

Кафедра

прикладной информатики и высшей математики

Б1.В.15

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА)

Учебная дисциплина	<i>Программная инженерия</i>
По направлению подготовки	<i>09.03.03</i>
	<i>«Прикладная информатика»</i>
Профиль (программа бакалавриата)	<i>«Дизайн и разработка цифровых программных продуктов»</i>
Форма обучения	<i>Очная</i>

Оценочные материалы (средства) дисциплины рассмотрены (актуализированы) и утверждены на заседании кафедры прикладной информатики и высшей математики

Протокол заседания № 10 от «25» мая 2026 г.

Заведующий кафедрой Стрекалова Наталья Борисовна

Оглавление

1. ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ И ЭТАПЫ ИХ ФОРМИРОВАНИЯ В ПРОЦЕССЕ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ	3
2. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА) ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ И ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ОБЕСПЕЧИВАЮЩИХ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ В ПРОЦЕССЕ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ	7
3. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА) ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ	31
4. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРЫ ОЦЕНОВАНИЯ ЗНАНИЙ, УМЕНИЙ И НАВЫКОВ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ	35
5. ДИАГНОСТИЧЕСКИЕ ЗАДАНИЯ ДЛЯ ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ХАРАКТЕРИЗУЮЩИХ УРОВЕНЬ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ	44
6. ЛИСТ СОГЛАСОВАНИЯ	45

1. ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ И ЭТАПЫ ИХ ФОРМИРОВАНИЯ В ПРОЦЕССЕ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Оценочные материалы (средства) сформированы в соответствии с ФГОС ВО - бакалавриат по направлению подготовки 09.03.03 «Прикладная информатика», утвержденный приказом Министерства образования и науки Российской Федерации от 19.09.2017 № 922 с изменениями и дополнениями от 26.11.2020, 08.02.2021). В соответствии с матрицей компетенций основной профессиональной образовательной программы «Дизайн и разработка цифровых программных продуктов» (уровень бакалавриата) в процессе обучения по дисциплине «Программная инженерия» происходит формирование закрепленных за дисциплиной компетенций обучающихся. Оценка сформированности компетенций на каждом этапе обучения происходит через оценку планируемых результатов обучения по дисциплине (знаний, умений, навыков).

Результаты освоения образовательной программы (компетенции)	Индикаторы достижения компетенций	Планируемые результаты обучения по дисциплине	Этапы формирования компетенции (семестры и темы)	Оценочное средство
ПК-1 Способен выявлять информационные потребности пользователей и составлять техническое задание на разработку цифрового программного продукта	ПК-1.2. Разрабатывает техническое задание, формализует требования заказчика к разработке цифрового программного продукта	Знать: - понятия, особенности и архитектуру программного обеспечения, профессиональные и этические требования профессионального сообщества программистов к деятельности по разработке информационных систем; Уметь: - составлять техническую документацию процесса разработки программного обеспечения; Владеть: - навыками разработки отдельных технических документов;	7 семестр Тема 1. Введение в программную инженерию Тема 6. Архитектура программного обеспечения	- устный опрос по темам 1, 6 - ответ на теоретический вопрос экзамена

ПК-2 Способен проектировать цифровой программный продукт	ПК-2.1. Разрабатывает модель цифрового программного продукта с учетом внедрения передовых ИТ-решений прикладной сферы	Знать: - процесс и средства разработки информационных систем и программных комплексов; - факторы сложности разработки информационных систем, технологии, подходы и принципы к исследованию и созданию информационных систем, эргономические требования к ним Уметь: - определять факторы сложности разработки программных систем, выбирать адекватные технологии для их проектирования Владеть: - навыками критического анализа технической информации	7 семестр Тема 2. Разработка сложных программных систем Тема 3. Оценка качества процессов создания программного обеспечения Тема 4. Процесс разработки программного обеспечения Тема 5. DevOps и инфраструктурная инженерия	- устный опрос по темам 2,3, 5 - практическая работа по теме 4,7 - выступление с докладом по теме 4 - ответ на теоретический вопрос экзамена
ПК-4 Способен управлять процессом разработки цифрового программного продукта	ПК-4.1. Применяет гибкие технологии управления и цифровые средства контроля жизненного цикла разработки цифрового программного продукта	Знать: - суть процесса разработки программного обеспечения и виды деятельности в нем, модели жизненного цикла разработки программного обеспечения; - принципы конфигурационного управления и управления требованиями, суть методологии MSF - принципы и средства управления программным проектом и командой разработчиков; Уметь: - планировать этапы разработки	7 семестр Тема 4. Процесс разработки программного обеспечения Тема 5. DevOps и инфраструктурная инженерия Тема 6. Архитектура программного обеспечения Тема 8. Управление программным проектом	- устный опрос по темам 5, 6 - практическая работа по теме 4, 8 - выступление с докладом по теме 4 - ответ на теоретический вопрос экзамена

		информационной системы с использованием инструментальных средств; Владеть: - навыками управления программным проектом с помощью программно-технических средств, - навыками работы с сервисами контроля версий программного продукта		
	ПК-4.2. Обеспечивает разработку проектной и технической документации цифрового программного продукта	Знать: - нормативные документы и стандарты в процессе создания информационных систем Уметь: - применять нормативные документы и стандарты в процессе создания информационных систем Владеть: - навыками разработки информационных систем с использованием нормативных документов и стандартов.	7 семестр Тема 3. Оценка качества процессов создания программного обеспечения Тема 4. Процесс разработки программного обеспечения	- устный опрос по теме 3 - практическая работа по теме 4 - выступление с докладом по теме 4 - ответ на теоретический вопрос экзамена
ПК-5 Способен обеспечивать качество работы цифрового программного продукта	ПК-5.3. Осуществляет работы по интеграционному тестированию цифрового программного продукта с внешним окружением	Знать: - международные и отечественные стандарты оценки качества разработки цифровых программных продуктов; - методы и способы тестирования программного обеспечения - основы эффективной организации разработки цифрового	7 семестр Тема 5. DevOps и инфраструктурная инженерия Тема 7. Тестирование ПО	- устный опрос по теме 5 - практическая работа по теме 7 - ответ на теоретический вопрос экзамена

		программного продукта и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта Уметь: - осуществлять эффективную организацию разработки цифрового программного продукта и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта - проводить тестирование цифровых программных продуктов Владеть: - навыками проверки работоспособности информационных систем и эффективность интеграции всех технологических процессов - навыками построения плана и протокола тестирования		
--	--	---	--	--

2. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА) ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ И ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ОБЕСПЕЧИВАЮЩИХ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ В ПРОЦЕССЕ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Устные опросы

*Вопросы для проведения устного опроса
по теме «Введение в программную инженерию»*

1. Что такое программная инженерия?
2. Назовите дату зарождения программной инженерии как отдельной науки.
3. В чем отличие программной инженерии от информатики?
4. В чем отличие программной инженерии от системотехники?
5. Приведите примеры дисциплин информатики и программной инженерии (дисциплины не путать с учебными предметами).
6. Что такое ПО?
7. С какими иными видами человеческой деятельности соотносится создание ПО в данном разделе?
8. Что такое процесс создания ПО?
9. Причины отсутствия универсального процесса разработки ПО.
10. Почему возможно и целесообразно стандартизировать процесс на уровне компании?
11. Какие методологии разработки ПО поддерживают понятие конкретного процесса и какими средствами?

*Вопросы для проведения устного опроса
по теме «Разработка сложных программных систем»*

1. Что такое процесс создания ПО?
2. Расскажите о причинах отсутствия универсального процесса разработки ПО.
3. Почему возможно и целесообразно стандартизировать процесс на уровне компании?
4. Что такое стандартный и конкретный процессы и как они соотносятся?
5. Чем отличаются между собой текущий и конкретный процессы? Какие методологии разработки ПО поддерживают понятие конкретного процесса и какими средствами?
6. Что такое процесс создания ПО?
7. Расскажите о причинах отсутствия универсального процесса разработки ПО.
8. Почему возможно и целесообразно стандартизировать процесс на уровне компании?
9. Что такое стандартный и конкретный процессы и как они соотносятся?

*Вопросы для проведения устного опроса
по теме «Оценка качества процессов создания программного обеспечения»*

1. Какими характеристиками должен обладать качественный программный продукт?
2. Какие нефункциональные требования определяют качество программного продукта?
3. Какая роль тестирования в обеспечении качества программного продукта?
4. Какие типы тестов используют для проверки качества программного продукта?
5. Для чего применяется регрессионное тестирование?

*Вопросы для проведения устного опроса
по теме «Процесс разработки программного обеспечения»*

1. Что включает понятие «технология разработки программного обеспечения»?
2. Что определяет жизненный цикл программного обеспечения?
3. Поясните содержание каскадной модели разработки программного обеспечения.
4. Поясните содержание итерационной спиральной модели разработки программного обеспечения.
5. Поясните содержание итеративной модели разработки программного обеспечения.
6. Что должен обеспечивать эффективный подход к управлению процессом разработки ПО?
7. Что понимается под зрелостью процессов для компании, разрабатывающей ПО?
8. Приведите основные положения гибкого подхода к созданию ПО.
9. Приведите основное назначение методологии управления жизненным циклом приложений.
10. Какие инструментальные средства предлагает компания Microsoft для управления жизненным циклом приложений?
11. Краткая характеристика программных средств и стадии ее разработки
12. Определение трудоемкости разработки программных средств
13. Определение состава группы исполнителей
14. Расчет и построение сетевого плана-графика разработки и внедрения программных средств
15. Организация документирования программных средств
16. Требования к документации программных средств
17. Планирование документирования программных средств
18. Состав и содержание документов программных средств
19. Стандарты документирования программного обеспечения.

*Вопросы для проведения устного опроса
по теме «DevOps и инфраструктурная инженерия»*

1. Какие шаблоны тестовых проектов имеются в Visual Studio?
2. Для чего применяется Microsoft Test Manager? Какие он имеет функциональные возможности?
3. Для чего используется Lab Management при тестировании?
4. Для чего применяют рефакторинг кода?
5. Приведите признаки некачественного кода.
6. Взаимодействие между подсистемами и архитектурные функции.
7. Чем отличаются между собой текущий и конкретный процессы? Какие методологии разработки ПО поддерживают понятие конкретного процесса и какими средствами?

*Вопросы для проведения устного опроса
по теме «Архитектура программного обеспечения»*

1. Понятие архитектуры и задачи ее описания.
2. Основные классы архитектур программных средств.
3. Контроль архитектуры программных средств.
4. Что такое архитектура программного обеспечения?
5. Важность архитектуры программного обеспечения
6. Каким способом программное обеспечение развертывается?
7. Кем обслуживается программное обеспечение в процессе эксплуатации?

8. Требования к качеству программного обеспечения (безопасность, производительность, возможность параллельной обработки, интернационализация и конфигурация)?

*Вопросы для проведения устного опроса
по теме «Тестирование ПО»*

1. В чем отличие тестирования от обеспечения качества? Цель тестирования.
2. Какие виды деятельности выполняет тестировщик в процессе тестирования?
3. Какие существуют виды тестирования по глубине покрытия?
4. Какие уровни качества выставляются по результатам каждого вида тестирования по глубине покрытия?
5. Какие существуют тестовые активности?
6. На какой глубине покрытия выполняется каждая тестовая активность?
7. Какие существуют виды тестирования по знанию кода?
8. Какие существуют виды тестирования по степени автоматизации?
9. Какие существуют виды тестирования по изолированности компонентов?
10. Какие существуют виды тестирования в зависимости от объекта?
11. Какие существуют нефункциональные виды тестирования?
12. Комбинация тестов для первой поставки программного обеспечения на тестирование.
Комбинация тестов для последующих поставок программного обеспечения на тестирование.

*Вопросы для проведения устного опроса
по теме «Управление программным проектом»*

1. Взаимосвязь управления проектами и функционального менеджмента.
2. Перспективы развития управления проектами. Переход к проектному управлению: задачи и этапы решения.
3. Классификация базовых понятий управления проектами. Классификация типов проектов. Цель и стратегия проектов. Результат проекта. Управление параметрами проекта. Проектный цикл.
4. Общая характеристика программных проектов.
5. Процессы управления проектом. Уровни зрелости процессов управления проектами.
6. Модели жизненного цикла ИТ-продукта. Соотношение жизненного цикла ИТ-решения и жизненного цикла проекта.
7. Теории управления программным проектом.
8. Классификация методов, моделей и стандартов разработки программного обеспечения. Методологии быстрой адаптивной разработки Agile(SCRUM).
9. Методологии разработки и внедрения ИТ-решений.

Критерии оценивая устного опроса:

- оценка «отлично» выставляется студенту, если он ответил развернуто на один из заданных вопросов, активно дополнял ответы других студентов или задавал им дополнительные вопросы;
- оценка «хорошо» выставляется студенту, если он ответил развернуто на один из заданных вопросов или ответил кратко на ряд вопросов;
- оценка «удовлетворительно» выставляется студенту, если он кратко ответил на один из задаваемых вопросов или просто дополнял ответы других студентов, задавал им дополнительные вопросы;

- оценка «*неудовлетворительно*» выставляется студенту, если он не смог ответить правильно на заданный вопрос, либо не отвечал на вопросы и не участвовал в их обсуждении.

Подготовка и выступление с докладами

Темы для подготовки докладов

по теме «Процесс разработки программного обеспечения»

1. Основные понятия и принципы разработки программного обеспечения.
2. Архитектура программного обеспечения.
3. Базовые принципы и ограничения структурного подхода к анализу и проектированию ПО
4. Методологии разработки программного обеспечения.
5. Жизненный цикл программного обеспечения.
6. Разработка программного обеспечения для повторного использования.
7. Компонентно-базированная разработка.
8. Качество программного обеспечения.
9. Сертификация и оценка процессов создания программного обеспечения.

Критерии оценивая докладов и выступлений:

- оценка «*отлично*» выставляется студенту, если доклад полностью раскрывает тему, структура доклада логично выстроена, обучающийся хорошо ориентируется в материале и текст доклада, в основном, рассказывает (а не читает), отвечает на дополнительные вопросы; сопровождающая доклад презентация выполнена в соответствии с требованиями и дополняет доклад, докладчик уложился в регламент, текст доклада оформлен в соответствии с требованиями;
- оценка «*хорошо*» выставляется студенту, если доклад в целом раскрывает тему, но остаются не освещенные стороны вопросы, не на все дополнительные вопросы обучающийся может дать ответ; структура доклада логично выстроена, текст доклада обучающийся, в основном, рассказывает (а не читает); презентация выполнена в соответствии с требованиями, но есть ряд замечаний по ее оформлению, доклад и презентация дополняют друг друга, докладчик уложился в регламент;
- оценка «*удовлетворительно*» выставляется студенту, если его доклад не раскрывает тему в достаточной мере, не отвечает на дополнительные вопросы; к оформлению доклада много замечаний; презентация выполнена не по требованиям, доклад дублирует презентацию, а не дополняет ее, студент не укладывается в регламент или наоборот;
- оценка «*неудовлетворительно*» выставляется студенту, если доклад отсутствует или доклад совсем не раскрывает тему, не оформлен по требованиям; отсутствует презентация или презентация не соответствует более половины требованиям.

Практические работы

Практическое задание по теме «Тестирование программного обеспечения»

Практическое занятие № 1(требуется составление отчета)

Тема: Качество и тестирование программного продукта

Теоретическая часть

Принципы тестирования программного обеспечения

Тестирование (software testing) – деятельность, выполняемая для оценки и улучшения качества программного обеспечения. Эта деятельность, в общем случае, базируется на обнаружении дефектов и проблем в программных системах.

Тестирование программных систем состоит из *динамической* верификации поведения программ на *конечном (ограниченном)* наборе тестов (set of test cases), *выбранных* соответствующим образом из обычно выполняемых действий прикладной области и обеспечивающих проверку соответствия *ожидаемому* поведению системы.

Общий взгляд на тестирование программного обеспечения последние годы активно эволюционировал, становясь все более конструктивным, прагматичным и приближенным к реалиям современных проектов разработки программных систем. Тестирование более не рассматривается как деятельность, начинающаяся только после завершения фазы конструирования. Сегодня тестирование рассматривается как деятельность, которую необходимо проводить на протяжении всего процесса разработки и сопровождения и является важной частью конструирования программных продуктов. Действительно, планирование тестирования должно начинаться на ранних стадиях работы

с требованиями, необходимо систематически и постоянно развивать и уточнять планы тестов и соответствующие процедуры тестирования. Даже сами по себе сценарии тестирования оказываются очень полезными для тех, кто занимается проектированием, позволяя выделять те аспекты требований, которые могут неоднозначно интерпретироваться или даже быть противоречивыми.

В области знаний “Качество программного обеспечения” (Software Quality) техники управления качеством четко разделены на *статические (без выполнения кода)* и *динамические (с выполнением кода)*. Обе эти категории важны. Данная область знаний - “Тестирование” – касается динамических техник. Как уже отмечалось ранее, тестирование тесно связано с областью знаний “Конструирование” (Software Construction). Более того, модульное (unit-) и интеграционное тестирование все чаще рассматривают как неотъемлемый элемент деятельности по конструированию.

Цели тестирования

Тестирование — это проверка соответствия ПО требованиям, осуществляемая с помощью наблюдения за его работой в специальных, искусственно построенных ситуациях. Такого рода ситуации называют тестовыми или просто **тестами**.

Тестирование — конечная процедура. Набор ситуаций, в которых будет проверяться тестируемое ПО, всегда конечен. Более того, он должен быть настолько мал, чтобы тестирование можно было провести в рамках проекта разработки ПО, не слишком увеличивая его бюджет и сроки. Это означает, что при тестировании всегда проверяется очень небольшая доля всех возможных ситуаций. По этому поводу Дейкстра заметил, что тестирование позволяет точно определить, что в программе есть ошибка, но не позволяет утверждать, что там нет ошибок.

Тем не менее, тестирование может использоваться для достаточно уверенного вынесения оценок о качестве ПО. Для этого необходимо иметь **критерии полноты** тестирования, описывающие важность различных ситуаций для оценки качества, а также эквивалентности и зависимости между ними. Этот критерий может утверждать, что все равно в какой из ситуаций, А или В, проверять правильность работы ПО, или, если программа правильно работает в ситуации А, то, скорее всего, в ситуации В все тоже будет правильно.

Часто критерий полноты тестирования задается при помощи определения эквивалентности ситуаций, дающей конечный набор классов ситуаций. В этом случае считают, что все равно, какую из ситуаций одного класса использовать в качестве теста. Такой критерий называют **критерием тестового покрытия**, а процент классов эквивалентности ситуаций, случившихся во время тестирования, — достигнутым **тестовым покрытием**.

Таким образом, основные задачи тестирования: построить такой набор ситуаций, который был бы достаточно представителен и позволял бы завершить тестирование с достаточной степенью уверенности в правильности ПО вообще, и убедиться, что в конкретной ситуации ПО работает правильно, в соответствии с требованиями (рис. 1.1).



Рис. 1.1. Схема процесса тестирования

Тестирование — наиболее широко применяемый метод контроля качества. Для оценки многих атрибутов качества не существует других эффективных способов, кроме тестирования.

Тестировать можно соблюдение любых требований, соответствие которым выявляется во время работы ПО. Из характеристик качества по ISO 9126 этим свойством не обладают только атрибуты удобства сопровождения. Поэтому выделяют виды тестирования, связанные с проверкой определенных характеристик и атрибутов качества — тестирование функциональности, надежности, удобства использования, переносимости и производительности, а также тестирование защищенности, функциональной пригодности и пр. Кроме того, особо выделяют **нагрузочное** или **стрессовое** тестирование, проверяющее работоспособность ПО и показатели его производительности в условиях повышенных нагрузок — при большом количестве пользователей, интенсивном обмене данными с другими системами, большом объеме передаваемых или используемых данных и пр.

Виды тестирования

На основе исходных данных, используемых для построения тестов, тестирование делят на следующие виды:

- Тестирование **черного ящика**, нацеленное на проверку требований. Тесты для него и критерии полноты тестирования строятся на основе требований и ограничений, четко зафиксированных в спецификациях, стандартах, внутренних нормативных документах. Часто такое тестирование называется тестированием **на соответствие (conformance testing)**. Частным случаем его является **функциональное** тестирование — тесты для него, а также используемые критерии полноты проведенного тестирования определяют на основе требований к функциональности.
- Еще одним примером тестирования на соответствие является **аттестационное** или **квалификационное** тестирование, по результатам которого программная система получает (или не получает) официальный документ, подтверждающий ее соответствие определенным требованиям и стандартам.
- Тестирование **белого ящика**, оно же **структурное** тестирование — тесты создаются на основе знаний о структуре самой системы и о том, как она работает. Критерии полноты основаны на проценте элементов кода, которые отработали в ходе выполнения тестов. Для оценки степени соответствия требованиям могут привлекаться дополнительные знания о связи требований с определенными ограничениями на значения внутренних данных системы (например, на значения параметров вызовов, результатов и локальных переменных).
- Тестирование, нацеленное на определенные ошибки. Тесты для такого тестирования строятся так, чтобы гарантированно выявлять определенные виды ошибок. Полнота тестирования определяется на основе количества проверенных ситуаций по отношению к общему числу ситуаций, которые мы пытались проверить. К этому виду относится, например, тестирование **на отказ (smoke testing)**, в ходе которого просто

пытаются вывести систему из строя, давая ей на вход как обычные данные, так и некорректные, с намеренно внесенными ошибками.

- Другим примером служит метод оценки полноты тестирования при помощи набора **мутантов** — программ, совпадающих с тестируемой всюду, кроме нескольких мест, где специально внесены некоторые ошибки. Чем больше мутантов не проходит успешно через данный набор тестов, тем полнее и качественнее проводимое с его помощью тестирование.

Еще одна классификация видов тестирования основана на том уровне, на который оно нацелено. Эти же разновидности тестирования можно связать с фазой жизненного цикла, на которой они выполняются.

- **Модульное** тестирование (**unit testing**) предназначено для проверки правильности отдельных модулей, вне зависимости от их окружения. При этом проверяется, что если модуль получает на вход данные, удовлетворяющие определенным критериям
- корректности, то и результаты его корректны. Модульное тестирование является важной составной частью **отладочного** тестирования, выполняемого разработчиками для отладки написанного ими кода.
- **Интеграционное** тестирование (**integration testing**) предназначено для проверки правильности взаимодействия модулей некоторого набора друг с другом. При этом проверяется, что в ходе совместной работы модули обмениваются данными и вызовами операций, не нарушая взаимных ограничений на такое взаимодействие, например, предусловий вызываемых операций.
- **Системное** тестирование (**system testing**) предназначено для проверки правильности работы системы в целом, ее способности правильно решать поставленные пользователями задачи в различных ситуациях. Системное тестирование выполняется через внешние интерфейсы ПО и тесно связано с тестированием **пользовательского интерфейса** (или через пользовательский интерфейс), проводимым при помощи имитации действий пользователей над элементами этого интерфейса.

Если интеграционное и модульное тестирование чаще всего проводят, воздействуя на компоненты системы при помощи операций предоставляемого ими программного интерфейса (Application Programming Interface, API), то на системном уровне без использования пользовательского интерфейса не обойтись, хотя тестирование через API в этом случае также вполне возможно.

Проверка на моделях

Проверка свойств на моделях (model checking) [4] — проверка соответствия ПО требованиям при помощи формализации проверяемых свойств, построения формальных моделей проверяемого ПО (чаще всего в виде автоматов различных видов) и автоматической проверки выполнения этих свойств на построенных моделях. Проверка свойств на моделях позволяет проверять достаточно сложные свойства автоматически, при минимальном участии человека. Однако она оставляет открытым вопрос о том, насколько выявленные свойства модели можно переносить на само ПО (рис. 1.2).



Рис. 1.2. Схема процесса проверки свойств ПО на моделях

Обычно при помощи проверки свойств на моделях анализируют два вида свойств алгоритмов, использованных при построении ПО. **Свойства безопасности (safety properties)** утверждают, что нечто нежелательное никогда не случится в ходе работы ПО. **Свойства живучести (liveness properties)** утверждают, наоборот, что нечто желательное при любом развитии событий произойдет в ходе его работы.

В классическом подходе к проверке на моделях проверяемые свойства формализуются в виде формул так называемых **временных логик**. Их общей чертой является наличие операторов «всегда в будущем» и «когда-то в будущем».

Проверяемая программа в классическом подходе моделируется при помощи конечного автомата. Проверка, выполняемая автоматически, состоит в том, что для всех достижимых при работе системы состояний этого автомата проверяется нужное свойство. Если оно оказывается выполненным, выдается сообщение об успешности проверки, если нет — выдается трасса, последовательность выполнения отдельных шагов программы, моделируемых переходами автомата, приводящая из начального состояния в такое, в котором нужное свойство нарушается.

Задания

1. Сформулировать принципы тестирования ПО.
2. Перечислить основные цели тестирования ПО.
3. Проанализировать схему тестирования ПО.
4. Определение тестирования черного ящика.
5. Дать определение модульного тестирования.
6. Проведение интеграционного тестирования.
7. Системное тестирование ПО.
8. Тестирование на моделях.
9. Общая схема тестирования на моделях.

Критерии оценивая:

- оценка «*отлично*» выставляется студенту, если задание выполнено в полном объеме, студент хорошо ориентируется в технологии тестирования программного обеспечения, логично и корректно демонстрирует работу всех функций;
- оценка «*хорошо*» выставляется студенту, если он выполнил задание почти в полном объеме (может отсутствовать пункт), допускает небольшие «проектно-технологические» ошибки в разработке тестирования программного обеспечения;
- оценка «*удовлетворительно*» выставляется студенту, если он выполнил первые 5 пунктов задания;
- оценка «*неудовлетворительно*» выставляется студенту, если он выполнил менее 5 пунктов задания.

Практическая работа № 2 (требуется составление отчета)

Тема: Настройка модульного тестирования для кода Python

Модульные тесты — это сегменты кода, которые проверяют работу других частей кода в приложении, например изолированных функций, классов и т. д. Если приложение успешно проходит все модульные тесты, то вы по меньшей мере уверены, что все низкоуровневые функции работают правильно.

В Python модульное тестирование широко используется для проверки скриптов в процессе разработки. Поддержка Python в Visual Studio включает обнаружение, выполнение и отладку модульных тестов непосредственно в контексте процесса разработки, а значит, вам не потребуется выполнять эти тесты отдельно.

Эта статья содержит краткий обзор модульного тестирования в Visual Studio для Python.

Выбор платформы тестирования для проекта Python

Visual Studio поддерживает две платформы тестирования для Python — [unittest](#) и [pytest](#) (доступна в Visual Studio 2019, начиная с версии 16.3). По умолчанию при создании проекта

Python платформа не выбрана. Чтобы указать платформу, щелкните правой кнопкой мыши имя проекта в обозревателе решений и выберите параметр **Свойства**. Открывается конструктор проектов, позволяющий настраивать тесты с помощью вкладки **Тест**. На этой вкладке можно выбрать платформу тестирования, используемую для вашего проекта.

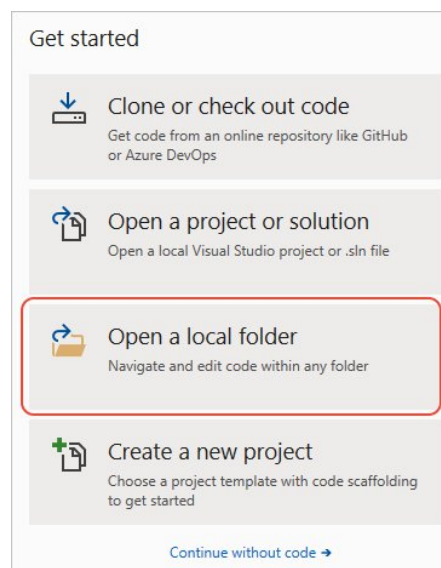
- Для платформы **unittest** корневой каталог проекта используется для обнаружения тестов. На вкладке **Тест** это расположение, а также текстовый шаблон для идентификации тестов можно изменить на значения, заданные пользователем.
- Для платформы **pytest** параметры тестирования, такие как расположение тестов и шаблоны имен файлов, указываются с помощью стандартного INI-файла конфигурации pytest. Дополнительные сведения см. в [справочной документации для pytest](#).

После сохранения выбранных параметров и платформы в обозревателе тестов иницируется обнаружение тестов. Если окно обозревателя тестов еще не открыто, перейдите на панель инструментов и выберите **Тест > Обозреватель тестов**.

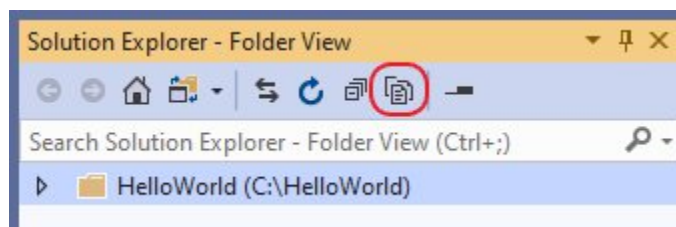
Настройка тестирования для Python без проекта

Visual Studio позволяет запускать и тестировать существующий код Python без проекта, [открыв папку](#) с кодом Python. В этих случаях для настройки тестирования потребуется файл **PythonSettings.json**.

1. Откройте существующий код Python с помощью параметра **Открыть локальную папку**.



2. В окне обозревателя решений щелкните значок **Показать все файлы**, чтобы отобразить все файлы в текущей папке.



3. Перейдите к файлу **PythonSettings.json** в папке **Локальные параметры**. Если этот файл отсутствует в папке **Локальные параметры**, создайте его вручную.
4. Добавьте поле **TestFramework** в файл параметров и задайте для него значение **pytest** или **unittest** в зависимости от требуемой платформы тестирования.

```
JSON

{
  "TestFramework": "unittest",
  "UnitTestRootDirectory": "testing",
  "UnitTestPattern": "test_*.py"
}
```

Примечание

Если для платформы **unittest** в файле PythonSettings.json не указаны поля **UnitTestRootDirectory** и **UnitTestPattern**, они добавляются и получают значения по умолчанию «.» и «test*.py», соответственно.

5. Если папка содержит каталог **src**, отличный от папки с вашими тестами, укажите путь к папке **src** с помощью поля **SearchPaths** в файле **PythonSettings.json**.

```
JSON

{
  "TestFramework": "unittest",
  "UnitTestRootDirectory": "testing",
  "UnitTestPattern": "test_*.py",
  "SearchPaths": [ ".\\src" ]
}
```

Сохраните изменения в файле PythonSettings.json, чтобы начать обнаружение тестов для указанной платформы.

Примечание

Если окно обозревателя тестов уже открыто нажатие клавиш **CTRL + R,A** также активирует обнаружение.

Обнаружение и просмотр тестов

По умолчанию Visual Studio определяет тесты **unittest** и **pytest** как методы, имена которых начинаются с test. Чтобы просмотреть обнаружение тестов, сделайте следующее:

1. Откройте проект Python.
2. После загрузки проекта в Visual Studio щелкните проект правой кнопкой мыши в обозревателе решений и выберите платформу **unittest** или **pytest** на вкладке **Тест** в области свойств.

Примечание

При использовании платформы **pytest** можно указать расположение тестов и шаблоны имен файлов с помощью стандартного INI-файла конфигурации **pytest**. По умолчанию используется папка рабочей области или проекта с шаблоном **test_*.py** и ***_test.py**. Дополнительные сведения см. в справочной документации для **pytest**.

3. Выбрав платформу, снова щелкните проект правой кнопкой мыши и выберите **Добавить > Новый Элемент**, затем **Модульный тест Python** и нажмите кнопку **Добавить**.
4. Это действие создает файл **test_1.py** с кодом, который импортирует стандартный модуль **unittest**, производный от тестового класса **unittest.TestCase**, и вызывает метод **unittest.main()** при запуске скрипта напрямую:

```

Python

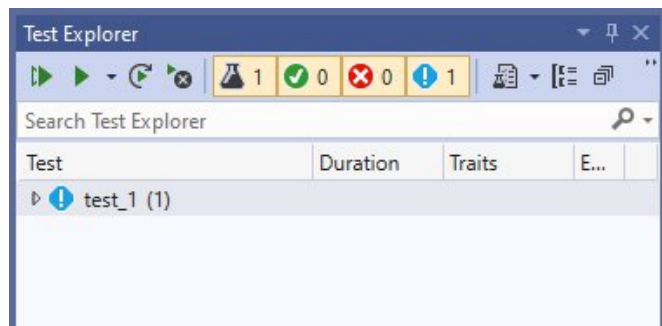
import unittest

class Test_test1(unittest.TestCase):
    def test_A(self):
        self.fail("Not implemented")

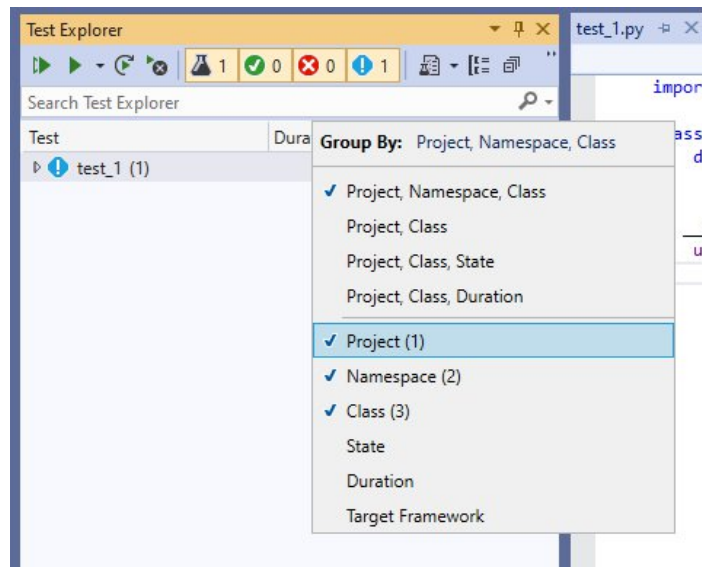
if __name__ == '__main__':
    unittest.main()

```

5. Если нужно, сохраните этот файл, а затем откройте **обозреватель тестов**, последовательно выбрав команды **Тест > Обзоратель тестов**.
6. **Обозреватель тестов** ищет тесты в проекте и отображает результаты, как показано ниже. Дважды щелкните тест, чтобы открыть его исходный файл.



7. Когда тестов в проекте будет больше, их можно упорядочить в **обозревателе тестов**, используя команду **Группировать** на панели инструментов:



8. Также вы можете ввести текст в поле **Поиск**, чтобы отфильтровать тесты по именам.

Дополнительные сведения о модуле unittest и создании тестов можно получить в [документации по Python 2.7](#) или в [документации по Python 3.7](#) (python.org).

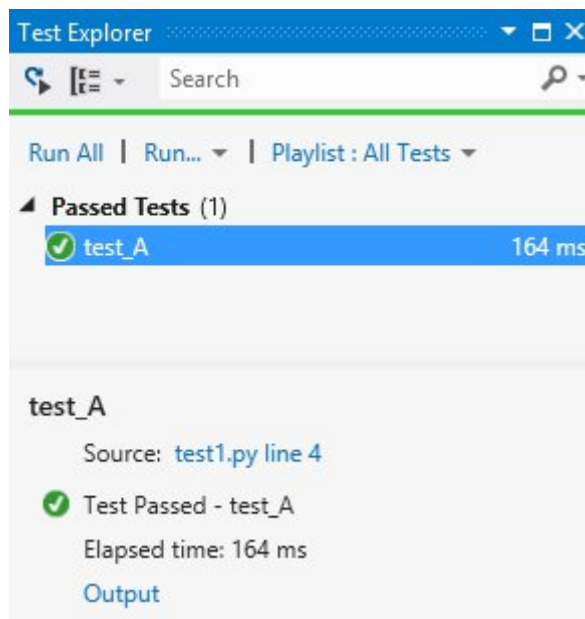
Выполнить тесты

В **обозревателе тестов** можно запустить тесты несколькими способами:

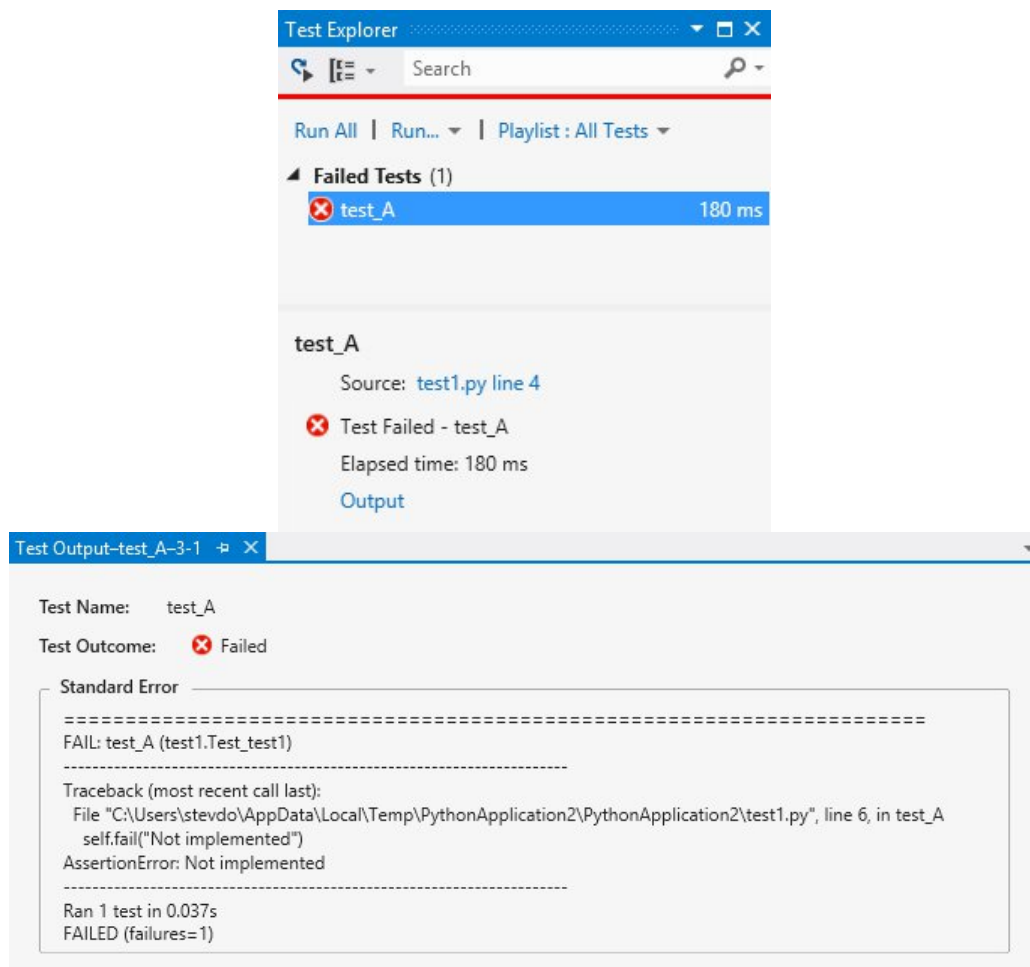
- Команда **Запустить все** явно выполняет все тесты из списка (с учетом фильтров).
- Команда меню **Запуск** позволяет одновременно выполнить все тесты, которые завершились успешно, завершились неудачно или еще не выполнялись.
- Также можно выбрать один или несколько тестов, щелкнуть их правой кнопкой мыши и выбрать команду **Запустить выбранные тесты**.

Тесты выполняются в фоновом режиме, а **обозреватель тестов** обновляет их состояние по мере завершения:

- Успешно выполненные тесты обозначаются зеленым флажком, и для них указывается затраченное время:



- Неудачные тесты обозначаются красным крестом и дополняются ссылкой **Output** (Вывод), которая позволяет изучить выходные данные на консоли и выходные данные unittest, полученные по результатам этого теста:



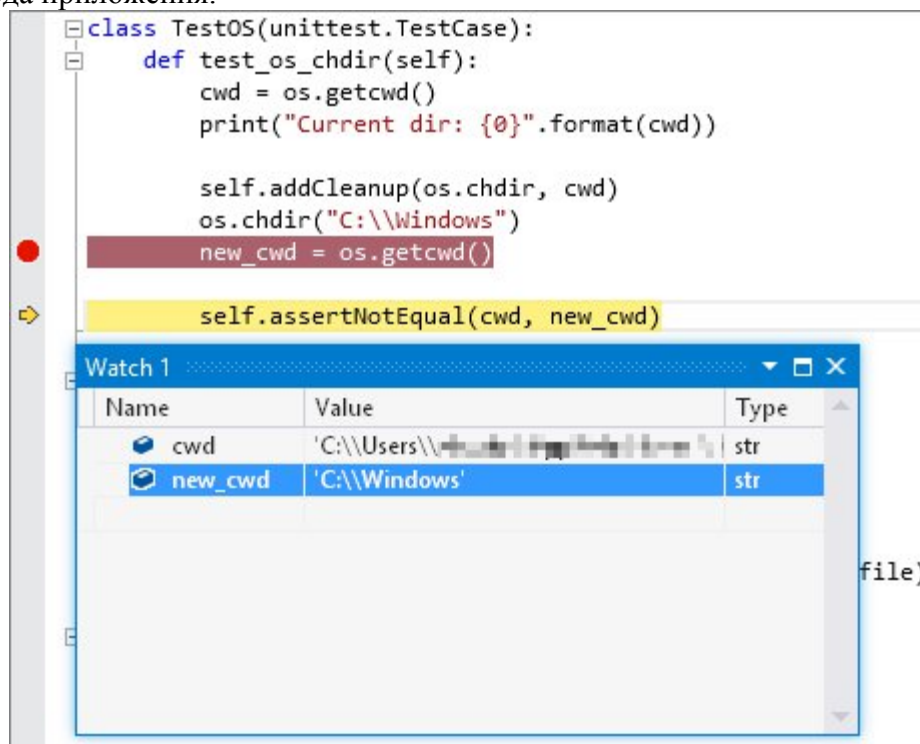
Отладка тестов

Модульные тесты являются частью кода проекта, и в них могут встречаться ошибки, как и в любом другом коде. Периодически тест следует запускать в отладчике, где можно установить точки останова, просмотреть переменные или выполнить код пошагово. Также Visual Studio предоставляет средства диагностики для модульных тестов.

Примечание

По умолчанию для тестовой отладки использует отладчик ptvsd 4. Если вы хотите использовать вместо него ptvsd 3, можно выбрать параметр **Использовать устаревший отладчик** в меню **Сервис > Параметры > Python > Отладка**.

Чтобы начать отладку, установите в коде начальную точку останова, а затем щелкните в **обозревателе тестов** правой кнопкой мыши этот тест (или выделенный набор тестов) и выберите **Отладить выбранные тесты**. Visual Studio запускает отладчик Python, как для обычного кода приложения.



Вы также можете использовать команду **Анализ покрытия кода для выбранных тестов**.

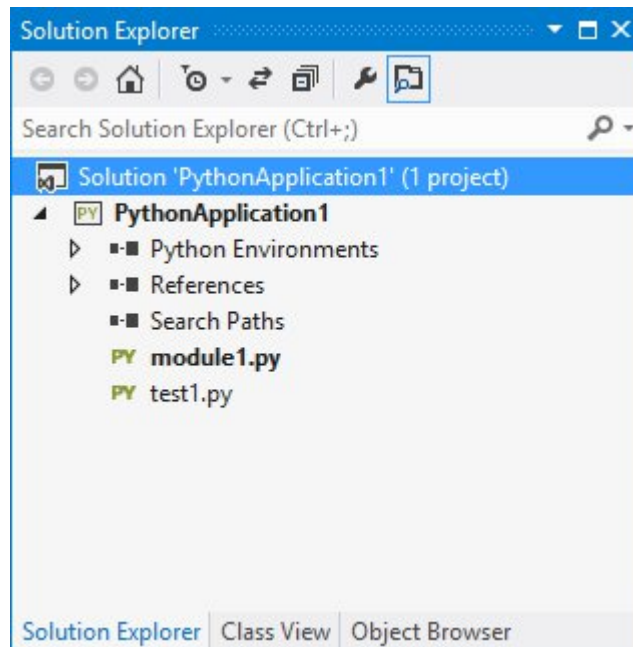
- оценка «отлично» выставляется студенту, если задание выполнено в полном объеме, студент хорошо ориентируется в технологии тестирования программного обеспечения, логично и корректно демонстрирует работу всех функций Visual Studio;
- оценка «хорошо» выставляется студенту, если он выполнил задание почти в полном объеме (может отсутствовать пункт), допускает небольшие «проектно-технологические» ошибки в разработке тестирования программного обеспечения;
- оценка «удовлетворительно» выставляется студенту, если он не выполнил полностью тестирование программного обеспечения, но составил отчет;
- оценка «неудовлетворительно» выставляется студенту, если он не полностью выполнил тестирование программного обеспечения и не составил отчет.

Практическое задание по теме «Управление программным проектом»

Практическая работа № 1 (требуется составление отчета)

Тема: Проекты Python в Visual Studio

Приложения Python обычно определяются только с помощью файлов и папок, но такая структура может усложнить работу, так как приложения увеличиваются в размере и могут содержать автоматически сгенерированные файлы, JavaScript для веб-приложений и т. д. Проект Visual Studio помогает управлять этими сложными моментами. Проект (файл `.pyproj`) определяет все исходные файлы и файлы содержимого, связанные с проектом, содержит сведения о сборке каждого файла, хранит информацию для интеграции с системами управления версиями и помогает упорядочить приложение в виде логических компонентов.



Кроме того, проекты всегда управляются в *решении* Visual Studio, которое может содержать любое число проектов с возможностью ссылаться друг на друга. Например, проект Python может ссылаться на проект C++, который реализует модуль расширения. Благодаря этой связи Visual Studio автоматически создает проект C++ (при необходимости), когда вы запускаете отладку проекта Python.

Visual Studio предоставляет множество шаблонов проектов Python, позволяющих быстро настроить несколько структур приложений, а также шаблон для создания проекта из существующего дерева папок и шаблон для создания пустого проекта. Список шаблонов см. в разделе [Шаблоны проектов](#).

Рекомендации

В Visual Studio 2019 можно открыть папку с кодом Python и выполнить этот код, не создавая проект Visual Studio и файлы решения. Дополнительные сведения см. в разделе [Краткое руководство. Открытие и выполнение кода Python в папке](#). Но файл проекта имеет ряд важных преимуществ, которые мы описали в этой статье.

Рекомендации

Все версии Visual Studio нормально работают с кодом Python даже без проекта. Например, можно открыть сам файл Python и выполнить автозавершение и отладку, а также использовать функцию IntelliSense. Для этого щелкните правой кнопкой мыши в редакторе и выберите пункт **Запуск с отладкой**. Такой код всегда использует глобальное окружение по умолчанию, однако при работе могут возникать неверные завершения или ошибки, если код предназначен для другого окружения. Кроме того, Visual Studio анализирует все файлы и пакеты в папке, из которой открыт один файл, что может значительно расходовать время ЦП.

Добавление файлов, назначение файла запуска и настройка сред

При разработке приложения обычно требуется добавить в проект новые файлы различных типов. Чтобы это сделать, щелкните правой кнопкой мыши проект, выберите **Добавить > Существующий элемент** и перейдите к нужному файлу. Можно также выбрать **Добавить > Новый элемент**. Появится диалоговое окно с различными шаблонами элементов. Как описано в справочнике по [шаблонам элементов](#), доступны такие варианты: пустые файлы Python, класс Python, модульный тест и различные файлы, связанные с веб-приложениями. Вы можете испробовать эти варианты с помощью тестового проекта, чтобы узнать о возможностях вашей версии Visual Studio.

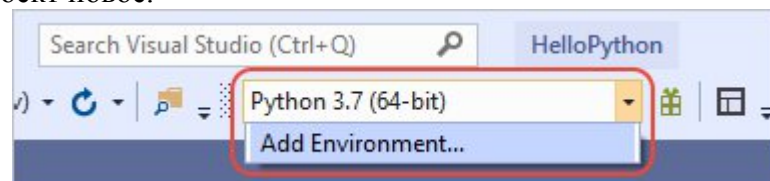
Каждый проект Python имеет один назначенный файл запуска, выделенный полужирным шрифтом в **обозревателе решений**. Это файл, который запускается, когда вы запускаете отладку (F5 или **Отладка > Начать отладку**) или выполняете проект в **интерактивном окне** (SHIFT+ALTF+5 или **Отладка > Выполнить проект в интерактивном окне Python**). Чтобы изменить его, щелкните правой кнопкой мыши новый файл и выберите действие **Назначить автозапускаемым элементом** (или **Задать как файл запуска** в ранних версиях Visual Studio).

Рекомендации

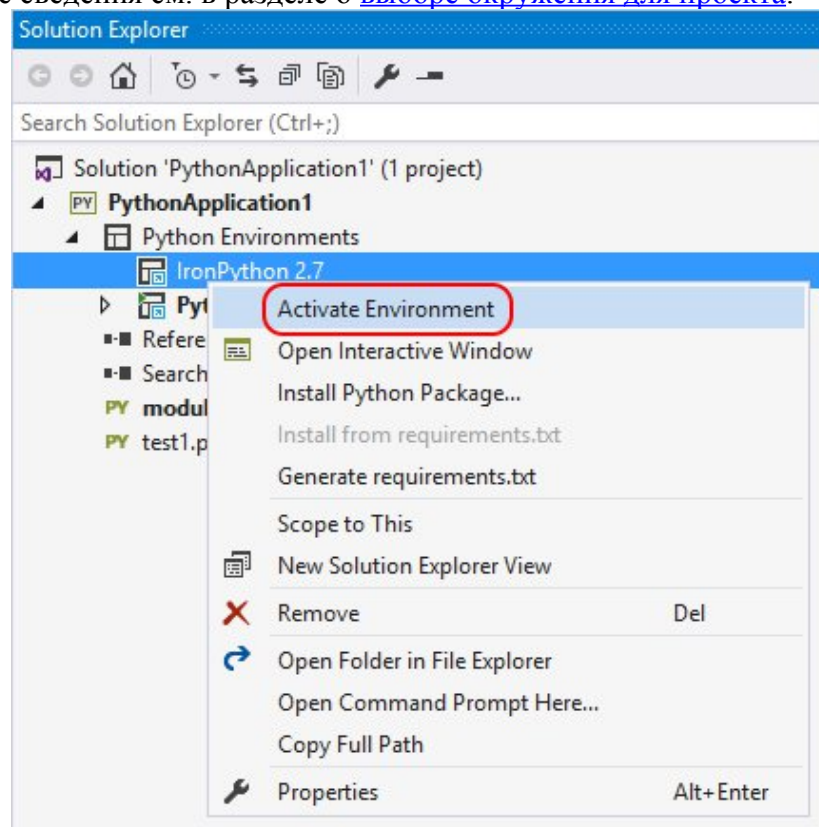
Если удалить выбранный файл запуска из проекта и не выбрать новый, Visual Studio не будет знать, с какого файла Python нужно начинать выполнение проекта. В этом случае в Visual Studio 2017 версии 15.6 и более поздних версий возникает ошибка. В более ранних версиях либо открывается окно вывода с запущенным интерпретатором Python, либо окно вывода появляется, но почти сразу же исчезает. Если у вас возникла подобная ситуация, убедитесь, что назначен файл запуска.

Чтобы окно вывода оставалось открытым, щелкните правой кнопкой мыши проект, выберите **Свойства**, откройте вкладку **Отладка**, а затем добавьте `-i` в поле **Аргументы интерпретатора**. Этот аргумент вынуждает интерпретатор перейти в интерактивный режим после завершения программы, оставив окно открытым, пока вы не нажмете клавиши **CTRL+Z > ВВОД** для выхода.

Новый проект всегда по умолчанию связан с глобальной средой Python. Чтобы связать проект с другим окружением (в том числе виртуальным), в проекте щелкните правой кнопкой мыши узел **Окружения Python** и выберите команду **Добавить окружение...**, а затем выберите нужные окружения. Можно также использовать элемент управления с раскрывающимся списком окружений на панели инструментов, чтобы выбрать окружение или добавить в проект новое.

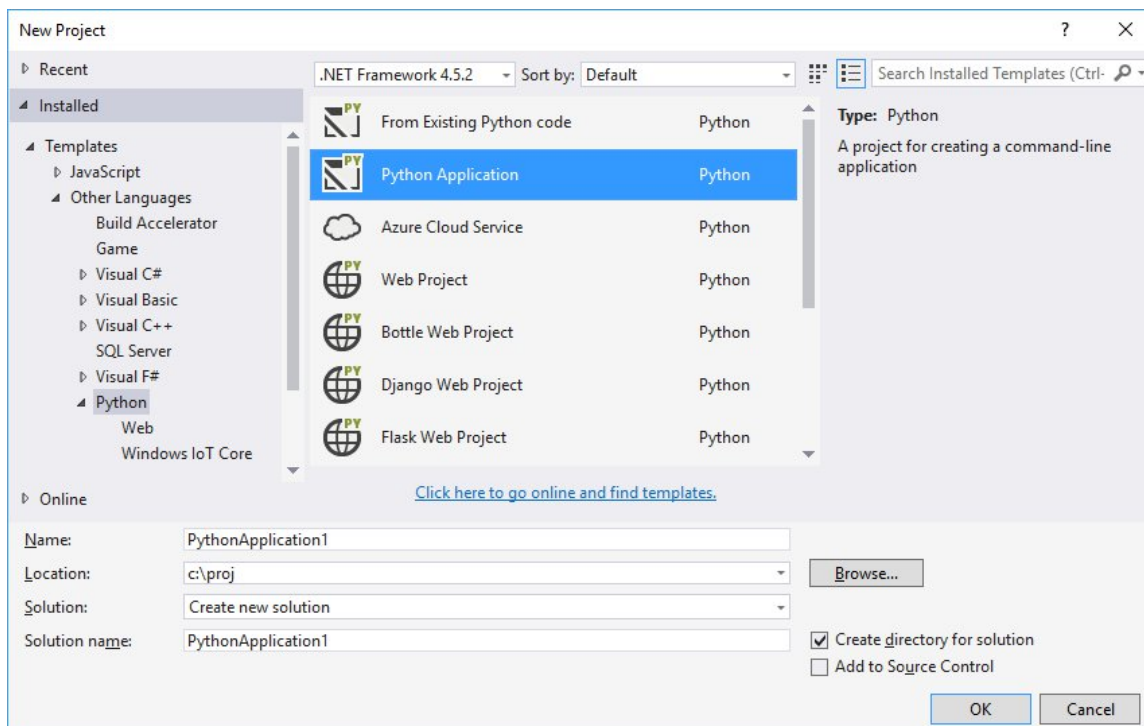


Чтобы изменить активное окружение, в **обозревателе решений** щелкните правой кнопкой мыши нужное окружение и выберите действие **Активировать окружение**. Дополнительные сведения см. в разделе о [выборе окружения для проекта](#).



Шаблоны проектов

Visual Studio предоставляет несколько способов настройки проекта Python — с нуля или из существующего кода. Чтобы использовать шаблон, выберите команду меню **Файл > Создать > Проект** или щелкните правой кнопкой мыши решение в **обозревателе решений** и выберите **Добавить > Новый проект**. В любом случае отобразится диалоговое окно **Новый проект**. Чтобы просмотреть шаблоны конкретно для Python, выполните поиск по запросу «Python» или последовательно выберите **Установленные > Python**:



В следующей таблице перечислены шаблоны, доступные в Visual Studio 2017 и более поздних версий (не все шаблоны доступны в предыдущих версиях):

Шаблон	ОПИСАНИЕ
На основе существующего кода Python	Создает проект Visual Studio из существующего кода Python в структуре папок.
Приложение Python	Структура базового проекта для нового приложения Python с одним пустым исходным файлом. По умолчанию проект выполняется в консоли интерпретатора глобальной среды по умолчанию, которую можно изменить, назначив другую среду .
Облачная служба Azure	Проект для облачной службы Azure, написанный на Python.
Веб-проекты	Проекты для веб-приложений на базе различных платформ, включая Bottle, Django и Flask.
Приложение IronPython	Аналогичен шаблону приложения Python, но по умолчанию использует интерпретатор IronPython, поддерживающий взаимодействие .NET и смешанный режим отладки с использованием языков .NET.
Приложение WPF IronPython	Структура проекта, использующая IronPython с XAML-файлами Windows Presentation Foundation для пользовательского интерфейса приложения. Visual Studio предоставляет конструктор пользовательского интерфейса XAML, возможность написания кода программной части на Python, а также возможность запуска приложения без отображения консоли.
Веб-страница IronPython Silverlight	Это проект IronPython, который выполняется в браузере с подключаемым модулем Silverlight. Код приложения Python добавляется на веб-страницу в виде скрипта. Стандартный тег скрипта получает часть кода JavaScript, который инициализирует IronPython, выполняющийся в Silverlight, откуда код Python может взаимодействовать с моделью DOM.
Приложение Windows Forms IronPython	Структура проекта, использующая IronPython с пользовательским интерфейсом, созданным с помощью кода и Windows Forms. Приложение запускается без вывода консоли.
Фоновое приложение	Поддерживает развертывание проектов Python для работы в качестве фоновых служб на устройствах. Дополнительные

Шаблон	ОПИСАНИЕ
(Интернет вещей)	сведения см. на странице центра разработчиков Интернета вещей Windows .
Модуль расширения Python	Этот шаблон отображается в области Visual C++, если вы уже установили Собственные средства разработки Python с рабочей нагрузкой Python в Visual Studio 2017 или более поздней версии (см. раздел Установка). Он предоставляет базовую структуру для библиотеки DLL расширения C++, как описано в статье Создание расширения C++ для Python .

Примечание

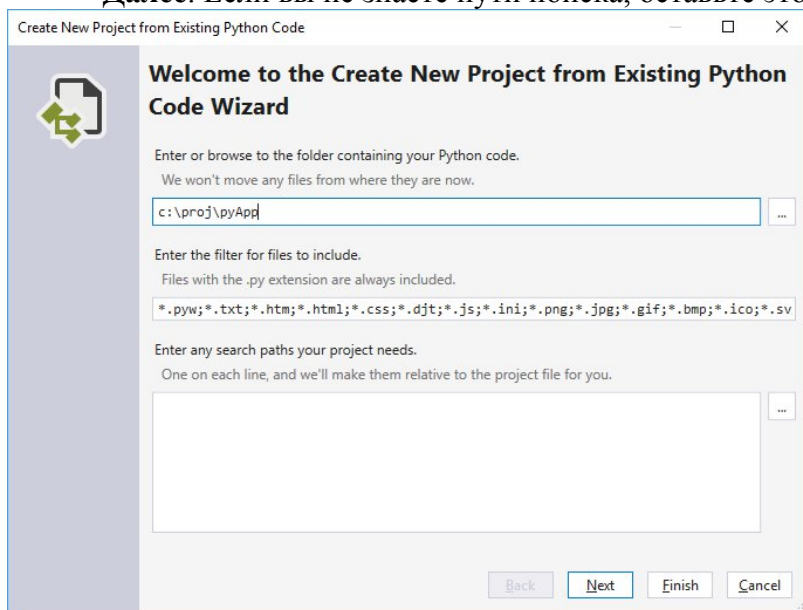
Так как Python является интерпретируемым языком, проекты Python в Visual Studio не создают отдельный исполняемый файл, как это делают проекты, написанные на других компилируемых языках программирования (например, C#). Дополнительные сведения см. в разделе [вопросов и ответов](#).

Создание проекта на основе имеющихся файлов

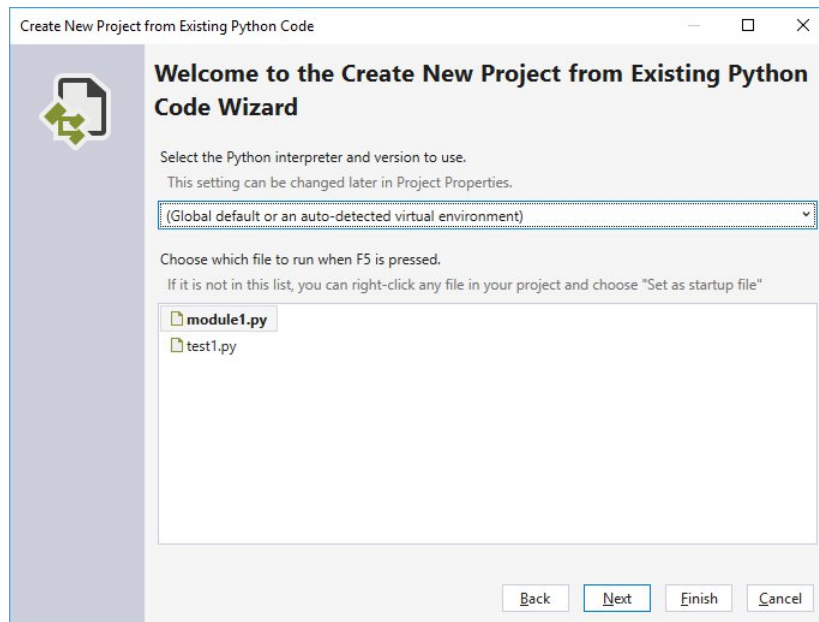
Важно!

Описанный здесь процесс не перемещает и не копирует исходные файлы. Если вы хотите работать с копией, сначала создайте дубликат папки.

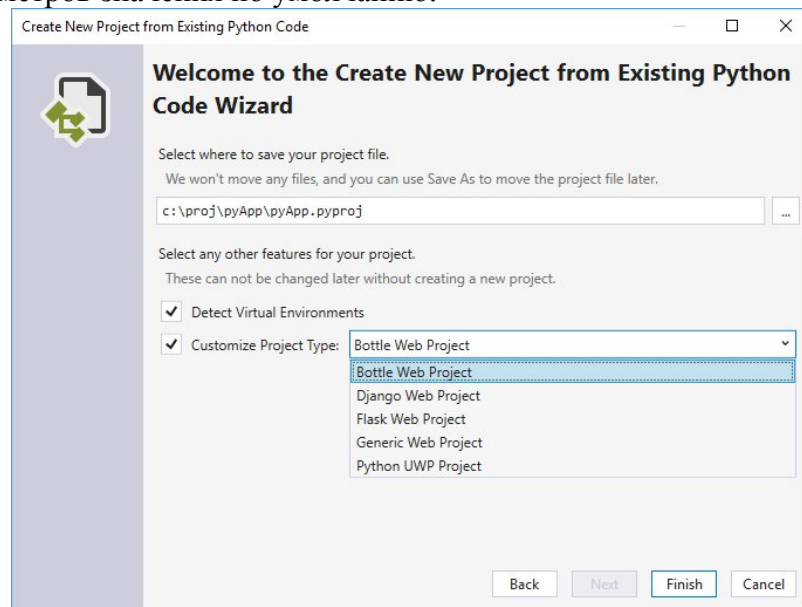
1. Запустите Visual Studio и последовательно выберите **Файл > Создать > Проект**.
2. В диалоговом окне **Создание проекта** выполните поиск по запросу «Python», выберите шаблон **На основе существующего кода Python**, укажите имя и расположение проекта, а затем нажмите кнопку **ОК**.
3. В появившемся мастере задайте путь к существующему коду, фильтр для типов файлов и любые пути поиска, необходимые для проекта, а затем нажмите кнопку **Далее**. Если вы не знаете пути поиска, оставьте это поле пустым.



4. В следующем диалоговом окне выберите файл запуска для проекта и нажмите кнопку **Далее**. (При необходимости выберите среду; в противном случае оставьте значения по умолчанию.) Обратите внимание, что в диалоговом окне отображаются только файлы в корневой папке. Если нужный файл находится во вложенной папке, не указывайте файл запуска и укажите его позже в **обозревателе решений** (инструкции см. ниже).



5. Выберите место, где следует сохранить файл проекта (файл *.pyproj* на диске). При необходимости можно также включить автоматическое обнаружение виртуальных сред и настроить проект для разных веб-платформ. Если вы не уверены, оставьте для этих параметров значения по умолчанию.



6. Нажмите кнопку **Готово**. Visual Studio создаст проект и откроет его в **обозревателе решений**. Если вы хотите переместить *PYPROJ*-файл в другое место, выберите его в **обозревателе решений** и щелкните **Файл > Сохранить как**. Это действие обновляет ссылки на файлы в проекте, но не перемещает файлы с кодом.
7. Чтобы указать другой файл запуска, найдите его в **обозревателе решений**, щелкните его правой кнопкой мыши и выберите пункт **Задать как файл запуска**.

Связанные файлы

Связанные файлы — это файлы, которые добавлены в проект, но при этом находятся за пределами папок проекта приложения. Они отображаются в **обозревателе решений** как



обычные файлы с перекрывающимся значком ярлыка:

Связанные файлы указаны в файле *PYPROJ* с помощью элемента `<Compile Include=«...»>`. Связанные файлы могут быть неявными, если они используют относительный путь за пределами структуры каталогов, или явными, если они используют пути в **обозревателе решений**:

XML

```
<Compile Include="..\test2.py">
  <Link>MyProject\test2.py</Link>
</Compile>
```

Связанные файлы игнорируются при выполнении любого из следующих условий:

- Связанный файл содержит метаданные связи, и путь, указанный в атрибуте Include, находится в пределах каталога проекта.
- Связанный файл дублирует файл, который существует в иерархии проекта.
- Связанный файл содержит метаданные связи, и путь является относительным путем вне иерархии проекта.
- Путь связи является корневым.

Работа со связанными файлами

Чтобы добавить существующий элемент в качестве связи, щелкните правой кнопкой мыши папку проекта, в которую вы хотите добавить файл, а затем выберите **Добавить > Существующий элемент**. В открывшемся диалоговом окне выберите файл и щелкните **Добавить как связь** в раскрывающемся списке кнопки **Добавить**. Если конфликтующие файлы отсутствуют, эта команда создает связь в выбранной папке. Связь не добавится, если файл с таким именем уже существует или связь с этим файлом уже существует в проекте.

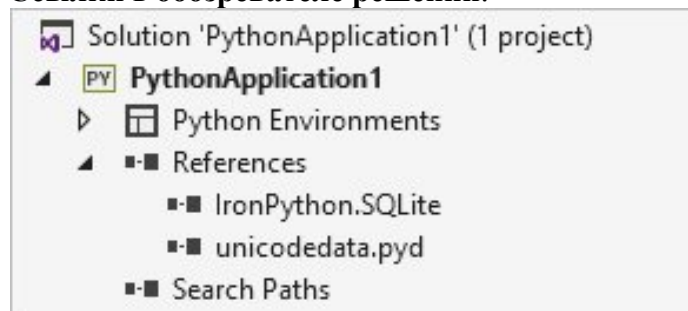
При попытке создать связь с файлом, который уже существует в папке проекта, он добавляется как обычный файл, а не как связанный. Чтобы преобразовать файл в связь, выберите **Файл > Сохранить как** и сохраните файл в расположение вне иерархии проекта. Visual Studio автоматически преобразует его в связь. Аналогичным образом можно преобразовать связь обратно с помощью команды **Файл > Сохранить как** и сохранить файл в иерархии проекта.

При перемещении связанного файла в **обозревателе решений** связь также перемещается, но фактический файл при этом не затрагивается. Аналогичным образом, удаление связи приведет к удалению только связи, не затрагивая сам файл.

Связанные файлы нельзя переименовать.

Ссылки

Проекты Visual Studio поддерживают добавление ссылок на проекты и расширения, которые отображаются в узле **Ссылки в обозревателе решений**.



Ссылки на расширения обычно указывают зависимости между проектами и используются для обеспечения IntelliSense во время разработки или связывания во время компиляции. Проекты Python используют ссылки подобным образом, но из-за динамической природы Python они в основном используются во время разработки для предоставления усовершенствованной функции IntelliSense. Они также могут использоваться для развертывания в Microsoft Azure с целью установки дополнительных зависимостей.

Модули расширений

Ссылка на файл *PYD* позволяет использовать IntelliSense для созданного модуля. Visual Studio загружает файл *PYD* в интерпретатор Python и анализирует его типы и функции. Программа также пытается выполнить синтаксический анализ строк функций в документе, чтобы предоставить справку по сигнатурам.

Если в любой момент модуль расширения обновляется на диске, Visual Studio повторно анализирует модуль в фоновом режиме. Это не влияет на поведение во время выполнения, но некоторые варианты завершения остаются недоступными до завершения анализа.

Необходимо добавить [путь поиска](#) к папке, содержащей модуль.

Проекты .NET

При работе с IronPython можно добавить ссылки на сборки .NET, чтобы активировать использование IntelliSense. Для проектов .NET в решении щелкните правой кнопкой мыши узел **Ссылки** в проекте Python, выберите **Добавить ссылку**, щелкните вкладку **Проекты** и найдите нужный проект. Для библиотек DLL, которые вы скачали отдельно, выберите вкладку **Обзор** и перейдите к требуемой библиотеке DLL.

Так как ссылки в IronPython недоступны до вызова `clr.AddReference('<AssemblyName>')`, в сборку также нужно добавить соответствующий вызов `clr.AddReference`. Обычно он добавляется в начале кода. Например, код, созданный в Visual Studio с помощью шаблона проекта **Приложение Windows Forms IronPython**, включает в себя два вызова в начале файла:

```
Python

import clr
clr.AddReference('System.Drawing')
clr.AddReference('System.Windows.Forms')

from System.Drawing import *
from System.Windows.Forms import *

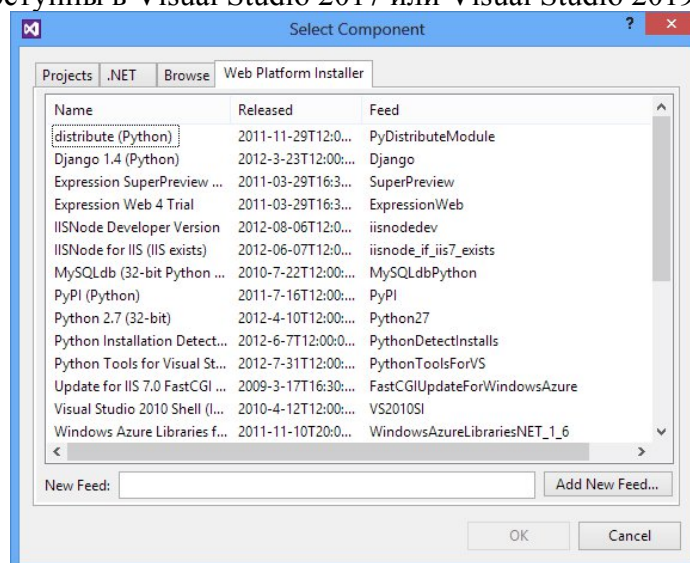
# Other code omitted
```

Проекты WebPI

Можно добавить ссылки на записи продукта WebPI для развертывания в облачных службах Microsoft Azure, где можно установить дополнительные компоненты с помощью веб-канала WebPI. По умолчанию отображаемый веб-канал предназначен только для Python и содержит Django, CPython и другие основные компоненты. Также можно выбрать собственный веб-канал, как показано ниже. При публикации в Microsoft Azure задача установки устанавливает все продукты, на которые имеются ссылки.

Важно!

Проекты WebPI недоступны в Visual Studio 2017 или Visual Studio 2019.



Критерии оценивая:

- оценка «отлично» выставляется студенту, если задание выполнено в полном объеме, студент хорошо ориентируется в технологии управления проектами в Visual Studio;

- оценка «хорошо» выставляется студенту, если он выполнил задание почти в полном объеме (может отсутствовать пункт), допускает небольшие «проектно-технологические» в работе с Visual Studio;
- оценка «удовлетворительно» выставляется студенту, если он не выполнил полностью развертывание проекта программного обеспечения, но составил отчет;
- оценка «неудовлетворительно» выставляется студенту, если он не полностью выполнил развертывание проекта программного обеспечения и не составил отчет.

Практическая работа №2 (требуется составление отчета)

Тема: Методологии управления ИТ-проектами

Цель: знакомство с методологиями управления ИТ-проектами.

Теоретические вопросы

В настоящее время управление проектами имеет свою методологию и основывается на определенных стандартах. В основе таких методов лежат методики сетевого планирования, которые появились США в конце 1950-х гг. При этом методики управления проектами широко распространились не только в странах с рыночной экономикой, но и в странах с т. н. «плановой» экономикой. Они начали использоваться в строительстве, что послужило основой их распространения в других отраслях и появлению методов проектного управления.

Примером здесь может служить объединение усилий фирмы «Дюпон» и фирмы «Ремингтон Рэнд» для составления плана графика комплексных работ по модернизации заводов «Дюпон» с помощью вычислительной машины Univac в 1956 г. Результатом стало создание рационального и простого метода описания проекта на ЭВМ – метода критического пути (CPM – Critical Path Method).

Практически параллельно и независимо был создан метод анализа и оценки программ PERT (Program Evaluation and Review Technique) в военно-морских силах США. Данный метод разрабатывался корпорацией «Локхид» консалтинговой фирмой «Буз, Аллен энд Гамильтон» для ракетной системы «Поларис». Проект включал 3800 основных подрядчиков и состоял из 60 000 операций. Благодаря использованию данного метода проект стал успешным и завершился на два года раньше срока. После наглядного успеха метод PERT стал использоваться в вооруженных силах США для планирования проектов.

Задание: найдите примеры других крупных успешных проектов прошлого века, использующих для своего управления ЭВМ.

Учтите, что с одной стороны, применение информационных технологий для планирования и управления проектами давало в те времена значительное преимущество, а с другой стороны, первые компьютеры были дорогостоящими и оставались доступными только крупным компаниям. Все изменилось с появлением персональных компьютеров, теперь ЭВМ стала рабочим инструментом для руководителей более широкого круга.

Система управления проектами постоянно развивалась и стала самостоятельной областью профессиональной деятельности. В итоге были созданы унифицированные методологии, инструментари, механизмы, стандарты, доступные и для проектов в ИТ-сфере. Например, на сегодняшний день существует единая Международная ассоциация управления проектами – IPMA (International Project Management Association) с центром в г. Цюрих (Швейцария).

Из наиболее распространенных можно отметить процессную модель, используемую в документах методологических основ управления проектами – Project Management Body of Knowledge (PMBOK) Американского института управления проектами (PMI). Данный документ признается международным стандартом де-факто. Кроме того, стандарт ISO 10006:1997 придал ряду наиболее важных положений PMBOK статус стандарта де-юре.

В 2014 г. вышло пятое издание PMBOK, содержащее указания на 589 страницах. Все положения представлены на сайте <http://www.pmi.org/default.aspx> [6].

Данные Международной ассоциации управления проектами (IPMA) говорят о том, что использование современных методологий управления проектами экономит около 20–30% времени и около 15–20% средств на осуществление проектов.

Все методологии (еще их называют моделями, методиками) разработки программного обеспечения классифицируют по «весу», т. е. по количеству формализованных процессов и детальности их регламентации. Следовательно, чем больше процессов документировано, чем более детально описана методология, тем больше будет ее «вес».

1. Тяжеловесные методологии:

- ГОСТ 19 «Единая система программной документации» и ГОСТ 34 «Стандарты на разработку и сопровождение автоматизированных систем» ориентированы на последовательный подход к разработке программного обеспечения;

- Capability Maturity Model for Software (SW-CMM) определяет пять уровней «зрелости проекта»;

- Rational Unified Process (RUP) – итеративная модель разработки;

- Microsoft Solutions Framework (MSF) – база знаний компании Microsoft по разработке программ;

- Personal Software Process – модель определяет требования к компетенциям разработчика. Team Software Process – модель ориентирует на самоуправляемые команды от 3 до 20 разработчиков.

- и др.

2. Легковесные или agile-методики:

- eXtreme Programming или XP – экстремальное программирование, предлагающее 12 инженерных практик;

- Crystal Clear – семейство методологий, определяют необходимую степень формализации процесса разработки в зависимости от количества участников и критичности задач [13];

- Feature Driven Development (FDD) – функционально-ориентированная разработка;

- OpenUP – итеративно-инкрементальный метод разработки программ, позиционируется как легкий и гибкий вариант тяжеловесной методологии RUP;

- Scrum – управление разработкой информационных систем с высокой степенью неопределенности;

- Kanban – методология «бережливого производства»;

- и др.

Результаты исследования Agile Survey о популярности гибких методологий представлены на рис. 1.

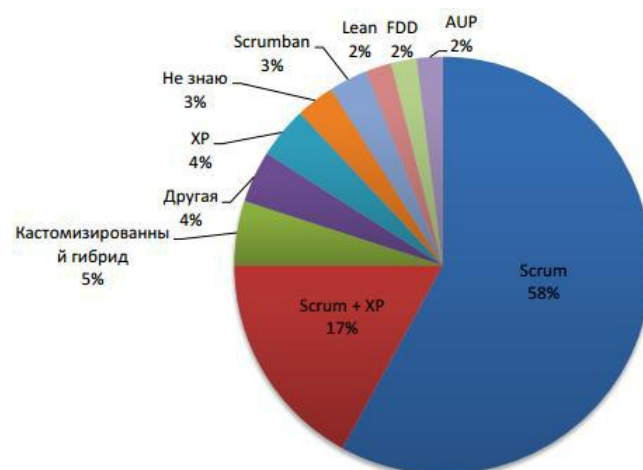


Рис. 1. Популярность гибких методологий

Легковесные методологии Agile появились сравнительно недавно. В феврале 2001 г. 17 специалистов (консультантов и практиков) провели семинар, на котором сформулировали основные принципы гибкой разработки ПО – Agile Manifesto – манифест гибкой разработки. Он переведен на многие языки мира и доступен на сайте <http://agilemanifesto.org/iso/ru/>.

Идея всех гибких (легковесных) методологий состоит в том, что применяемый в разработке ПО процесс должен быть адаптивным, ориентированным на людей и их взаимодействие. По сути это скорее набор практик, чем методология.

На том же семинаре было предложено новое название таких методологий – гибкая разработка Agile Software Development.

При выборе модели управления проектом можно ориентироваться на следующую таблицу (таблица 2).

Таблица 2 – Выбор методологии управления проектом

Вес модели	Достоинства	Недостатки
Тяжеловесные	- процессы рассчитаны на среднюю Квалификацию исполнителей; – большая специализация исполнителей; – низкие требования к стабильности команды; – отсутствуют ограничения по объему и сложности выполняемых проектов.	- требуют существенной управленческой надстройки; – длительные стадии анализа и проектирования; – формализованные коммуникации.
Легковесные (гибкие)	- снижение непроизводительных расходов, связанных с управлением проектом, рисками, изменениями, конфигурациями; – упрощение стадий анализа и проектирования, основной упор на разработку функциональности, совмещение ролей; – неформальные коммуникации	- эффективность сильно зависит от индивидуальных способностей, требуют более квалифицированной, универсальной и стабильной команды; – объем и сложность выполняемых проектов ограничены.

А. Коуберн, один из авторов «Манифеста», провел анализ ИТ-проектов за последние 20 лет [4], выполненных на основании разных моделей управления: от облегченных, гибких до тяжелых (СММ-5). Данный анализ показал отсутствие корреляции между успехом или провалом проектов и выбранными моделями разработки, применяемыми в проектах.

А. Коуберн сделал также вывод о том, что:

1. У каждого проекта должна быть своя модель процесса разработки.
2. У каждой модели – свое время.

Для разработчиков это означает, что не существует единственного правильного процесса разработки. В каждом новом проекте процесс должен определяться каждый раз заново по «закону четырех П» (рис. 2).

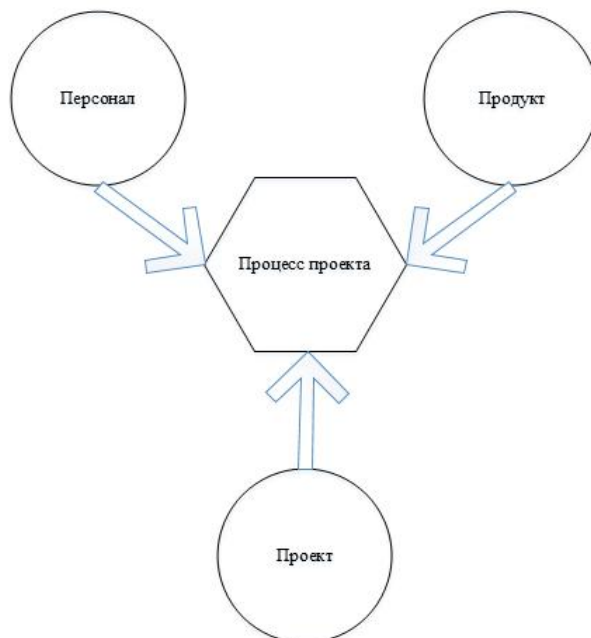


Рис. 2. Закон четырех П

Ход работы

Задание 1. С помощью поиска в сети Интернет найдите информацию о современных методологиях управления ИТ-проектами. Представьте основания для их классификации. Для каждого основания приведите примеры методологий.

Задание 2. Из полученного списка тяжеловесных методологий управления ИТ-проектами выберите один. Проведите исследование методологии. Результат представьте в таблице (таблица 3).

Таблица 3 – Особенности методики

Характеристика	Описание
Полное название методологии	
Авторы	
История возникновения	
Страна появления	
Основные принципы, подходы	
Имеются ли программные средства реализации методологии, какие?	
Используется ли в настоящее время	
Примеры успешных проектов, реализованных с помощью данной методологии	

Задание 3. Из полученного списка легковесных (agile) методологий управления ИТ-проектами выберите один. Проведите исследование методологии. Результат представьте в таблице (таблица 3).

Задание 4. Выберите любую из проанализированных методологий. Создайте о ней презентацию на 10-15 слайдов. Выступите в группе, будьте готовы ответить на вопросы.

Контрольные вопросы:

1. Что такое методология управления ИТ-проектом?
2. Какие виды методологий вы знаете?
3. В чем особенности тяжеловесных и легковесных методологий управления?

4. Приведите примеры методологий, используемых для разработки ИТ-проектов.

По завершении занятия студент должен:

1. Знать понятие методологии управления ИТ-проектами, их виды.
2. Приводить примеры различных методологий.
3. Перечислять преимущества тяжеловесных и легковесных методологий.
4. Осуществлять выбор методологий управления при работе над ИТ-проектом.

Критерии оценивая:

- оценка «*отлично*» выставляется студенту, если задание выполнено в полном объеме, студент хорошо ориентируется в методологиях управления проектами программного обеспечения;
- оценка «*хорошо*» выставляется студенту, если он выполнил задание почти в полном объеме (может отсутствовать пункт), допускает небольшие «проектно-технологические» в работе с методологиями управления проектами программного обеспечения;
- оценка «*удовлетворительно*» выставляется студенту, если он выполнил 3 задания, но составил отчет;
- оценка «*неудовлетворительно*» выставляется студенту, если он выполнил менее 3 заданий и не составил отчет.

3. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА) ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

ОЦЕНКА ЗНАНИЙ ОБУЧАЮЩИХСЯ НА ЭКЗАМЕНЕ – 7 СЕМЕСТР

ПК-1 Способен выявлять информационные потребности пользователей и составлять техническое задание на разработку цифрового программного продукта

ПК-1.2. Разрабатывает техническое задание, формализует требования заказчика к разработке цифрового программного продукта

Обучающийся знает: понятия, особенности и архитектуру программного обеспечения, профессиональные и этические требования профессионального сообщества программистов к деятельности по разработке информационных систем

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам его участия в устных опросах и ответа на экзамене.

1. Что включает понятие «технология разработки программного обеспечения»?
2. Что определяет жизненный цикл программного обеспечения?
3. Поясните содержание каскадной модели разработки программного обеспечения.
4. Поясните содержание итерационной спиральной модели разработки программного обеспечения.
5. Поясните содержание итеративной модели разработки программного обеспечения.
6. Что должен обеспечивать эффективный подход к управлению процессом разработки ПО?
7. Что понимается под зрелостью процессов для компании, разрабатывающей ПО?
8. Приведите основные положения гибкого подхода к созданию ПО.
9. Приведите основное назначение методологии управления жизненным циклом приложений
10. Контроль архитектуры программных средств.
11. Что такое архитектура программного обеспечения?
12. Важность архитектуры программного обеспечения

ПК-2 Способен проектировать цифровой программный продукт

ПК-2.1. Разрабатывает модель цифрового программного продукта с учетом внедрения передовых ИТ-решений прикладной сферы

Обучающийся знает: процесс и средства разработки информационных систем и программных комплексов, а также факторы сложности разработки информационных систем, технологии, подходы и принципы к исследованию и созданию информационных систем, эргономические требования к ним.

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам его участия в устных опросах и ответа на экзамене.

1. Требования к качеству программного обеспечения (безопасность, производительность, возможность параллельной обработки, интернационализация и конфигурация)?
2. Какими характеристиками должен обладать качественный программный продукт?
3. Какие нефункциональные требования определяют качество программного продукта?
4. Какие инструментальные средства предлагает компания Microsoft для управления жизненным циклом приложений?
5. Организация документирования программных средств
6. Требования к документации программных средств
7. Планирование документирования программных средств
8. Состав и содержание документов программных средств
9. Каким способом программное обеспечение развертывается?
10. Кем обслуживаться программное обеспечение в процессе эксплуатации?

ПК-4 Способен управлять процессом разработки цифрового программного продукта

ПК-4.1. Применяет гибкие технологии управления и цифровые средства контроля жизненного цикла разработки цифрового программного продукта.

Обучающийся знает: суть процесса разработки программного обеспечения и виды деятельности в нем, модели жизненного цикла разработки программного обеспечения; принципы конфигурационного управления и управления требованиями, суть методологии MSF, принципы и средства управления программным проектом и командой разработчиков,

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам его участия в устных опросах и ответа на экзамене.

1. Взаимосвязь управления проектами и функционального менеджмента.
2. Перспективы развития управления проектами. Переход к проектному управлению: задачи и этапы решения.
3. Классификация базовых понятий управления проектами. Классификация типов проектов. Цель и стратегия проектов. Результат проекта. Управление параметрами проекта. Проектный цикл.
4. Общая характеристика программных проектов.
5. Процессы управления проектом. Уровни зрелости процессов управления проектами.
6. Модели жизненного цикла ИТ-продукта. Соотношение жизненного цикла ИТ-решения и жизненного цикла проекта.
7. Теории управления программным проектом.
8. Классификация методов, моделей и стандартов разработки программного обеспечения. Методологии быстрой адаптивной разработки Agile(SCRUM).
9. Методологии разработки и внедрения ИТ-решений.
10. Какая роль тестирования в обеспечении качества программного продукта?
11. Какие типы тестов используют для проверки качества программного продукта?
12. Для чего применяется регрессионное тестирование?

ПК-4 Способен управлять процессом разработки цифрового программного продукта

ПК-4.2. Обеспечивает разработку проектной и технической документации цифрового программного продукта

Обучающийся знает: нормативные документы и стандарты в процессе создания информационных систем.

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам его участия в устных опросах и ответа на экзамене.

1. Что такое программная инженерия?
2. Назовите дату зарождения программной инженерии как отдельной науки.
3. В чем отличие программной инженерии от информатики?
4. В чем отличие программной инженерии от системотехники?
5. Приведите примеры дисциплин информатики и программной инженерии (дисциплины не путать с учебными предметами).
6. Что такое ПО?
7. С какими иными видами человеческой деятельности соотносится создание ПО в данном разделе?
8. Процессы управления проектом. Уровни зрелости процессов управления проектами.
9. Модели жизненного цикла ИТ-продукта. Соотношение жизненного цикла ИТ-решения и жизненного цикла проекта.
10. Теории управления программным проектом.
11. Классификация методов, моделей и стандартов разработки программного обеспечения. Методологии быстрой адаптивной разработки Agile(SCRUM).
12. Методологии разработки и внедрения ИТ-решений.

ПК-5 Способен обеспечивать качество работы цифрового программного продукта

ПК-5.3. Осуществляет работы по интеграционному тестированию цифрового программного продукта с внешним окружением

Обучающийся знает: международные и отечественные стандарты оценки качества разработки цифровых программных продуктов; методы и способы тестирования программного обеспечения основы эффективной организации разработки цифрового программного продукта и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта.

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам его участия в устных опросах и ответа на экзамене.

1. Краткая характеристика программных средств и стадии ее разработки
2. Определение трудоемкости разработки программных средств
3. Определение состава группы исполнителей
4. Расчет и построение сетевого плана-графика разработки и внедрения программных средств
5. Стандарты документирования программного обеспечения
6. Понятие архитектуры и задачи ее описания.
7. Основные классы архитектур программных средств.
8. Какие шаблоны тестовых проектов имеются в Visual Studio?
9. Для чего применяется Microsoft Test Manager? Какие он имеет функциональные возможности?
10. Для чего используется Lab Management при тестировании?
11. Для чего применяют рефакторинг кода?
12. Приведите признаки некачественного кода.
13. Взаимодействие между подсистемами и архитектурные функции.
14. Чем отличаются между собой текущий и конкретный процессы? Какие методологии разработки ПО поддерживают понятие конкретного процесса и какими средствами?

ОЦЕНКА УМЕНИЙ И НАВЫКОВ НА ЭКЗАМЕНЕ

ПК-1 Способен выявлять информационные потребности пользователей и составлять техническое задание на разработку цифрового программного продукта

ПК-1.2. Разрабатывает техническое задание, формализует требования заказчика к разработке цифрового программного продукта

Обучающийся умеет: составлять техническую документацию процесса разработки программного обеспечения.

Обучающийся владеет: навыками разработки отдельных технических документов.

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практических работ.

ПК-2 Способен проектировать цифровой программный продукт

ПК-2.1. Разрабатывает модель цифрового программного продукта с учетом внедрения передовых ИТ-решений прикладной сферы

Обучающийся умеет: определять факторы сложности разработки программных систем, выбирать адекватные технологии для их проектирования.

Обучающийся владеет: навыками критического анализа технической информации.

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практических работ.

ПК-4 Способен управлять процессом разработки цифрового программного продукта

ПК-4.1. Применяет гибкие технологии управления и цифровые средства контроля жизненного цикла разработки цифрового программного продукта.

Обучающийся умеет: планировать этапы разработки информационной системы с использованием инструментальных средств.

Обучающийся владеет: навыками управления программным проектом с помощью программно-технических средств, навыками работы с сервисами контроля версий программного продукта.

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практических работ.

ПК-4 Способен управлять процессом разработки цифрового программного продукта

ПК-4.2. Обеспечивает разработку проектной и технической документации цифрового программного продукта

Обучающийся умеет: применять нормативные документы и стандарты в процессе создания информационных систем.

Обучающийся владеет: навыками разработки информационных систем с использованием нормативных документов и стандартов.

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практических работ.

ПК-5 Способен обеспечивать качество работы цифрового программного продукта

ПК-5.3. Осуществляет работы по интеграционному тестированию цифрового программного продукта с внешним окружением

Обучающийся умеет: осуществлять эффективную организацию разработки цифрового программного продукта и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта, проводить тестирование цифровых программных продуктов

Обучающийся владеет: навыками проверки работоспособности информационных систем и эффективность интеграции всех технологических процессов, навыками построения плана и протокола тестирования.

Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам выполнения практической работы.

ОБРАЗЕЦ ЭКЗАМЕНАЦИОННОГО БИЛЕТА

Билет состоит из трех теоретических вопросов.

Тольяттинская Академия управления	Дисциплина Программная инженерия
Вариант 1	
1. В чем отличие программной инженерии от системотехники?	
2. Модели жизненного цикла ИТ-продукта. Соотношение жизненного цикла ИТ-решения и жизненного цикла проекта.	
3. Расскажите о причинах отсутствия универсального процесса разработки ПО.	

4. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРЫ ОЦЕНОВАНИЯ ЗНАНИЙ, УМЕНИЙ И НАВЫКОВ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ

КРИТЕРИИ ОЦЕНИВАНИЯ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
ПК-1 Способен выявлять информационные потребности пользователей и составлять техническое задание на разработку цифрового программного продукта ПК-1.2. Разрабатывает техническое задание, формализует требования заказчика к разработке цифрового программного продукта					
Знать: понятия, особенности и архитектуру программного обеспечения, профессиональные и этические требования профессионального сообщества программистов к деятельности по разработке информационн	Не знает	Фрагментарны е знания о понятиях, особенностях и архитектуре программного обеспечения, профессиональ ных и этических требованиях профессиональ ного сообщества программистов к деятельности по разработке информационн	Общие, но не структурирова нные знания о понятиях, особенностях и архитектуре программного обеспечения, профессиональ ных и этических требованиях профессиональ ного сообщества программистов к деятельности по разработке	Сформирован ные, но содержащие отдельные пробелы знания о понятиях, особенностях и архитектуре программног о обеспечения, профессионал ьных и этических требованиях профессионал ьного	Сформированн ые систематизиро ванные прочные знания о понятиях, особенностях и архитектуре программного обеспечения, профессиональ ных и этических требованиях профессиональ ного сообщества

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
ых систем		ых систем	информационн ых систем	сообщества программист ов к деятельности по разработке информационных систем	программистов к деятельности по разработке информационных систем
Уметь: составлять техническую документацию процесса разработки программного обеспечения	Не умеет	Частично освоенные умения составлять техническую документацию процесса разработки программного обеспечения	В целом освоенные, но не подкрепляемые знаниями и самостоятельно стью выполнения умения составлять техническую документацию процесса разработки программного обеспечения	В целом сформированные и самостоятельные выполняемые, но не всегда интеллектуально обоснованные умения составлять техническую документацию процесса разработки программного обеспечения	Успешно сформированные, самостоятельные и осознанно выполняемые умения составлять техническую документацию процесса разработки программного обеспечения
Владеть: навыками разработки отдельных технических документов	Не владеет	Фрагментарно сформированные навыки разработки технических спецификаций на web-ресурсы	В целом сформированные, но выполняемые на элементарном уровне навыки разработки технических спецификаций на web-ресурсы	Сформированные, но не устойчивые в сложных ситуациях навыки разработки технических спецификаций на web-ресурсы	Успешно сформированные, устойчивые, технологически и грамотно выполняемые навыки разработки технических спецификаций на web-ресурсы

ПК-2 Способен проектировать цифровой программный продукт

ПК-2.1. Разрабатывает модель цифрового программного продукта с учетом внедрения передовых ИТ-решений прикладной сферы

Знать: - процесс и средства разработки информационных систем и программных комплексов;	Не знает	Фрагментарные знания о процессе и средствах разработки информационных систем и программных комплексов;	Общие, но не структурированные знания о процессе и средствах разработки информационных систем и программных	Сформированные, но содержащие отдельные пробелы знания о процессе и средствах разработки	Сформированные систематизированные прочные знания о процессе и средствах разработки
--	----------	--	---	--	---

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
- факторы сложности разработки информационных систем, технологии, подходы и принципы к исследованию и созданию информационных систем, эргономические требования к ним		о факторах сложности разработки информационных систем, технологиях, подходах и принципах к исследованию и созданию информационных систем, эргономических требованиях к ним	комплексов; о факторах сложности разработки информационных систем, технологиях, подходах и принципах к исследованию и созданию информационных систем, эргономических требованиях к ним	информационных систем и программных комплексов; о факторах сложности разработки информационных систем, технологиях, подходах и принципах к исследованию и созданию информационных систем, эргономических требованиях к ним	информационных систем и программных комплексов; о факторах сложности разработки информационных систем, технологиях, подходах и принципах к исследованию и созданию информационных систем, эргономических требованиях к ним
Уметь: - определять факторы сложности разработки программных систем, выбирать адекватные технологии для их проектирования	Не умеет	Частично освоенные умения определять факторы сложности разработки программных систем, выбирать адекватные технологии для их проектирования	В целом освоенные, но не подкрепляемые знаниями и самостоятельностью выполнения умения определять факторы сложности разработки программных систем, выбирать адекватные технологии для их проектирования	В целом сформированные и самостоятельные выполняемые, но не всегда интеллектуально обоснованные умения определять факторы сложности разработки программных систем, выбирать адекватные технологии для их проектирования	Успешно сформированные, самостоятельно и осознанно выполняемые умения определять факторы сложности разработки программных систем, выбирать адекватные технологии для их проектирования
Владеть: навыками критического анализа технической информации	Не владеет	Фрагментарно сформированные навыки разработки архитектуры web-приложения с	В целом сформированные, но выполняемые на элементарном уровне навыки	Сформированные, но не устойчивые в сложных ситуациях навыки разработки	Успешно сформированные, устойчивые, технологические и грамотно выполняемые

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
		учетом внедрения сквозных цифровых технологий и передовых ИТ-решений прикладной сферы	разработки архитектуры web-приложения с учетом внедрения сквозных цифровых технологий и передовых ИТ-решений прикладной сферы	архитектуру web-приложения с учетом внедрения сквозных цифровых технологий и передовых ИТ-решений прикладной сферы	навыки разработки архитектуры web-приложения с учетом внедрения сквозных цифровых технологий и передовых ИТ-решений прикладной сферы
ПК-4 Способен управлять процессом разработки цифрового программного продукта ПК-4.1. Применяет гибкие технологии управления и цифровые средства контроля жизненного цикла разработки цифрового программного продукта.					
Знать: - суть процесса разработки программного обеспечения и виды деятельности в нем, модели жизненного цикла разработки программного обеспечения; - принципы конфигурационного управления и управления требованиями, суть методологии MSF - принципы и средства управления программным проектом и командой разработчиков	Не знает	Фрагментарные знания о сути процесса разработки программного обеспечения и видах деятельности в нем, моделях жизненного цикла разработки программного обеспечения; принципах конфигурационного управления и управления требованиями, сути методологии MSF принципах и средства управления программным проектом и командой разработчиков	Общие, но не структурированные знания о сути процесса разработки программного обеспечения и видах деятельности в нем, моделях жизненного цикла разработки программного обеспечения; принципах конфигурационного управления и управления требованиями, сути методологии MSF принципах и средства управления программным проектом и командой разработчиков	Сформированные, но содержащие отдельные пробелы знания о сути процесса разработки программного обеспечения и видах деятельности в нем, моделях жизненного цикла разработки программного обеспечения; принципах конфигурационного управления и управления требованиями, сути методологии MSF принципах и средства управления программным проектом и командой разработчиков	Сформированные систематизированные прочные знания о сути процесса разработки программного обеспечения и видах деятельности в нем, моделях жизненного цикла разработки программного обеспечения; принципах конфигурационного управления и управления требованиями, сути методологии MSF принципах и средства управления программным проектом и командой разработчиков

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
				в	
Уметь: планировать этапы разработки информационной системы с использованием инструментальных средств	Не умеет	Частично освоенные умения планировать этапы разработки информационной системы с использованием инструментальных средств	В целом освоенные, но не подкрепляемые знаниями и самостоятельно выполнению умения планировать этапы разработки информационной системы с использованием инструментальных средств	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения планировать этапы разработки информационной системы с использованием инструментальных средств	Успешно сформированные, самостоятельно и осознанно выполняемые умения планировать этапы разработки информационной системы с использованием инструментальных средств
Владеть: - навыками управления программным проектом с помощью программно-технических средств, - навыками работы с сервисами контроля версий программного продукта	Не владеет	Фрагментарно сформированные навыки управления программным проектом с помощью программно-технических средств и навыки работы с сервисами контроля версий программного продукта	В целом сформированные, но выполняемые на элементарном уровне навыки управления программным проектом с помощью программно-технических средств и навыки работы с сервисами контроля версий программного продукта	Сформированные, но не устойчивые в сложных ситуациях навыки управления программным проектом с помощью программно-технических средств и навыки работы с сервисами контроля версий программного продукта	Успешно сформированные, устойчивые, технологически и грамотно выполняемые навыки управления программным проектом с помощью программно-технических средств и навыки работы с сервисами контроля версий программного продукта
ПК-4 Способен управлять процессом разработки цифрового программного продукта ПК-4.2. Обеспечивает разработку проектной и технической документации цифрового программного продукта					
Знать: нормативные документы и стандарты в процессе создания	Не знает	Фрагментарные знания о нормативных документах и стандартах в процессе	Общие, но не структурированные знания о нормативных документах и стандартах в	Сформированные, но содержащие отдельные пробелы знания о	Сформированные систематизированные прочные знания о

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
информационных систем		создания информационных систем	процессе создания информационных систем	нормативных документах и стандартах в процессе создания информационных систем	нормативных документах и стандартах в процессе создания информационных систем
Уметь: применять нормативные документы и стандарты в процессе создания информационных систем	Не умеет	Частично освоенные умения применять нормативные документы и стандарты в процессе создания информационных систем	В целом освоенные, но не подкрепляемые знаниями и самостоятельно выполнению умения применять нормативные документы и стандарты в процессе создания информационных систем	В целом сформированные и самостоятельные, но не всегда выполняемые, интеллектуально обоснованные умения применять нормативные документы и стандарты в процессе создания информационных систем	Успешно сформированные, самостоятельные и осознанно выполняемые умения применять нормативные документы и стандарты в процессе создания информационных систем
Владеть: навыками разработки информационных систем с использованием нормативных документов и стандартов	Не владеет	Фрагментарно сформированные навыки разработки информационных систем с использованием нормативных документов и стандартов	В целом сформированные, но выполняемые на элементарном уровне навыки разработки информационных систем с использованием нормативных документов и стандартов	Сформированные, но не устойчивые в сложных ситуациях навыки разработки информационных систем с использованием нормативных документов и стандартов	Успешно сформированные, устойчивые, технологически и грамотно выполняемые навыки разработки информационных систем с использованием нормативных документов и стандартов
ПК-5 Способен обеспечивать качество работы цифрового программного продукта ПК-5.3. Осуществляет работы по интеграционному тестированию цифрового программного продукта с внешним окружением					
Знать: основы эффективной организации разработки информационных систем и	Не знает	Фрагментарные знания об основах эффективной организации разработки информационных систем	Общие, но не структурированные знания об основах эффективной организации разработки	Сформированные, но содержащие отдельные пробелы знания об основах	Сформированные систематизированные прочные знания об основах

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта		ых систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта	информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта	эффективной организации разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта	эффективной организации разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта
Уметь: осуществлять эффективную организацию разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта	Не умеет	Частично освоенные умения осуществлять эффективную организацию разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта	В целом освоенные, но не подкрепляемые знаниями и самостоятельно выполнению умения осуществлять эффективную организацию разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения осуществлять эффективную организацию разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта	Успешно сформированные, самостоятельно осознанно выполняемые умения осуществлять эффективную организацию разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта
Владеть: навыками проверки работоспособности	Не владеет	Фрагментарно сформированные навыки проверки	В целом сформированные, но выполняемые	Сформированные, но не устойчивые в сложных	Успешно сформированные, устойчивые,

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
ости информационн ых систем и эффективность интеграции всех технологическ их процессов навыками построения плана и протокола тестирования		работоспособн ости информационн ых систем и эффективность интеграции всех технологическ их процессов	на элементарном уровне навыки проверки работоспособн ости информационн ых систем и эффективность интеграции всех технологическ их процессов	ситуациях навыки проверки работоспособ ности информацион ных систем и эффективност ь интеграции всех технологичес ких процессов	технологическ и грамотно выполняемые навыки проверки работоспособн ости информационн ых систем и эффективность интеграции всех технологическ их процессов

ПРОЦЕДУРА ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ И КРИТЕРИИ ОЦЕНКИ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Для контроля усвоения обучающимися данной дисциплины и достижения запланированных результатов обучения учебным планом предусмотрены экзамен. Экзамен проводится в форме устного ответа на три теоретических вопроса. Результаты текущего контроля являются допуском к экзамену: обучающийся должен выполнить все практические задания и тест на отметку не менее чем «удовлетворительно».

В ходе промежуточной аттестации перевод результатов работы обучающихся (результатов текущего контроля, рейтинговых баллов) в систему оценки знаний осуществляется с ориентацией на критерии оценивания сформированности компетенций (см. Таблицу далее) следующим образом:

- оценка «отлично» выставляется обучающемуся, который показал сформированные систематизированные прочные знания о документах, сопровождающих цикл управления разработкой программного обеспечения, и ошибок документирования; о сути процесса разработки программного обеспечения и видах деятельности в нем, о моделях жизненного цикла разработки программного обеспечения; о принципах и средствах управления программным проектом и командой разработчиков; о методах и способах тестирования программного обеспечения; об особенностях программного обеспечения, факторов сложности его разработки, технологий, подходов и принципов к исследованию и созданию информационных систем, эргономических требований к ним; о международных и отечественных стандартах оценки качества разработки информационных систем; базовых издержек на разработку программных средств и крупных программных комплексов, возможных рисков функционирования информационных систем; об архитектуре, процессе, особенностей и средств разработки информационных систем и крупных программных комплексов.
- оценка «хорошо» выставляется обучающемуся, который показал сформированные, но содержащие отдельные пробелы знания о документах, сопровождающих цикл управления разработкой программного обеспечения, и ошибок документирования; о

сути процесса разработки программного обеспечения и видах деятельности в нем, о моделях жизненного цикла разработки программного обеспечения; о принципах и средствах управления программным проектом и командой разработчиков; о методах и способах тестирования программного обеспечения; об особенностях программного обеспечения, факторов сложности его разработки, технологий, подходов и принципов к исследованию и созданию информационных систем, эргономических требований к ним; о международных и отечественных стандартах оценки качества разработки информационных систем; базовых издержек на разработку программных средств и крупных программных комплексов, возможных рисков функционирования информационных систем; об архитектуре, процессе, особенностях и средств разработки информационных систем и крупных программных комплексов..

- оценка «удовлетворительно» выставляется обучающемуся, который показал общие, но не структурированные знания о документах, сопровождающих цикл управления разработкой программного обеспечения, и ошибок документирования; о сути процесса разработки программного обеспечения и видах деятельности в нем, о моделях жизненного цикла разработки программного обеспечения; о принципах и средствах управления программным проектом и командой разработчиков; о методах и способах тестирования программного обеспечения; об особенностях программного обеспечения, факторов сложности его разработки, технологий, подходов и принципов к исследованию и созданию информационных систем, эргономических требований к ним; о международных и отечественных стандартах оценки качества разработки информационных систем; базовых издержек на разработку программных средств и крупных программных комплексов, возможных рисков функционирования информационных систем; об архитектуре, процессе, особенностях и средств разработки информационных систем и крупных программных комплексов..
- оценка «неудовлетворительно» выставляется обучающемуся, который имеет фрагментарные знания о документах, сопровождающих цикл управления разработкой программного обеспечения, и ошибок документирования; о сути процесса разработки программного обеспечения и видах деятельности в нем, о моделях жизненного цикла разработки программного обеспечения; о принципах и средствах управления программным проектом и командой разработчиков; о методах и способах тестирования программного обеспечения; об особенностях программного обеспечения, факторов сложности его разработки, технологий, подходов и принципов к исследованию и созданию информационных систем, эргономических требований к ним; о международных и отечественных стандартах оценки качества разработки информационных систем; базовых издержек на разработку программных средств и крупных программных комплексов, возможных рисков функционирования информационных систем; об архитектуре, процессе, особенностях и средств разработки информационных систем и крупных программных комплексов.

5. ДИАГНОСТИЧЕСКИЕ ЗАДАНИЯ ДЛЯ ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ХАРАКТЕРИЗУЮЩИХ УРОВЕНЬ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ

ПК-1 Способен выявлять информационные потребности пользователей и составлять техническое задание на разработку цифрового программного продукта

Обучающийся знает: понятия, особенности и архитектуру программного обеспечения, профессиональные и этические требования профессионального сообщества программистов к деятельности по разработке информационных систем.

Обучающийся умеет: составлять техническую документацию процесса разработки программного обеспечения.

Обучающийся владеет: навыками разработки технических спецификаций на web-ресурсы.

ЗАДАНИЯ ЗАКРЫТОГО ТИПА

1. Что такое DevOps?

- а) Процесс интеграции разработки и тестирования.
- б) Набор практик и инструментов для автоматизации процессов.
- в) Стратегия управления проектами.
- г) Всё вышеперечисленное.

Верный ответ: г) Всё вышеперечисленное.

2. Какие инструменты DevOps используются для контроля версий?

- а) Git и SVN.
- б) Jenkins и Puppet.
- в) Docker и Ansible.
- г) Все вышеперечисленные инструменты.

Верный ответ: г) Все вышеперечисленные инструменты.

3. Что такое контроль версий?

- а) Процесс отслеживания изменений кода.
- б) Метод управления конфигурацией.
- в) Способ автоматизации развёртывания.
- г) Всё вышеперечисленное.

Верный ответ: г) Всё вышеперечисленное.

4. Что такое инфраструктура как код (IaC)?

- а) Управление инфраструктурой с помощью инструментов автоматизации.
- б) Создание виртуальных машин с помощью скриптов.
- в) Настройка сетевых параметров с использованием шаблонов.
- г) Всё вышеперечисленное.

Верный ответ: г) Всё вышеперечисленное.

5. Как IaC реализуется с использованием AWS?

- а) С помощью сервисов AWS CloudFormation и AWS OpsWorks.
- б) С использованием инструментов Puppet и Chef.
- в) С применением сервисов AWS Elastic Beanstalk и AWS Lambda.
- г) Все вышеперечисленные методы.

Верный ответ: г) Все вышеперечисленные методы.

6. Что такое автоматизация развёртывания (deployment automation)?

- а) Процесс обновления приложений на серверах.
- б) Использование инструментов для настройки серверов.
- в) Применение методов для быстрого развёртывания новых версий приложений.
- г) Всё вышеперечисленное.

Верный ответ: в) Применение методов для быстрого развёртывания новых версий приложений.

7. Какие основные принципы DevOps?

- а) Сотрудничество, непрерывность и автоматизация.
- б) Быстрота, гибкость и надёжность.
- в) Эффективность, оптимизация и стабильность.
- г) Все вышеперечисленные принципы.

Верный ответ: а) Сотрудничество, непрерывность и автоматизация.

8. Какие основные этические принципы программирования сформулированы в Кодексе этики и профессиональной практики программной инженерии IEEE?

- а) Вклад в общество и благополучие человека.
- б) Избегание причинения вреда другим.
- в) Честность и заслуживание доверия.
- г) Уважение к частной жизни других людей.

Верный ответ: б) Избегание причинения вреда другим.

9. Какие основные принципы этики программирования касаются отношения к интеллектуальной собственности?

- а) Вклад в общество и благополучие человека.
- б) Избегание причинения вреда другим.
- в) Честность и заслуживание доверия.
- г) Уважение к частной жизни других людей.

Верный ответ: а) Вклад в общество и благополучие человека.

10. Какие основные принципы этики программирования касаются конфиденциальности?

- а) Вклад в общество и благополучие человека.
- б) Избегание причинения вреда другим.
- в) Честность и заслуживание доверия.
- г) Уважение к частной жизни других людей.

Верный ответ: г) Уважение к частной жизни других людей.

ЗАДАНИЯ ОТКРЫТОГО ТИПА

Задание 1: Какие виды технической документации существуют и для чего они нужны?

Ответ: существует три вида технической документации: пользовательская, технологическая и проектная. Они предназначены для описания задач и функций программы, настройки и поддержки ПО, а также объяснения логики принятых проектных решений.

Задание 2: На каком этапе разработки следует начинать составлять техническую документацию?

Ответ: документацию следует начинать писать во время разработки ПО, а не после завершения процесса. Это позволит зафиксировать детали создания программы и логику решений.

Задание 3: Кто является целевой аудиторией технической документации?

Ответ: целевая аудитория технической документации зависит от вида самой документации. Например, пользовательская документация предназначена для обычных пользователей, технологическая — для разработчиков и администраторов, а проектная — для специалистов, принимающих решения о разработке.

Задание 4: Какие основные элементы должны быть включены в техническую документацию?

Ответ: техническая документация должна содержать спецификации, технические требования, базы данных, описание функций ПО, список ошибок и их причин, дизайн-макеты, руководство для пользователей и администраторов.

Задание 5: Какие программы и сервисы можно использовать для составления технической документации?

Ответ: для составления технической документации можно использовать программы и сервисы, такие как Doxygen, Sphinx, GitHub и GitBook.

Задание 6: Как обеспечить актуальность и полноту информации в технической документации?

Ответ: для обеспечения актуальности и полноты информации в технической документации следует указывать дату обновления документа, использовать полные данные и краткие тексты, избегать витиеватых выражений и шуток, а также регулярно обновлять и проверять код.

Задание 7: Как облегчить процесс составления технической документации для разработчиков?

Ответ: облегчить процесс составления технической документации можно с помощью использования специализированных программ и сервисов, таких как Doxygen, Sphinx, GitHub и GitBook, а также следуя принципам краткости, актуальности и визуала (таблицы, диаграммы, схемы).

Задание 8: Что такое техническое задание (ТЗ) и какова его роль в управлении проектами?

Ответ: Техническое задание (ТЗ) — это документ, который содержит детальное описание требований, характеристик и параметров для проекта. Оно играет важную роль в управлении проектами, обеспечивая ясное понимание требований и ожиданий между сторонами.

Задание 9: Что такое техническое задание на создание сайта и из каких разделов оно состоит?

Ответ: Техническое задание на создание сайта — это документ, который подробно описывает все требования и ожидания заказчика от разрабатываемого веб-сайта. Разделы ТЗ включают введение, общее описание сайта, структуру и навигацию, дизайн и макет, функциональность, требования к контенту и техническим требованиям.

Задание 10: Зачем нужно ТЗ при создании сайта и что оно даёт заказчику?

Ответ: ТЗ помогает избежать недоразумений между заказчиком и разработчиками, обеспечивает чёткое понимание требований и спецификаций проекта, а также служит основой для оценки выполненной работы.

Задание 11: Какие основные разделы должны быть в техническом задании на создание сайта?

Ответ: Основные разделы ТЗ включают введение, общее описание сайта, структуру и навигацию, дизайн и макет, функциональность, требования к контенту и технические требования.

Задание 12: Что такое адаптивный дизайн и почему он важен для современных веб-ресурсов?

Ответ: Адаптивный дизайн — это подход к созданию веб-сайтов, при котором сайт корректно отображается на разных устройствах и экранах, включая мобильные телефоны и планшеты. Он важен, потому что позволяет пользователям удобно просматривать сайт на разных устройствах и повышает удобство использования.

Задание 13: Какие типы контента могут быть размещены на веб-сайте и какие требования к ним предъявляются?

Ответ: Типы контента, размещаемого на веб-сайте, включают текст, изображения, видео, аудио. Требования к контенту могут включать размер файлов, форматы файлов, разрешение изображений и другие параметры, зависящие от специфики проекта.

Задание 14: Что такое юзабилити и как оно влияет на успешность веб-ресурса?

Ответ: Юзабилити — это показатель удобства использования веб-ресурса. Оно влияет на успешность веб-ресурса, потому что пользователи должны легко находить нужную информацию, выполнять необходимые действия и не сталкиваться с проблемами при использовании сайта.

Задание 15: Что такое SEO-оптимизация и как она помогает продвигать веб-ресурс в интернете?

Ответ: SEO-оптимизация — это комплекс мер, направленных на повышение видимости веб-ресурса в поисковых системах. Она помогает продвигать веб-ресурс в интернете, потому что позволяет улучшить позиции сайта в результатах поиска, привлечь больше целевых посетителей и увеличить конверсию.

ПК-2 Способен проектировать цифровой программный продукт

Обучающийся знает: процесс и средства разработки информационных систем и программных комплексов; факторы сложности разработки информационных систем, технологии, подходы и принципы к исследованию и созданию информационных систем, эргономические требования к ним.

Обучающийся умеет: определять факторы сложности разработки программных систем, выбирать адекватные технологии для их проектирования.

Обучающийся владеет: навыками разработки архитектуры web-приложения с учетом внедрения сквозных цифровых технологий и передовых ИТ-решений прикладной сферы.

ЗАДАНИЯ ЗАКРЫТОГО ТИПА

1. Какой компонент является основой информационной системы?

- а) Структура данных.
- б) Компьютерная сеть.
- в) Методы обработки информации.
- г) Вычислительная мощность компьютера.
- д) База данных.

Правильный ответ: а) Структура данных.

2. Какой стандарт регламентирует жизненный цикл программного обеспечения?

- а) ГОСТ 34.601–90.
- б) ISO/IEC 12207:1995.
- в) ISO/IEC 15288:2002.
- г) Custom Development Method.
- д) Rational Unified Process (RUP).

Правильный ответ: а) ГОСТ 34.601–90.

3. Что такое MVC в контексте веб-разработки?

- а) Модульный веб-компонент
 - б) Модуль контроля версий
 - с) Модель-вид-контроллер
 - д) Многоверсионный контейнер
- Правильный ответ: с) Модель-вид-контроллер

4. Какая роль отвечает за представление данных в MVC архитектуре?

- а) Модель

- b) Вид
- c) Контроллер
- d) Сервер

Правильный ответ: b) Вид

5. Что такое RESTful API?

- a) Программный интерфейс для доступа к базе данных
- b) Модуль управления сетью
- c) Архитектурный стиль для построения веб-сервисов
- d) Контроллер обработки запросов

Правильный ответ: c) Архитектурный стиль для построения веб-сервисов

6. Какая роль отвечает за бизнес-логику в MVC архитектуре?

- a) Модель
- b) Вид
- c) Контроллер
- d) Хранилище данных

Правильный ответ: a) Модель

7. Какая методика используется для построения модели «черного ящика»?

- a) Функциональная методика IDEF0.
- b) Объектно-ориентированная методика.
- в) Синтетическая методика.
- г) Диаграммы дерева узлов и FEO.
- д) Стоимостный анализ.

Правильный ответ: a) Функциональная методика IDEF0.

8. Какие инструменты относятся к средствам поддержки жизненного цикла программного обеспечения?

- a) Графические редакторы.
- б) Средства разработки приложений.
- в) Инструменты тестирования.
- г) CASE-средства.
- д) Все вышеперечисленное.

Правильный ответ: д) Все вышеперечисленное.

9. Какие функции выполняют CASE-средства?

- a) Хранение и разработка версий проекта.
- б) Графическое представление моделей ИС.
- в) Разработка приложений.
- г) Конфигурационное управление.
- д) Все вышеперечисленное.

Правильный ответ: д) Все вышеперечисленное.

10. Какой тип архитектуры обычно используется для создания масштабируемых веб-приложений?

- a) Монолитная архитектура
- б) Клиент-серверная архитектура
- с) Микросервисная архитектура
- д) Централизованная архитектура

Правильный ответ: c) Микросервисная архитектура

ЗАДАНИЯ ОТКРЫТОГО ТИПА

11. Опишите процесс разработки архитектуры информационной системы.

Ответ: Разработка архитектуры информационной системы обычно включает в себя несколько этапов. На первом этапе проводится анализ требований, в ходе которого

определяются основные функциональные и нефункциональные требования к приложению. Затем происходит проектирование, на этом этапе решается, какие технологии, инструменты и архитектурные шаблоны будут использоваться для реализации приложения. После этого разрабатывается архитектурный дизайн, включающий в себя построение модели данных, определение компонентов приложения и их взаимосвязей. Наконец, на этапе реализации кодируются компоненты приложения согласно разработанной архитектуре.

12. Какие основные преимущества имеет микросервисная архитектура перед монолитной?

Ответ: Микросервисная архитектура предлагает ряд преимуществ перед монолитной архитектурой. Во-первых, микросервисы могут быть независимо развернуты и масштабированы, что облегчает поддержку и управление приложением. Во-вторых, они обеспечивают гибкость и модульность, позволяя быстрее разрабатывать, тестировать и развертывать новые функции. Кроме того, микросервисы позволяют использовать различные технологии и языки программирования для каждого сервиса, что способствует выбору наилучшего инструмента для конкретной задачи.

13. Что такое RESTful API и почему его часто используют в веб-приложениях?

Ответ: RESTful API - это архитектурный стиль для построения веб-сервисов, который основан на принципах REST (Representational State Transfer). Он позволяет организовать взаимодействие между клиентом и сервером с помощью стандартных HTTP методов, таких как GET, POST, PUT и DELETE, и обмен данными в формате JSON или XML. RESTful API широко используется в веб-приложениях из-за своей простоты, гибкости и масштабируемости. Он упрощает разработку и интеграцию различных компонентов приложения, а также обеспечивает четкую структуру запросов и ответов.

14. Какие основные компоненты входят в MVC архитектуру?

Ответ: MVC (Model-View-Controller) - это архитектурный шаблон, который разделяет приложение на три основных компонента: модель (Model), представление (View) и контроллер (Controller). Модель отвечает за бизнес-логику и работу с данными приложения, представление отображает данные пользователю и обрабатывает пользовательский ввод, а контроллер управляет взаимодействием между моделью и представлением, а также обрабатывает запросы от пользователя.

15. Что такое SPA (Single Page Application) и какие преимущества оно предлагает?

Ответ: SPA (Single Page Application) - это тип веб-приложения, которое загружает единственную HTML страницу и динамически обновляет ее содержимое без перезагрузки страницы при взаимодействии пользователя. Основные преимущества SPA включают более быструю загрузку и отзывчивость интерфейса, так как большая часть ресурсов загружается один раз при запуске приложения, а также улучшенный пользовательский опыт благодаря плавным переходам и анимациям.

16. Какие методы аутентификации могут быть использованы в информационных системах?

Ответ: В информационных системах могут использоваться различные методы аутентификации, включая аутентификацию на основе токенов (например, JWT), аутентификацию с использованием сессий, аутентификацию через социальные сети, а также аутентификацию с помощью сторонних сервисов (например, OAuth). Выбор метода зависит от требований к безопасности, удобства использования и функциональности приложения.

17. Какие инструменты можно использовать для тестирования производительности информационных систем?

Ответ: Для тестирования производительности информационных систем можно использовать различные инструменты, включая Apache JMeter, LoadRunner, Gatling, Siege и другие.

18. Задание: Разработайте модель информационной системы для онлайн-магазина. Опишите структуру базы данных, включающую таблицы для хранения информации о продуктах, пользователях, заказах и платежах. Предложите схему взаимодействия клиентской и серверной частей приложения, включая функциональные возможности каждой части.

Решение: Для модели информационной системы онлайн-магазина мы можем разработать следующую структуру базы данных:

Таблица «Продукты» с полями: id продукта, название, описание, цена, количество на складе и т.д.

Таблица «Пользователи» с полями: id пользователя, имя, адрес электронной почты, хэш пароля и т.д.

Таблица «Заказы» с полями: id заказа, id пользователя, дата заказа, статус заказа и т.д.

Таблица «Платежи» с полями: id платежа, id заказа, сумма, дата и т.д.

Взаимодействие клиентской и серверной частей приложения будет осуществляться с помощью RESTful API, где клиент будет отправлять запросы на добавление товаров в корзину, оформление заказа и оплату, а сервер будет обрабатывать эти запросы, взаимодействуя с базой данных.

19. Задание: Представьте модель информационной системы для социальной сети.

Опишите структуру базы данных для хранения профилей пользователей, сообщений, фотографий, друзей и лайков.

Обсудите архитектурные решения для реализации новостной ленты, системы уведомлений и поиска пользователей.

Решение: Для модели информационной системы социальной сети мы можем разработать следующую структуру базы данных:

Таблица «Пользователи» с полями: id пользователя, имя, дата рождения, фото профиля и т.д.

Таблица «Сообщения» с полями: id сообщения, id отправителя, id получателя, текст сообщения, дата отправки и т.д.

Таблица «Друзья» с полями: id пользователя, id друга и т.д.

Таблица «Лайки» с полями: id пользователя, id записи, дата лайка и т.д.

Архитектурное решение будет включать в себя разделение приложения на клиентскую и серверную части, где клиентская часть будет отвечать за отображение профилей пользователей, новостной ленты и личных сообщений, а серверная часть будет обрабатывать запросы от клиента и взаимодействовать с базой данных.

20. Задание: Разработайте модель информационной системы для онлайн-обучения.

Опишите сущности базы данных, такие как курсы, уроки, студенты и преподаватели.

Предложите структуру для организации курсов, включая моделирование уровней доступа и рейтинговой системы для оценки успеваемости студентов.

Решение: Для модели информационной системы онлайн-обучения мы можем разработать следующую структуру базы данных:

Таблица «Курсы» с полями: id курса, название, описание, длительность и т.д.

Таблица «Уроки» с полями: id урока, id курса, название, описание, материалы и т.д.

Таблица «Студенты» с полями: id студента, имя, адрес электронной почты, курсы и т.д.

Таблица «Преподаватели» с полями: id преподавателя, имя, адрес электронной почты, преподаваемые курсы и т.д.

Архитектурное решение будет включать в себя создание интерфейса для студентов и преподавателей, где студенты смогут просматривать курсы, регистрироваться на них и проходить уроки, а преподаватели смогут добавлять новые уроки и оценивать успеваемость студентов.

21. Задание: Представьте модель информационной системы для финансового учета.

Опишите базу данных для хранения информации о транзакциях, счетах пользователей, бюджетах и категориях расходов. Рассмотрите алгоритмы для автоматического распознавания и категоризации транзакций.

Решение: Для модели информационной системы финансового учета мы можем разработать следующую структуру базы данных:

Таблица «Транзакции» с полями: id транзакции, id пользователя, сумма, категория, дата и т.д.

Таблица «Счета» с полями: id счета, id пользователя, баланс, валюта и т.д.

Таблица «Категории расходов» с полями: id категории, название, описание и т.д.

Архитектурное решение включает разработку интерфейса для пользователей, позволяющего им добавлять новые транзакции, просматривать и анализировать свои расходы и доходы, а также управлять своими счетами и бюджетами.

22. Задание: Разработайте модель информационной системы для электронной медицинской карты. Обсудите структуру базы данных для хранения информации о пациентах, истории болезней, назначениях и результатах анализов. Рассмотрите механизмы обеспечения безопасности и конфиденциальности данных.

Решение: Для модели информационной системы электронной медицинской карты мы можем разработать следующую структуру базы данных:

Таблица «Пациенты» с полями: id пациента, имя, дата рождения, медицинская история и т.д.

Таблица «Диагнозы» с полями: id диагноза, id пациента, дата постановки, описание и т.д.

Таблица «Назначения» с полями: id назначения, id пациента, дата, описание, лекарства и т.д.

Архитектурное решение будет включать в себя создание защищенного интерфейса для врачей и медицинского персонала, где они смогут просматривать и редактировать медицинские записи пациентов.

23. Задание: Представьте модель информационной системы для планирования путешествий. Опишите структуру базы данных для хранения информации о турах, отелях, бронированиях и отзывах пользователей.

Решение: Для модели информационной системы для планирования путешествий нам необходимо разработать структуру базы данных, которая будет содержать информацию о турах, отелях, бронированиях и отзывах пользователей. Давайте опишем возможную структуру базы данных:

Таблица «Туры»:

tour_id: уникальный идентификатор тура

название: название тура

описание: описание тура

стоимость: стоимость тура

дата_начала: дата начала тура

дата_окончания: дата окончания тура

место_посещения: место, которое будет посещено в рамках тура

тип_тура: тип тура (например, экскурсионный, пляжный и т.д.)

Таблица «Отели»:

hotel_id: уникальный идентификатор отеля

название: название отеля

описание: описание отеля

адрес: адрес отеля

рейтинг: рейтинг отеля

цена_за_ночь: стоимость проживания за ночь

Таблица «Бронирования»:

booking_id: уникальный идентификатор бронирования

tour_id: идентификатор тура, на который осуществляется бронирование

hotel_id: идентификатор отеля, в котором осуществляется бронирование

дата_заселения: дата заселения в отель

дата_выселения: дата выселения из отеля

количество_гостей: количество гостей в брони

статус: статус бронирования (подтверждено, ожидает подтверждения и т.д.)

Таблица «Отзывы»:

review_id: уникальный идентификатор отзыва

user_id: идентификатор пользователя, оставившего отзыв

tour_id: идентификатор тура, к которому относится отзыв

отель_id: идентификатор отеля, к которому относится отзыв

оценка: оценка тура или отеля

текст_отзыва: текст отзыва

24. Рассмотрите способы интеграции с сервисами бронирования и картами для отображения местоположения для предметной области из предыдущего задания (задание 23):

Решение:

1. Интеграция с сервисами бронирования:

Мы можем интегрировать API сервисов бронирования, таких как Booking.com или Airbnb, чтобы пользователи могли просматривать доступные отели и совершать бронирование без покидания нашего приложения.

Для этого мы можем использовать асинхронные запросы к API сервисов бронирования для получения доступных вариантов и отображения их на странице приложения. После того как пользователь сделает выбор и заполнит данные для бронирования, мы можем отправить запрос на создание брони через API сервиса бронирования.

2. Интеграция с картами для отображения местоположения:

Мы можем использовать картографические сервисы, такие как Google Maps или Mapbox, для отображения местоположения туров и отелей на карте.

Для этого мы можем использовать JavaScript API картографического сервиса для создания интерактивной карты на странице приложения. Местоположение туров и отелей можно будет отобразить на карте с помощью соответствующих маркеров или областей. Пользователи смогут взаимодействовать с картой, выбирая интересующие их места или маршруты.

25. Задание: Разработайте модель информационной системы для онлайн-заказа еды. Обсудите структуру базы данных для хранения меню, заказов, клиентов и доставок. Предложите интерфейс для администраторов ресторанов для управления меню и заказами.

Для модели информационной системы онлайн-заказа еды нам необходимо разработать структуру базы данных, которая будет содержать информацию о меню, заказах, клиентах и доставках. Предложим также интерфейс для администраторов ресторанов для управления меню и заказами.

Вот предложенная структура базы данных:

1. Таблица "Меню":

item_id: уникальный идентификатор блюда

название: название блюда

описание: описание блюда

цена: цена блюда

2. Таблица «Заказы»:

order_id: уникальный идентификатор заказа

client_id: идентификатор клиента, сделавшего заказ

item_id: идентификатор блюда из меню

количество: количество порций блюда в заказе

статус: статус заказа (принят, готовится, доставляется и т.д.)

дата_заказа: дата и время размещения заказа

3. Таблица "Клиенты":

client_id: уникальный идентификатор клиента

имя: имя клиента

адрес: адрес доставки

телефон: контактный телефон клиента

email: адрес электронной почты клиента

4. Таблица «Доставки»:

delivery_id*: уникальный идентификатор доставки

order_id: идентификатор заказа

статус: статус доставки (в пути, доставлено и т.д.)

дата_доставки: дата и время доставки

Интерфейс для администраторов ресторанов может включать следующие функции:

1. Управление меню:

- Добавление, редактирование и удаление блюд из меню.

- Изменение цен на блюда.

- Добавление описания к блюдам.

2. Управление заказами:

- Просмотр списка всех заказов с информацией о клиенте, составе заказа и его статусе.

- Возможность изменения статуса заказа (например, отметка о том, что заказ готов к доставке).

- Просмотр истории заказов.

3. Управление доставками:

- Просмотр списка всех активных доставок с информацией о заказе, адресе доставки и статусе доставки.

- Возможность отметки о доставке заказа (подтверждение доставки).

- Просмотр истории доставок.

4. Управление клиентами:

- Просмотр списка всех клиентов с контактной информацией.

- Возможность добавления новых клиентов в случае оформления заказа от нового клиента.

- Возможность редактирования контактной информации клиентов.

Таким образом, создав данную структуру базы данных и предоставив администраторам ресторанов удобный интерфейс управления, мы можем эффективно организовать процесс онлайн-заказа еды.

ПК-4 Способен управлять процессом разработки цифрового программного продукта

Обучающийся знает: суть процесса разработки программного обеспечения и виды деятельности в нем, модели жизненного цикла разработки программного обеспечения; принципы конфигурационного управления и управления требованиями, суть методологии MSF; принципы и средства управления программным проектом и командой разработчиков; нормативные документы и стандарты в процессе создания информационных систем.

Обучающийся умеет: планировать этапы разработки информационной системы с использованием инструментальных средств; применять нормативные документы и стандарты в процессе создания информационных систем.

Обучающийся владеет: навыками управления программным проектом с помощью программно-технических средств; навыками работы с сервисами контроля версий программного продукта; навыками разработки информационных систем с использованием нормативных документов и стандартов.

ЗАДАНИЯ ЗАКРЫТОГО ТИПА

1. Какие этапы входят в жизненный цикл программы?

а) Формулирование требований, анализ требований, проектирование, тестирование, эксплуатация.

б) Формулирование требований, анализ требований, проектирование, тестирование,

оптимизация, эксплуатация.

в) Формулирование требований, анализ требований, проектирование, тестирование, корректировка ошибок, эксплуатация.

г) Формулирование требований, анализ требований, проектирование, автономное тестирование, комплексное тестирование, эксплуатация.

Верный ответ: г) Формулирование требований, анализ требований, проектирование, автономное тестирование, комплексное тестирование, эксплуатация.

2. Какая модель жизненного цикла программного обеспечения предполагает разделение процесса разработки на последовательные этапы?

а) Итерационная модель.

б) Последовательная модель.

в) Гибкая модель.

г) Каскадная модель.

Правильный ответ: г) Каскадная модель.

3. Как называется модель жизненного цикла программного обеспечения, которая включает в себя разработку, тестирование и развёртывание информационных систем?

а) SDLC.

б) Agile.

в) Waterfall.

г) Iterative.

Правильный ответ: а) SDLC.

4. Какая модель жизненного цикла программного обеспечения предполагает многократное повторение фаз разработки и тестирования до полного соответствия продукта требованиям заказчика?

а) Итерационная модель.

б) Каскадная модель.

в) Гибкая модель.

г) Последовательная модель.

Правильный ответ: а) Итерационная модель.

5. Как называется модель жизненного цикла программного обеспечения, основанная на принципе кодирования и исправления ошибок?

а) Каскадная модель.

б) Инкрементная модель.

в) Водопадная модель.

г) Итерационная модель.

Правильный ответ: б) Инкрементная модель.

6. Что является основой методологии MSF?

а) Принципы конфигурационного управления и управления требованиями.

б) Принципы объектно-ориентированного программирования.

в) Принципы Agile.

г) Принципы экстремального программирования.

Правильный ответ: а) Принципы конфигурационного управления и управления требованиями.

7. Что является основной целью модели проектной группы MSF?

а) Создание качественного продукта.

б) Минимизация затрат на разработку.

в) Удовлетворение потребностей заказчика.

г) Соблюдение сроков проекта.

Правильный ответ: в) Удовлетворение потребностей заказчика.

8. Как MSF определяет роли в проектной группе?

- а) На основе квалификации участников.
- б) На основе их предпочтений.
- в) На основе потребностей проекта.
- г) На основе опыта работы в команде.

Правильный ответ: в) На основе потребностей проекта.

9. Какие стандарты регулируют разработку, внедрение и сопровождение информационных систем?

- а) Стандарты по документации на ИС.
- б) Стандарты по процессам документирования ИС.
- в) Стандарты по оценке качества документации на ИС.
- г) Стандарты по информационной безопасности (кибербезопасности), относящиеся к процессам документирования ИС.

Правильный ответ: г) Стандарты по информационной безопасности (кибербезопасности), относящиеся к процессам документирования ИС.

10. Какие государственные стандарты Российской Федерации (межгосударственные стандарты для стран СНГ) относятся к Единой системе программной документации (ЕСПД)?

- а) Стандарты по документации на ИС.
- б) Стандарты по процессам документирования ИС.
- в) Стандарты по оценке качества документации на ИС.
- г) Стандарты по информационной безопасности (кибербезопасности), относящиеся к процессам документирования ИС.

Правильный ответ: а) Стандарты по документации на ИС.

ЗАДАНИЯ ОТКРЫТОГО ТИПА

Задание 1: Какие государственные стандарты и международные стандарты используются для разработки и внедрения информационных систем?

Ответ: Для разработки и внедрения информационных систем используются государственные стандарты Российской Федерации (ГОСТ) и межгосударственные стандарты для стран СНГ. Также применяются международные стандарты, такие как ISO (Международная организация по стандартизации) и ИЕС (Международная электротехническая комиссия).

Задание 2: Какие стандарты регулируют документацию на информационные системы (ИС)?

Ответ: Стандарты, регулирующие документацию на ИС, включают Единую систему программной документации (ЕСПД) и комплекс стандартов с индексом «19». Эти стандарты устанавливают взаимосвязанные правила разработки, оформления и обращения программ и программной документации, обеспечивая унификацию, снижение трудоёмкости и повышение эффективности разработки документации.

Задание 3: Как стандарты по информационной безопасности (кибербезопасности) влияют на процессы документирования ИС?

Ответ: Стандарты по информационной безопасности относятся к процессам документирования ИС и определяют требования к разработке, сопровождению, изготовлению и эксплуатации программ. Они обеспечивают возможность унификации программных изделий, снижения трудоёмкости и повышения эффективности разработки, а также автоматизации изготовления и хранения программной документации.

Задание 4: Какие стандарты используются для оценки качества документации на ИС?

Ответ: Для оценки качества документации на ИС используются стандарты ГОСТ 28195-89, ГОСТ Р ИСО/МЭК 25041-2014, ГОСТ Р ИСО/МЭК ТО 9294-93 и СТ РК 34.010-2002. Эти

стандарты определяют требования к оценке программной и эксплуатационной документации, обеспечивая сертификацию или аттестацию информационных систем.

Задание 5: Планирование задач. Опишите возможности программного обеспечения для управления проектами в планировании задач и составлении расписания.

Ответ: Программное обеспечение для управления проектами позволяет планировать задачи, составлять расписание и определять критический путь проекта. Оно также помогает идентифицировать вехи проекта и декомпозировать его структуру.

Задание 6: Управление данными и предоставление информации. Объясните, какую информацию предоставляет программное обеспечение для управления проектами и как оно управляет данными.

Ответ: Программное обеспечение предоставляет информацию о списке задач, ресурсах, сроках выполнения задач, рисках и рабочей нагрузке. Оно также позволяет управлять коммуникациями команды проекта и предоставлять доступ к информации о ходе проекта.

Задание 7: Управление коммуникациями команды проекта. Опишите, как программное обеспечение для управления проектами помогает в обсуждении и согласовании рабочих вопросов проекта, фиксации проблем и предоставлении доступа к информации.

Ответ: Программное обеспечение позволяет обсуждать рабочие вопросы проекта, фиксировать проблемы и запросы на изменения, обрабатывать риски и предоставлять доступ к информации о ходе проекта через живую ленту событий.

Задание 8: Типы программного обеспечения для управления проектами. Сравните настольные, веб-интерфейсные, персональные и многопользовательские системы управления проектами. Укажите их преимущества и недостатки.

Ответ: Настольные системы предоставляют гибкий интерфейс и возможность сохранения информации в файл. Веб-интерфейсные системы доступны с любого компьютера и имеют простой контроль доступа, но скорость работы может быть ниже. Персональные системы используются для управления домашними проектами, а многопользовательские системы предназначены для координации действий нескольких пользователей. Преимущества и недостатки зависят от типа системы и конкретных потребностей проекта.

Задание 9: Что такое система контроля версий (СКВ) и каковы её основные функции?

Ответ: СКВ — это инструмент, используемый разработчиками программного обеспечения для управления изменениями в исходном коде и других файловых ресурсах. Функции СКВ включают отслеживание изменений, восстановление предыдущих версий, слияние изменений и контроль доступа.

Задание 10: Какие типы систем контроля версий существуют и чем они отличаются?

Ответ: Типы СКВ включают локальные, централизованные и распределённые системы. Локальные системы хранят изменения на локальном компьютере разработчика, централизованные системы используют центральный сервер, а распределённые системы распределяют историю проекта между несколькими копиями репозитория.

Задание 11: Какие популярные системы контроля версий используются в разработке программного обеспечения?

Ответ: Наиболее популярные системы контроля версий — это Git, Subversion (SVN) и Mercurial.

Задание 12: Что такое репозиторий в системе контроля версий и как с ним работать?

Ответ: Репозиторий — это хранилище, в котором хранятся все версии и история изменений проекта. Разработчики могут работать с репозиторием, скачивать последние версии файлов, вносить изменения и загружать их обратно.

Задание 13: Какие команды используются в Git для работы с репозиторием?

Ответ: Git предоставляет множество команд для работы с репозиторием, включая git init, git clone, git add, git commit, git push, git pull, git branch и git merge.

Задание 14: Какие типы конфликтов могут возникнуть при слиянии изменений в системе контроля версий и как их разрешить?

Ответ: При слиянии изменений могут возникнуть конфликты, такие как конфликты слияния, конфликты удаления и конфликты имени файла. Разработчик должен вручную разрешить конфликт, выбрав правильные изменения.

Задание 15: Как осуществляется контроль качества разработки информационных систем с использованием нормативных документов и стандартов?

Ответ: Контроль качества разработки информационных систем осуществляется путём проверки соответствия разработанных документов требованиям ГОСТов и стандартов, а также проведения тестирования и проверки работоспособности системы.

ПК-5 Способен обеспечивать качество работы цифрового программного продукта

Обучающийся знает: основы эффективной организации разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта.

Обучающийся умеет: осуществлять эффективную организацию разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта.

Обучающийся владеет: навыками проверки работоспособности информационных систем и эффективность интеграции всех технологических процессов.

ЗАДАНИЯ ЗАКРЫТОГО ТИПА

Вопрос 1:

Что определяет двухуровневая архитектура клиент-сервер?

- а) Доступ к удалённой базе данных и файловым ресурсам.
- б) Удалённое представление данных.
- в) Удалённое приложение.
- г) Распределённое представление данных.

Правильный ответ: а).

Вопрос 2:

Что такое архитектура клиент-сервер?

- а) Вычислительная или сетевая архитектура.
- б) Сетевая архитектура.
- в) Вычислительная архитектура.
- г) Сетевая и вычислительная архитектура.

Правильный ответ: а).

Вопрос 3:

Какой принцип пользовательского интерфейса предполагает, что все функции и данные, необходимые для решения определённой задачи, должны быть видны пользователю?

- а) Принцип структуризации.
- б) Принцип обратной связи.
- в) Принцип видимости.
- г) Принцип толерантности.

Верный ответ: в) Принцип видимости.

Вопрос 4:

Какой принцип пользовательского интерфейса предполагает, что интерфейс должен быть гибким и терпимым к ошибкам пользователя?

- а) Принцип структуризации.

- б) Принцип обратной связи.
 - в) Принцип видимости.
 - г) Принцип толерантности.
- Верный ответ: г) Принцип толерантности.

Вопрос 5:

Какой принцип пользовательского интерфейса предполагает, что пользователь должен получать сообщения о действиях системы и о важных событиях внутри неё?

- а) Принцип структуризации.
- б) Принцип обратной связи.
- в) Принцип видимости.
- г) Принцип простоты.

Верный ответ: б) Принцип обратной связи.

Вопрос 6:

Какой принцип пользовательского интерфейса предполагает, что пользовательский интерфейс должен быть целесообразно структурирован?

- а) Принцип структуризации.
- б) Принцип обратной связи.
- в) Принцип видимости.
- г) Принцип простоты.

Верный ответ: а) Принцип структуризации.

Вопрос 7:

Что является основной целью юзабилити-тестирования?

- а) Измерение отношения пользователей к продукту.
- б) Оценка пользовательского опыта.
- в) Определение барьеров для выполнения задач пользователями.
- г) Анализ навигации по сайту.

Верный ответ: в) Определение барьеров для выполнения задач пользователями.

Вопрос 8:

Что помогает выявить юзабилити-тестирование?

- а) Уровень пользовательского опыта.
- б) Осведомлённость пользователей о бренде.
- в) Мотивацию пользователей посещать сайт.
- г) Предпочтения пользователей в отношении дизайна.

Верный ответ: в) Мотивацию пользователей посещать сайт.

Вопрос 9:

Что такое юзабилити-аудит?

- а) Оценка удобства использования веб-приложения для пользователей.
- б) Анализ структуры и навигации веб-приложения.
- в) Тестирование производительности веб-приложения.
- г) Проверка соответствия веб-приложения требованиям заказчика.

Правильный ответ: а) Оценка удобства использования веб-приложения для пользователей.

Вопрос 10:

Какие основные этапы включает юзабилити-аудит?

- а) Сбор информации, анализ, рекомендации.
- б) Анализ конкурентов, тестирование прототипов.
- в) Разработка дизайна, внедрение изменений.
- г) Анализ данных, разработка стратегии.

Правильный ответ: а) Сбор информации, анализ, рекомендации.

ЗАДАНИЯ ОТКРЫТОГО ТИПА

Задание 1: Опишите основные принципы юзабилити-тестирования и объясните, почему они важны для создания удобного пользовательского интерфейса.

Ответ:

Основные принципы юзабилити-тестирования:

1. Тестирование с участием реальных пользователей. Позволяет получить обратную связь от целевой аудитории и выявить проблемы, незаметные для разработчиков.
2. Раннее начало тестирования. Начинайте проверять идеи на ранних стадиях разработки, чтобы избежать больших затрат времени и ресурсов на исправление ошибок.
3. Выявление проблем с удобством использования. Пользователи могут указать на недочёты в навигации, сложности поиска информации и другие проблемы, которые разработчики могут исправить.
4. Понимание поведения пользователей. Наблюдение за взаимодействием пользователей с продуктом даёт представление о том, как они будут его использовать, и позволяет оптимизировать пользовательский опыт.
5. Юзабилити-тестирование важно для создания удобного пользовательского интерфейса, потому что оно помогает:
 - ~ проверить работоспособность идей перед большими затратами времени и ресурсов;
 - ~ обнаружить и исправить проблемы удобства использования, которые могут остаться незамеченными разработчиками;
 - ~ понять поведение пользователей и обеспечить наилучший пользовательский опыт;
 - ~ оптимизировать затраты и сэкономить время, исключая возможные недочёты до выпуска продукта на рынок.

Задание 2: Расскажите о различных методах сбора данных при проведении юзабилити-тестирования и оцените их эффективность.

Ответ:

Методы сбора данных при юзабилити-тестировании включают:

Сортировка карточек — тестирование и создание архитектуры сайта.

Бумажные прототипы — оценка информационной структуры на ранней стадии.

Цифровые низкодетализированные прототипы — простая навигация и ограниченная обратная связь.

Высокодетализированные прототипы — получение точной и содержательной обратной связи.

Отслеживание кликов — информация о наиболее часто нажимаемых клавишах.

Отслеживание взглядов — определение приоритетных областей для пользователей.

Лабораторное тестирование — выполнение заданий с соблюдением сценария.

Этнографическое тестирование — наблюдение за пользователями в реальной среде.

Подготовка дизайна респондентами — разработка дизайна с использованием предоставленных инструментов.

Фокус-группы — обсуждение продукта с группой участников.

Глубинные интервью — личные встречи с модераторами для обсуждения потребностей пользователей.

Эффективность этих методов зависит от конкретной ситуации и целей тестирования. Выбор метода определяется стадией разработки продукта, бюджетом и доступностью ресурсов.

Задание 3: Как систематизировать и анализировать данные, полученные в результате юзабилити-тестирования? Какие метрики следует учитывать?

Ответ:

Для систематизации и анализа данных юзабилити-тестирования следуйте этим шагам:

Определите категории данных: выберите переменные, которые хотите проанализировать, например, идентификатор участника, номер задачи, статус завершения, время выполнения, ошибки, оценки удовлетворённости и комментарии.

Создайте структуру таблицы: используйте электронные таблицы (Excel или Google Sheets) для построения таблицы с соответствующими столбцами для каждой категории данных.

Заполните данные: введите информацию в ячейки таблицы для каждого участника и задания.

Проанализируйте данные: изучите всю информацию, собранную в ходе тестирования, определите ключевые показатели (выполнение задач, время, частота ошибок, удовлетворённость пользователей, комментарии) и суммируйте количественные данные.

Проанализируйте качественные данные: проведите тематический анализ комментариев и наблюдений пользователей, выявите повторяющиеся темы, проблемы и закономерности.

Группируйте и категоризируйте результаты: распределите результаты по функциям или темам оцениваемого продукта.

Определите приоритеты: расставьте приоритеты для выявленных проблем юзабилити на основе степени серьёзности, частоты возникновения и влияния на пользовательский опыт.

Составьте отчёт: представьте результаты тестирования с указанием проблем, рекомендаций по улучшению и приоритетов.

Задание 4: Какие бывают типы ошибок пользователей при взаимодействии с интерфейсом, которые могут быть выявлены с помощью юзабилити-тестирования? Приведите примеры каждого типа.

Ответ:

1. Типы ошибок пользователей при взаимодействии с интерфейсом, выявляемых с помощью юзабилити-тестирования:
2. Структура: отсутствие логики или дублирование элементов. Пример: наличие нескольких форм регистрации на сайте.
3. Поиск: незаметный или неудобный. Пример: отсутствие подсказки при вводе поискового запроса.
4. Хлебные крошки: отсутствие или расположение вне зоны видимости. Пример: невозможность отследить путь к нужной странице.
5. Листинг: отсутствие подробных описаний товара, услуги или цен. Пример: трудности с определением стоимости товара.
6. Оформление заказа: человеку предлагается пройти несколько этапов для оформления заказа. Пример: сложная процедура регистрации.
7. Тексты: отсутствие абзацев и форматирования, недостоверная информация, нет блоков для выделения главной информации или подзаголовков. Пример: запутанное описание товара или услуги.
8. Визуализация: нет медиаматериалов, сопровождающих тексты, в результате теряется интерес к изучению информации. Пример: отсутствие изображений или видео для демонстрации товара.
9. Дизайн: слишком пёстрое оформление с большим количеством деталей, не имеющих практического значения и не несущих пользы для посетителей. Пример: избыток анимации или мигающих элементов.
10. Скорость загрузки: ресурс загружается слишком долго. Пример: медленная загрузка страниц сайта.
11. 404 страница и битые ссылки: негативно влияют на ранжирование. Пример: некорректные ссылки на страницах сайта.
12. Адаптивная вёрстка: неудобство при чтении сайта с мобильных устройств. Пример: проблемы с отображением сайта на экранах смартфонов.
13. Капча: слишком сложная или неудобная. Пример: капча, требующая от пользователя выполнения сложных действий для подтверждения регистрации.
14. Кнопки: на сайте есть неработающие кнопки. Пример: кнопки, которые не выполняют своих функций.
15. Поддержка: отсутствие онлайн-чатов либо они нерабочие. Пример: невозможность связаться с технической поддержкой сайта.

16. Интерфейс: большое количество цветов, неудобные шрифты, сложный дизайн. Пример: запутанные меню и навигация по сайту.

Задание 5: Определите цели тестирования информационной системы.

Цель тестирования: проверить корректность работы системы бронирования авиабилетов.

Ответ: для достижения этой цели необходимо провести тестирование функциональности системы, включая регистрацию пользователей, поиск и бронирование авиабилетов, оплату и подтверждение бронирования, а также обработку возвратов и изменений.

Задание 6: Определите стратегии тестирования.

Стратегия тестирования: модульное, интеграционное и системное тестирование.

Ответ: модульное тестирование проводится для проверки отдельных компонентов системы, интеграционное тестирование — для проверки взаимодействия между компонентами, а системное тестирование — для проверки работы системы в целом.

Задание 7: Составьте график тестирования.

График тестирования:

этап 1: модульное тестирование основных функций системы (1 неделя);

этап 2: интеграционное тестирование взаимодействия компонентов (2 недели);

этап 3: системное тестирование и исправление ошибок (3 недели).

Ответ: график тестирования должен быть составлен таким образом, чтобы обеспечить своевременное выполнение всех этапов и устранение возможных ошибок.

Задание 8: Какие принципы составления хороших сценариев пользовательского интерфейса?

Ответ:

Принципы составления хороших сценариев пользовательского интерфейса:

1. Определение целевой аудитории и проработка портрета клиента.
2. Выяснение причин, по которым пользователи приходят на сайт, и их целей.
3. Пошаговое описание достижения целей пользователями.
4. Создание пользовательских историй, концептуальных и конкретных сценариев.
5. Разработка сценариев использования, описывающих пользовательский опыт по шагам.

Задание 9: Какие методики исследования юзабилити могут применяться на стадии прототипирования интерфейса?

Ответ:

На стадии прототипирования интерфейса могут применяться следующие методики исследования юзабилити:

- ~ фокус-группы;
- ~ глубинное интервью (Customer Development);
- ~ проверка концепции;
- ~ привлечение к проектированию;
- ~ айтрекинг;
- ~ анализ кликов;
- ~ экспертный обзор.

Задание 10: Какие преимущества имеет проведение юзабилити-тестирования в реальной среде пользователя, а не в искусственных условиях лаборатории?

Ответ:

Преимущества проведения юзабилити-тестирования в реальной среде пользователя:

Получение объективной и точной оценки взаимодействия с продуктом.

Возможность наблюдать за реальным поведением пользователей и их реакциями на продукт.

Сбор ценных отзывов и предложений от пользователей, что помогает улучшить продукт.

Учёт особенностей целевой аудитории и её предпочтений.

Возможность выявить и устранить проблемы на ранних стадиях разработки, экономя время и деньги.

Создание интуитивно понятного и удобного дизайна, стимулирующего спрос на продукт.

Задание 11: Как можно улучшить пользовательский интерфейс на основе результатов юзабилити-тестирования? Какие шаги следует предпринять на основе полученных данных?

Ответ:

Чтобы улучшить пользовательский интерфейс на основе результатов юзабилити-тестирования, следуйте шагам:

1. Проанализируйте полученные данные и определите основные проблемы и слабые места пользовательского интерфейса.
2. Разработайте план изменений и улучшений на основе выявленных проблем.
3. Внедрите предложенные изменения в пользовательский интерфейс.
4. Проведите повторное юзабилити-тестирование обновлённого интерфейса, чтобы оценить эффективность изменений.
5. Внесите дополнительные корректировки и улучшения на основе результатов повторного тестирования.
6. Повторяйте этот процесс до тех пор, пока пользовательский интерфейс не станет максимально удобным и интуитивно понятным для пользователей.

Задание 12: Какие виды испытаний информационной системы существуют и в чём их различия?

Ответ: виды испытаний информационной системы делятся на функциональные и нефункциональные тесты. Функциональные тесты проверяют корректность предоставления системой функциональности пользователям, а нефункциональные тесты оценивают производительность, надёжность, эргономичность и другие свойства системы.

Задание 13: Что такое интеграционное тестирование и зачем оно нужно?

Ответ: интеграционное тестирование — это испытания, направленные на выявление проблем взаимодействия отдельных компонентов системы. Оно проводится для проверки корректности работы системы в целом и обнаружения возможных конфликтов между её составляющими.

Задание 14: Что такое тестирование прототипа и зачем оно нужно?

Ответ: тестирование прототипа — это испытания информационной системы на ранних этапах разработки, когда некоторые её функциональные блоки ещё не готовы. В этом случае недостающие компоненты заменяются функциональными заглушками, имитирующими их будущую работу. Тестирование прототипа позволяет оценить работоспособность системы и выявить возможные проблемы до начала основного этапа разработки.

Задание 15: Что такое нагрузочное тестирование и зачем оно нужно?

Ответ: нагрузочное тестирование — это испытание информационной системы в условиях прогнозируемой нормальной нагрузки. Оно оценивает количество запросов к системе и время их обработки, проверяя, способна ли система справляться с ожидаемым количеством пользователей и обрабатывать запросы без превышения заданных временных ограничений.

6. ЛИСТ СОГЛАСОВАНИЯ

Составил:

Н.Б. Стрекалова, д.п.н., доцент



(подпись)

Заведующий кафедрой

Н.Б. Стрекалова, д.п.н., доцент



(подпись)

Заведующий выпускающей кафедрой

Н.Б. Стрекалова, д.п.н., доцент



(подпись)