объеме, исчерпывающе, грамотно и логически стройно его излагает, четко формулирует основные понятия, приводит соответствующие примеры, уверенно владеет материалом.

Оценка «хорошо» выставляется обучающемуся, который твердо усвоил программный материал, грамотно и по существу излагает его без существенных ошибок, правильно применяет теоретические положения при решении конкретных задач, с небольшими погрешностями приводит формулировки определений, по ходу изложения допускает небольшие пробелы, не искажающие содержания ответа.

Оценка «удовлетворительно» выставляется обучающемуся, который не совсем твердо владеет программным материалом, знает основные теоретические положения изучаемой темы, при ответах допускает малосущественные погрешности, искажения логической последовательности при изложении материала, неточную аргументацию теоретических положений, испытывает затруднения при ответе на дополнительные вопросы.

Оценка «неудовлетворительно» выставляется обучающемуся, имеющему серьезные пробелы в знании учебного материала, допускающему принципиальные ошибки при ответе на вопросы.

Тестовые задания

Выберите один правильный вариант ответа и нажмите кнопку «Далее»

Классификация проектов по разработке программного обеспечения не включает в себя:

+ Проекты, связанные с финансами и инвестированием
Проекты на основе заказов клиентов
Внутренние проекты для создания новых инструментов
Проекты по созданию мобильных приложений

Основной критерий выбора модели жизненного цикла разработки ПО:

+ Характеристики проекта и специфические требования
Только требования заказчика к функциональности
Опыт команды разработчиков
Наличие бюджета для проекта

Специфика адаптации модели жизненного цикла определяется:

+ Уникальными условиями и требованиями конкретного проекта
Универсальными стандартами разработки программного обеспечения
Только технологическим стеком, используемым в проекте
Мировыми практиками в области управления проектами

Основной этап при использовании Rational Rose заключается в:

+ Моделировании объектов и их взаимодействий
Разработке внутреннего кода приложения
Подборе аппаратного обеспечения для проекта
Обучении пользователей работе с программным обеспечением

Методология Agile подразумевает:

+ Итеративный подход к разработке с частыми внедрениями
Полное завершение разработки перед тестированием
Использование только одного большого цикла разработки
Отсутствие документации на всех этапах проекта

В проекте с высоким уровнем неопределенности предпочтительнее использовать:

+ Гибкие и адаптивные модели разработки
Линейную модель с четко фиксированными этапами
Набор стандартных подходов без изменений
Программы управления проектами без гибкости

Одним из критериев для выбора модели жизненного цикла является:

+ Объем и масштаб проекта
Только опыт команды разработки
Бюджет без учета временных рамок
Предпочтения заказчика по форме отчетности

Основное преимущество Star UML заключается в:

+ Поддержке различных диаграмм и языков моделирования
Ограниченном наборе функций для проектирования
Простоте использования без глубокого обучения
Неподходящем интерфейсе для работы в команде

Наиболее часто используемая модель жизненного цикла для крупных проектов:
+ Водопадная модель
Модель спирали без анализа рисков
Модель быстрой разработки
Методология SCRUM без четких планов

Адаптация модели жизненного цикла необходима для:
+ Оптимизации под потребности проекта и команды
Полного отказа от любых изменений в процессах
Строгого следования международным стандартам
Перехода к модели с фиксированными сроками

Одной из характеристик бенчмаркинга является:
+ Сравнение с аналогичными проектами для выявления лучших практик
Оценка только внутренних процессов разработки
Анализ финансовых потоков компании
Сравнение команд разработчиков по количеству сотрудников

Этап планирования жизненного цикла ПО включает:
+ Определение требований и оценку рисков
Только финансовое планирование и бюджетирование
Подбор команды разработчиков
Исключительно маркетинговый анализ проекта

В модели спирали акцент делается на:
+ Управлении рисками и непрерывной доработке продукта
Жестком следовании заранее установленному плану
Полном завершении каждого этапа перед началом следующего
Откате к предыдущим версиям без анализа

Система Star UML позволяет создавать:
+ Диаграммы классов, состояния, деятельности и др.
Только общие схемы бизнеса и управления
Линейные документы и отчеты для заказчиков
Графики временных затрат на проект

Процесс разработки программного обеспечения включает в себя:
+ Определение требований, проектирование, кодирование и тестирование
Исключительно тестирование готового продукта
Только этапы анализа и подготовки документации
Подбор инструментария и формирование команды

Ключевой элемент оценки сложных проектов — это:
+ Учет всех неопределенностей и рисков, связанных с проектом
Строгое следование только одному плану
Полное игнорирование внешних факторов
Сравнение только по стоимостью проектов

При разработке ПО, основанного на Agile, главной задачей является:
+ Быстрое реагирование на изменения требований
Строгое следование проверенному плану
Оценка только по готовности проекта
Меньшее внимание к взаимодействию с клиентом

Модель V-образного жизненного цикла отличается тем, что:
+ Тестирование и валидация происходят параллельно с разработкой системы
Все этапы разработки строго линейные и строго прописанные
На каждом этапе идет рефакторинг кода
Нет необходимости в планировании тестирования заранее

Исходная документация в проекте будет включать в себя:
+ Требования заказчика, технические характеристики и планы
Только акт приемки и тестирования
Протоколы встречи с клиентом
Задания для тестировщиков без описания требований

При классификации проектов по разработке ПО важным аспектом является:
+ Определение цели и потребностей проекта
Только срок выполнения проекта
Предпочтения команды разработки
Общая стоимость разработки всех проектов

Методика проведения контроля

Параметры методики	Значение параметра
Предел длительности всего контроля	10 минут
Последовательность выбора вопросов	Случайная
Предлагаемое количество вопросов	10

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который правильно выполнил 9-10 тестовых заданий.

Оценка «хорошо» выставляется обучающемуся, который: правильно выполнил 7-8 тестовых заданий.

Оценка «удовлетворительно» выставляется обучающемуся, который правильно выполнил 5-6 тестовых заданий.

Оценка «неудовлетворительно» выставляется обучающемуся, который правильно выполнил менее 4 тестовых заданий.

Тема 4 Классические методологии разработки программных средств

Контролируемые компетенции (знания, умения) ОК 02 ПК 2.1 ПК 2.2 ПК 2.3

Вопросы для устного опроса

1. Объясните концепцию “водопадной модели” (Waterfall Model) жизненного цикла ПО. Какие этапы включает водопадная модель? Какие преимущества и недостатки имеет водопадная модель по сравнению с более современными методологиями?

2. Сравните и проанализируйте “структурное программирование” (Structured Programming) и “объектно-ориентированное программирование” (Object-Oriented Programming). В чем их основные отличия? В каких ситуациях каждый из подходов является более подходящим решением? Какие инструменты и технологии используются для реализации структурного и объектно-ориентированного программирования?

3. Как можно использовать “методы формальной верификации” (Formal Verification Methods) в разработке программного обеспечения? Какие типы формальных методов вы знаете? Как формальные методы могут помочь в обеспечении корректности и безопасности программного обеспечения?

4. Объясните концепцию “модульного тестирования” (Unit Testing) в контексте классических методологий разработки программного обеспечения. Как провести модульное тестирование? Какие инструменты используются для автоматизации модульного тестирования? Какую роль играет модульное тестирование в обеспечении качества программного обеспечения?

5. Сравните и проанализируйте “методы проектирования программного обеспечения” (Software Design Methods), например, “структурное проектирование” (Structured Design) и “объектно-ориентированное проектирование” (Object-Oriented Design). В чем их основные отличия? В каких ситуациях каждый из методов является более подходящим решением? Какие инструменты и технологии используются для реализации структурного и объектно-ориентированного проектирования?

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который прочно усвоил программный материал в полном объеме, исчерпывающе, грамотно и логически стройно его излагает, четко формулирует основные понятия, приводит соответствующие примеры, уверенно владеет материалом.

Оценка «хорошо» выставляется обучающемуся, который твердо усвоил программный материал, грамотно и по существу излагает его без существенных ошибок, правильно применяет теоретические положения при решении конкретных задач, с небольшими погрешностями приводит формулировки определений, по ходу изложения допускает небольшие пробелы, не искажающие содержания ответа.

Оценка «удовлетворительно» выставляется обучающемуся, который не совсем твердо владеет программным материалом, знает основные теоретические положения изучаемой темы, при ответах допускает малосущественные погрешности, искажения логической последовательности при изложении материала, неточную аргументацию теоретических положений, испытывает затруднения при ответе на дополнительные вопросы.

Оценка «неудовлетворительно» выставляется обучающемуся, имеющему серьезные пробелы в знании учебного материала, допускающему принципиальные ошибки при ответе на вопросы.

Тестовые задания

Выберите один правильный вариант ответа и нажмите кнопку «Далее»

Принцип инкапсуляции в структурном программировании подразумевает:
+ Защиту данных и функций от внешнего воздействия
Полное открытие всех деталей реализации
Упрощение структуры программы путем удаления модулей
Продление времени выполнения программы

Модульное проектирование позволяет:
+ Разделить программу на самостоятельные части для упрощения разработки
Создать один большой блок кода для повышения производительности
Усложнить структуру кода для повышения безопасности
Исключить использование функций и процедур

Метод восходящего проектирования (bottom-up) начинается с:
+ Уровня деталей и создания модулей для интеграции
Определения общего алгоритма и его последовательной реализации
Непосредственного кодирования приложения
Проектирования интерфейсов пользователя в первую очередь

Другим способом расширения программного ядра является:
+ Использование библиотек и API для добавления новых функций
Упрощение кода за счет его полного удаления
Повышение статической составляющей программы
Ограничение функциональных возможностей системы

Метод Джексона рекомендует:
+ Рекурсивное дробление задач на более мелкие подзадачи
Игнорирование взаимодействий между модулями
Создание большого количества взаимозависимых модулей
Проектирование на основе временных последовательностей

Оценка структурного разбиения ПС учитывает:
+ Соответствие модулей их предназначению и взаимодействиям
Только количество строк кода в каждом модуле
Сложность алгоритмов, используемых в программе
Упрощение интерфейсов для пользователя

Основная цель структурного программирования заключается в:
+ Улучшении читаемости и сопровождения кода
Увеличении общего количества кода в проекте
Создании максимально сложной архитектуры
Невозможности повторного использования функций

Модульное проектирование эффективно снижает:
+ Комплексность системы и облегчает её тестирование
Время, необходимое для выполнения программы
Потребности в обучении разработчиков
Количество модулей в проекте

Одним из приоритетных аспектов при выборе методов проектирования является:
+ Учитывание требований к надежности и производительности
Исключительное следование самым последним тенденциям
Использование устаревших подходов для экономии времени
Игнорирование потребностей конечного пользователя

При восходящем проектировании важно:
+ Начать с малых компонент и поэтапно интегрировать их
Работать только с высокоуровневыми абстракциями
Обеспечить полное соответствие всего кода стандартам
Сосредоточиться на создании документации в первую очередь

Метод расширения ядра направлен на:
+ Расширение функциональности существующих программ
Полное переписывание программы с нуля
Упрощение модульной структуры проекта
Сокращение числа взаимодействий между модулями

При оценке разбиения системы полезно учитывать:
+ Наличие четких границ между модулями
Лишь общую архитектуру без упоминания деталей
Тяжесть каждого модуля в системе
Устойчивость к изменениям в непосредственно пользовательском интерфейсе

Принципы модульного проектирования предполагают:
+ Разделение функций на логически связанные группы
Слияние всех функций в одну общую
Создание зависимостей между модулями
Использование крупных функций без видимого разделения

Структурное программирование эффективно помогает:
+ Упорядочить код и снизить вероятность ошибок
Загромоздить проект неуправляемым количеством функций
Создать трудные для понимания алгоритмы
Исключить комментарии для повышения скорости выполнения

Основными характеристиками структурного программирования являются:
+ Ясность структурирования и отсутствие спагетти-кода
Сложные взаимодействия между модулями
Полная свобода в написании кода без каких-либо стандартов
Неразделение логики и данных

Метод Юнга (Young's method) используется для:
+ Упрощения проектирования путем разбиения на модули
Усложнения замысла проекта
Создания произвольного количества инструкций
Объединения нескольких языков программирования в один

Методы восходящего проектирования популярны в ситуациях, когда:
+ Требуются глубокие знания о сущностях и их взаимодействиях
Необходимо создать простую и однозначную систему
Используются только рутинные задачи
Полная зависимость от сторонних библиотек

При расширении ядра программного обеспечения можно учитывать:
+ Существующие модули и их возможности
Только новые, еще не реализованные модули
Никакие модули не учитываются
Запросы от клиентов без анализа системы

Структурирование программного кода по принципу дерева позволяет:
+ Легче отслеживать зависимости и взаимодействия
Создавать спагетти-код с многими зависимостями
Упрощать процесс компиляции и компоновки
Увеличивать размер плагинов

Методы Джексона упрощают:
+ Проектирование систем и взаимодействие модулей
Наблюдение за кодом с использованием диаграмм
Подбор средств программирования
Небольшие команды, работающие над проектом

Методика проведения контроля

Параметры методики	Значение параметра
Предел длительности всего контроля	10 минут
Последовательность выбора вопросов	Случайная
Предлагаемое количество вопросов	10

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который правильно выполнил 9-10 тестовых заданий.

Оценка «хорошо» выставляется обучающемуся, который: правильно выполнил 7-8 тестовых заданий.

Оценка «удовлетворительно» выставляется обучающемуся, который правильно выполнил 5-6 тестовых заданий.

Оценка «неудовлетворительно» выставляется обучающемуся, который правильно выполнил менее 4 тестовых заданий.

Тема 5 CASE-технологии структурного анализа и проектирования программных средств

Контролируемые компетенции (знания, умения) ОК 02 ПК 2.1 ПК 2.2 ПК 2.3

Вопросы для устного опроса

1. Объясните концепцию “CASE-технологий” (Computer-Aided Software Engineering) и их роль в структурном анализе и проектировании программного обеспечения. Какие типы CASE-инструментов вы знаете (например, инструменты для моделирования, инструменты для генерации кода, инструменты для тестирования)? Сравните преимущества и недостатки использования CASE-инструментов.
2. Как можно использовать “диаграммы UML” (Unified Modeling Language) в структурном анализе и проектировании? Какие типы диаграмм UML подходят для представления структуры программы и ее поведения? Как можно использовать UML-диаграммы для генерации кода и тестирования программного обеспечения?
3. Опишите концепцию “структурного проектирования” (Structured Design) и её основные принципы. Какие методы структурного проектирования вы знаете (например, “метод функционального разложения”, “метод структурных диаграмм”)? Как можно использовать CASE-инструменты для реализации структурного проектирования?
4. Объясните, как “методы формальной верификации” (Formal Verification Methods) могут быть интегрированы с CASE-инструментами для повышения качества и безопасности программного обеспечения. Какие типы формальных методов можно использовать в сочетании с CASE-инструментами?
5. Сравните и проанализируйте преимущества и недостатки использования CASE-технологий в разработке программного обеспечения. В каких ситуациях использование CASE-инструментов является оправданным, а в каких – более эффективным решением будет традиционный подход к разработке?

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который прочно усвоил программный материал в полном объеме, исчерпывающе, грамотно и логически стройно его излагает, четко формулирует основные понятия, приводит соответствующие примеры, уверенно владеет материалом.

Оценка «хорошо» выставляется обучающемуся, который твердо усвоил программный материал, грамотно и по существу излагает его без существенных ошибок, правильно применяет теоретические положения при решении конкретных задач, с небольшими погрешностями приводит формулировки определений, по ходу изложения допускает небольшие пробелы, не искажающие содержания ответа.

Оценка «удовлетворительно» выставляется обучающемуся, который не совсем твердо владеет программным материалом, знает основные теоретические положения изучаемой темы, при ответах допускает малосущественные погрешности, искажения логической последовательности при изложении материала, неточную аргументацию теоретических положений, испытывает затруднения при ответе на дополнительные вопросы.

Оценка «неудовлетворительно» выставляется обучающемуся, имеющему серьезные пробелы в знании учебного материала, допускающему принципиальные ошибки при ответе на вопросы.

Тестовые задания

Выберите один правильный вариант ответа и нажмите кнопку «Далее»

CASE-технологии преимущественно используются для:
+ Автоматизации этапов разработки программного обеспечения
Увеличения сложности проектирования
Сокращения времени на тестирование
Упрощения взаимодействия со сторонними разработчиками

Методология функционального моделирования IDEF0 позволяет:
+ Моделировать функции и их взаимодействия
Игнорировать функциональные аспекты системы
Сосредоточиться исключительно на технических деталях
Создавать графические пользовательские интерфейсы

DFD (Диаграммы потоков данных) помогают визуализировать:
+ Потоки данных между процессами и хранилищами
Статические структуры базы данных
Исключительно алгоритмы обработки данных
Все участки кода программы

IDEF1X ориентируется на моделирование:
+ Структуры данных и их взаимосвязей
Процессов и их последовательности
Временных аспектов выполнения задач
Объектов и их свойств

Методология IDEF0 представляет систему как:
+ Набор взаимосвязанных функций
Серии линейных последовательностей
Статическую базу данных
Операции, выполняемые в сроки

Основной компонент DFD заключается в:
+ Обозначении процессов, данных и потоков между ними
Описании физической структуры данных
Подробном документировании кода
Отображении схемы взаимодействия пользователей

Модель IDEF1X позволяет документировать:
+ Связи между различными атрибутами и сущностями
Только физические элементы системы
Все возможные потоки данных
Процессы контроля качества

Процесс моделирования в IDEF0 включает:
+ Идентификацию функций и их декомпозицию
Создание единой модели с отсутствием иерархий
Полное игнорирование пользователей
Описание процессов без установления взаимосвязей

Ключевая задача DFD заключается в:
+ Определении взаимосвязей между источниками данных и процессами
Оптимизации времени выполнения алгоритмов
Документировании внутренней структуры базы данных
Определении физического расположения данных

При использовании IDEF1X основное внимание уделяется:
+ Определению логических отношений между данными
Статическим аспектам кода
Упрощению графических интерфейсов
Процессам, не связанным с данными

CASE-средства включают инструменты для:
+ Проектирования, анализа и документирования
Упрощения работы с языками программирования
Мониторинга и отладки выполненного кода
Изучения потребностей пользователей

Одной из особенностей методологии IDEF0 является:
+ Применение функционального подхода к моделированию
Сосредоточение исключительно на базе данных
Непривязка к конкретным технологиям
Игнорирование потока данных

DFD моделей наглядно демонстрируют:
+ Как информация перемещается и трансформируется в системе
Как данные хранятся в базе данных
Как осуществляется взаимодействие с пользователями
Как реализуются функции кодирования

Основным аспектом IDEF1X является:
+ Формализация и документация моделей данных
Описание конкретных алгоритмов
Визуализация интерфейса пользователя
Определение физической модели базы данных

Методология структурного анализа DFD существенно помогает:
+ Определить, как данные перемещаются и где они хранятся
Упрощать сложные алгоритмы
Игнорировать входящие и исходящие данные
Создавать временные отчеты

Моделирование в рамках IDEF0 делится на:
+ Уровни, представляющие функциональную иерархию
Только структурные и информационные компоненты
Линейные и цикличные процессы
Общие и специфические алгоритмы

Ключевой документ в DFD — это:
+ Диаграмма потоков данных, отображающая процессы и связи
Подробная инструкция по программированию
Техническая документация для разработки баз данных
Стандартный функциональный отчет

Основной выходной продукт IDEF1X — это:
+ Логическая модель базы данных
Физическая структура всех функций программы
Набор алгоритмов для обработки данных
Интерфейсные прототипы

Методология IDEF0 подходит для:
+ Разработки процессов и функциональных требований
Полного проектирования пользовательских интерфейсов
Игнорирования производственных процессов
Обработки исключительно данных

Главной целью DFD является:
+ Разработка ясных и наглядных диаграмм для анализа процессов
Упрощение алгоритмов реализации
Описание всех сущностей в системе
Статическое моделирование структуры программы

Методика проведения контроля

Параметры методики	Значение параметра
Предел длительности всего контроля	10 минут
Последовательность выбора вопросов	Случайная
Предлагаемое количество вопросов	10

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который правильно выполнил 9-10 тестовых заданий.

Оценка «хорошо» выставляется обучающемуся, который: правильно выполнил 7-8 тестовых заданий.

Оценка «удовлетворительно» выставляется обучающемуся, который правильно выполнил 5-6 тестовых заданий.

Оценка «неудовлетворительно» выставляется обучающемуся, который правильно выполнил менее 4 тестовых заданий.

Тема 6 Методология объектно-ориентированного анализа и проектирования сложных систем

Контролируемые компетенции (знания, умения) ОК 02 ПК 2.1 ПК 2.2 ПК 2.3

Вопросы для устного опроса

1. Объясните концепцию “объектно-ориентированного анализа” (Object-Oriented Analysis, OOA) и его основные этапы. Как используются “диаграммы UML” (Unified Modeling Language) для визуализации результатов OOA? Приведите пример, как можно использовать UML-диаграммы (например, диаграммы классов, диаграммы использования) для моделирования системы с помощью OOA.
2. Сравните и проанализируйте “объектно-ориентированное проектирование” (Object-Oriented Design, OOD) и “структурное проектирование” (Structured Design). В чем их основные отличия? Какие преимущества и недостатки имеет каждый подход с точки зрения разработки сложных систем? Какие инструменты и технологии используются для реализации OOD и структурного проектирования?
3. Как можно использовать “паттерны проектирования” (Design Patterns) в объектно-ориентированном проектировании? Какие типы паттернов проектирования вы знаете? Приведите пример, как можно использовать паттерн проектирования (например, “Фасад”, “Стратегия”) для решения конкретной задачи проектирования.
4. Объясните концепцию “инкапсуляции” (Encapsulation), “наследования” (Inheritance) и “полиморфизма” (Polymorphism) в контексте объектно-ориентированного анализа и проектирования. Как эти принципы способствуют созданию гибких, модульных и поддерживаемых систем? Приведите пример использования этих принципов в реализации сложной системы.
5. Как можно использовать “методы моделирования объектно-ориентированных систем” (Object-Oriented Modeling Methods) для визуализации структуры и поведения системы? Какие типы моделей используются в объектно-ориентированном моделировании (например, UML диаграммы)? Как модели могут помочь в коммуникации между разработчиками и заказчиками и в управлении разработкой сложной системы?

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который прочно усвоил программный материал в полном объеме, исчерпывающе, грамотно и логически стройно его излагает, четко формулирует основные понятия, приводит соответствующие примеры, уверенно владеет материалом.

Оценка «хорошо» выставляется обучающемуся, который твердо усвоил программный материал, грамотно и по существу излагает его без существенных ошибок, правильно применяет теоретические положения при решении конкретных задач, с небольшими погрешностями приводит формулировки определений, по ходу изложения допускает небольшие пробелы, не искажающие содержания ответа.

Оценка «удовлетворительно» выставляется обучающемуся, который не совсем твердо владеет программным материалом, знает основные теоретические положения изучаемой темы, при ответах допускает малосущественные погрешности, искажения логической последовательности при изложении материала, неточную аргументацию теоретических положений, испытывает затруднения при ответе на дополнительные вопросы.

Оценка «неудовлетворительно» выставляется обучающемуся, имеющему серьезные пробелы в знании учебного материала, допускающему принципиальные ошибки при ответе на вопросы.

Тестовые задания

Выберите один правильный вариант ответа и нажмите кнопку «Далее»

Объектно-ориентированное проектирование основывается на концепции:
+ Абстракций, инкапсуляции и наследования
Статических функций и процедур
Линеаризации всех процессов
Исходного кода без использования объектов

Основным принципом инкапсуляции является:
+ Скрытие внутренней реализации объекта от внешнего мира
Полное открытие всех методов и свойств объекта
Разделение графического интерфейса и логики
Запрет на использование наследования

Наследование в объектно-ориентированном программировании обеспечивает:
+ Унаследование свойств и методов от родительских классов
Полное дублирование кода в каждом классе
Идентичность всех объектов в системе
Несовместимость между классами

Принцип полиморфизма позволяет:
+ Использовать одно и то же имя метода для разных объектов
Объединить все методы в один
Применять только один тип данных в системе
Игнорировать параметры методов

Одной из математических основ объектно-ориентированного проектирования является:
+ Теория множеств и отношения между элементами
Исключительно алгоритмы обработки данных
Комбинаторика без применения объектов
Линейные зависимости в коде

Класс в UML представляет собой:
+ Шаблон для создания объектов
Конкретный экземпляр объекта
Логическую структуру для хранения данных
Функцию без параметров

Атрибуты класса определяются как:
+ Свойства, которым присваиваются значения
Методики обработки информации
Объекты, представляющие действия
Диаграммы, описывающие поведение системы

Ассоциация в UML обозначает:
+ Связь между двумя или более классами
Статическую характеристику объекта
Самостоятельный объект без взаимосвязей
Процесс, который не связан с классами

Композита в UML обозначает:
+ Часть-целое отношение между классами
Простую ассоциацию между классами
Объект, который не может существовать без других
Логическое объединение классов

Сущность в объектно-ориентированном анализе это:
+ Объект, представляющий реальный или абстрактный предмет
Всегда единственный экземпляр класса
Исключительно функция в программе
Набор методов, без состояния

Статический метод в классе может:
+ Вызываться без необходимости создания экземпляра класса
Изменять атрибуты экземпляра класса
Использовать только один параметр
Существовать без класса

Диаграмма состояний в UML используется для:
+ Моделирования жизненного цикла объекта
Описания структуры классов
Установления связей между классами
Создания временных графиков данных

Агрегация в UML характеризуется:
+ Отношением «часть-целое» с независимостью частей
Плотной зависимостью частей от целого
Полной иерархией объектов
Непосредственной зависимостью обоих объектов

Видимость метода класса описывается с помощью:
+ Модификаторов доступа, таких как public, private, protected
Зависимости от внешнего окружения
Логики алгоритмов
Статического контроля типов данных

Диаграмма последовательностей в UML используется для:
+ Моделирования взаимодействий объектов во времени
Определения структуры базы данных
Создания статических моделей классов
Описания характеристик единственного класса

Классы могут содержать:
+ Методы и атрибуты, определяющие их поведение и состояние
Только методы без состояния
Исключительно данные без обработки
Подобъекты, не относящиеся к основной логике

Принцип единственной ответственности подразумевает, что класс должен:
+ Иметь только одну причину для изменения
Содержать множество различных функций
Быть объединенным с другими классами
Игнорировать свои атрибуты

Объект в контексте ООП определяет:
+ Конкретный экземпляр класса с состоянием и поведением
Логическую единицу обработки данных
Процесс, связанный с функцией
Идентификатор, используемый для классификации

Математическая логика в объектно-ориентированном анализе используется для:
+ Формализации требований и описания взаимодействий
Оптимизации графического интерфейса
Устранения необходимости в документации
Разработки кодов для экспериментов

Модель в UML может состоять из:
+ Различных видов диаграмм, описывающих систему
Исключительно текстовой информации
Данных статистического характера
Простых графиков без описания классов

Методика проведения контроля

Параметры методики	Значение параметра
Предел длительности всего контроля	10 минут
Последовательность выбора вопросов	Случайная
Предлагаемое количество вопросов	10

Критерии оценки:

Оценка «отлично» выставляется обучающемуся, который правильно выполнил 9-10 тестовых заданий.

Оценка «хорошо» выставляется обучающемуся, который: правильно выполнил 7-8 тестовых заданий.

Оценка «удовлетворительно» выставляется обучающемуся, который правильно выполнил 5-6 тестовых заданий.

Оценка «неудовлетворительно» выставляется обучающемуся, который правильно выполнил менее 4 тестовых заданий.

Вопросы для собеседования по темам курса

Контролируемые компетенции (знания, умения) ОК 02 ПК 2.1 ПК 2.2 ПК 2.3

Критерии оценки:

1. Какие основные принципы объектно-ориентированного программирования применяются при разработке программных модулей?
2. Что такое абстракция данных и как она используется в разработке модулей?
3.Какие типы модулей существуют и для чего они применяются в программировании?
4.Какие инструменты и технологии используются при разработке модулей?
5. Каким образом происходит тестирование программных модулей и какие методы тестирования применяются?
6. Какие принципы SOLID применяются при проектировании модулей?
7. Каким образом происходит сопровождение и обновление программных модулей?
8. Какие методы и средства контроля версий используются при разработке модулей?
9. Каким образом происходит интеграция модулей в целостную программную систему?
10. Какие проблемы могут возникнуть при разработке и поддержке большого количества модулей?
11. Каким образом происходит оптимизация и улучшение производительности модулей?
12. Как организовать взаимодействие между разными модулями в программе?
13. Какие методы и практики разработки применяются для повышения безопасности программных модулей?
14. Какие стандарты и протоколы применяются при разработке модулей?
15. Каким образом можно оптимизировать процесс разработки модулей для ускорения проекта?
16. Какие методы обеспечения качества используются при разработке программных модулей?
17. Как оценить эффективность и эффективность работы программных модулей?
18. Как применять принцип DRY (Don't repeat yourself) при разработке модулей?
19. Каким образом можно обеспечить масштабируемость программных модулей?
20. Какую роль играют документация и комментарии при разработке программных модулей?

Оценка «отлично» выставляется обучающемуся, который правильно ответил на 5 случайно выбранных вопросов, показав достаточный уровень знаний. В случае если студент ответил на 4 вопроса правильно, но рассчитывает получить оценку «отлично» ему задаётся дополнительный вопрос ответить.

Оценка «хорошо» выставляется обучающемуся, который: правильно ответил на 4 случайно выбранных вопросов, показав достаточный уровень знаний. В случае если студент ответил на 3 вопроса правильно, но рассчитывает получить оценку «хорошо» ему задаётся дополнительный вопрос.

Оценка «удовлетворительно» выставляется обучающемуся, который: правильно ответил на 3 случайно выбранных вопросов, показав достаточный уровень знаний. В случае если студент ответил на 2 вопроса правильно, но рассчитывает получить оценку «удовлетворительно» ему задаётся дополнительный вопрос.

Оценка «неудовлетворительно» выставляется обучающемуся, который не ответил ни на один вопрос или ответил на 1 или 2 вопроса верно, но не ответил на дополнительный вопрос

Форма промежуточной аттестации по дисциплине *экзамен*.

Окончательные результаты обучения (формирования компетенций) определяются посредством перевода баллов, набранных студентом в процессе освоения дисциплины, в оценки:

– базовый уровень сформированности компетенции считается достигнутым если результат обучения соответствует оценке «удовлетворительно» (50 до 64 рейтинговых баллов);

– повышенный уровень сформированности компетенции считается достигнутым, если результат обучения соответствует оценкам «хорошо» (65-85 рейтинговых баллов) и «отлично» (86-100 рейтинговых баллов).

Дополнительные контрольные испытания

для студентов, набравших менее 50 баллов (в соответствии с Положением «О модульно-рейтинговой системе»), формируются из числа оценочных средств по темам, которые не освоены студентом.