ЧОУ ВО «Тольяттинская академия управления»

Кафедра прикладной информатики и высшей математики

Б1.В.14

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА)

Учебная дисциплина	*Программная инженерия*
По направлению подготовки	*09.03.03*
	«Прикладная информатика»
Профиль (программа бакалавриата)	*«Прикладная информатика в цифровой экономике»*
Форма обучения	*Очная*

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

2023

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 «Прикладная информатика в цифровой экономике» (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 информационной системы	ПК-1.2. Составляет техническое задание на разработку информационной системы	**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;	**8 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 информационные системы	ПК-2.2. Разрабатывает модель информационной системы, в том числе с опорой на современные облачные решения	**Знать:** - процесс и средства разработки информационных систем и программных комплексов; - факторы сложности разработки информационных	**8 семестр** Тема 2. Разработка сложных программных систем Тема 4. Процесс разработки программного обеспечения	- устный опрос по темам 2, 5 - практическая работа по теме 4 - выступление с докладом по теме 4 - ответ на

		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	Тема 5. DevOps и инфраструктурная инженерия	теоретический вопрос экзамена
	ПК-2.3. Составляет технико-экономическое обоснование проектных решений	**Знать:** - понятие технико-экономического обоснования , - базовые характеристики затрат на разработку программных средств; **Уметь:** - прогнозировать и оценивать объем ресурсов, необходимых для создания программных продуктов и комплексов - осуществлять выбор проектного решения с опорой на международные стандарты и методы оценки качества разработки программного обеспечения и современные требования к информационным системам **Владеть:** - навыками проведения технико-экономического	**8 семестр** Тема 3. Оценка качества процессов создания программного обеспечения	- устный опрос по теме 3 - ответ на теоретический вопрос экзамена

		обоснования проектного решения; навыками проведения системного анализа прикладной области;		
ПК-4 Способен управлять процессами создания информационных систем	ПК-4.1. Применяет гибкие технологии управления и цифровые средства контроля жизненного цикла создания информационных систем	**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 информационной системы с использованием инструментальных средств; **Владеть:** - навыками управления программным проектом с помощью программно-технических средств, - навыками работы с сервисами контроля версий программного продукта	**8 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
	ПК-4.2. Применяет актуальные нормативные документы и стандарты при создании информационных систем, используя общедоступные справочно-	**Знать:** - нормативные документы и стандарты в процессе создания информационных систем **Уметь:** - применять нормативные	**8 семестр** Тема 3. Оценка качества процессов создания программного обеспечения Тема 4. Процесс разработки программного	- устный опрос по теме 3 - практическая работа по теме 4 - выступление с докладом по теме 4 - ответ на

	правовые системы и профессиональные базы данных сферы ИТ	документы и стандарты в процессе создания информационных систем **Владеть:** навыками разработки информационных систем с использованием нормативных документов и стандартов.	обеспечения	теоретический вопрос экзамена
ПК-5 Способен обеспечивать качество разработки информационных систем	ПК-5.1. Проводит оценку качества программной разработки на разных этапах жизненного цикла информационных систем с привлечением современных отраслевых решений ИТ-сферы	**Знать:** - международные и отечественные стандарты оценки качества разработки информационных систем; - методы и способы тестирования программного обеспечения **Уметь:** - проводить тестирование информационных систем **Владеть:** - навыками построения плана и протокола тестирования;	**8 семестр** Тема 3. Оценка качества процессов создания программного обеспечения Тема 7. Тестирование ПО	- устный опрос по теме 3 - практическая работа по теме 7 - ответ на теоретический вопрос экзамена
	ПК-5.2. Проводит оценку рисков работы информационных систем в условиях цифровизации бизнеса и применения сквозных цифровых и отраслевых технологий	**Знать:** - возможные риски функционирования информационных систем - методы и способы оценки рисков информационных систем **Уметь:** - анализировать издержки и риски при создании информационных систем; - разрабатывать план упреждения рисков в процессе создания информационных систем **Владеть:** - навыками оценки рисков в различных моделях информационных	**8 семестр** Тема 7. Тестирование ПО	- практическая работа по теме 7 - ответ на теоретический вопрос экзамена

		систем;		
	ПК-5.3. Обеспечивает эффективную организацию разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта.	**Знать:** - основы 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 **Владеть:** - навыками проверки работоспособности информационных систем и эффективность интеграции всех технологических процессов	**8 семестр** Тема 5. DevOps и инфраструктурная инженерия	- устный опрос по теме 5 - ответ на теоретический вопрос экзамена

2. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ (СРЕДСТВА) ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ И ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ОБЕСПЕЧИВАЮЩИХ ФОРМИРОВАНИЕ 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 жизненные 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?

Вопросы для проведения устного опроса
по теме «Технико-экономическое обоснование проектов программных средств»

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. Организация документирования программных средств
6. Требования к документации программных средств
7. Планирование документирования программных средств
8. Состав и содержание документов программных средств
9. Стандарты документирования программного обеспечения.

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

1. Какие шаблоны тестовых проектов имеются вVisualStudio?
2. Для чего применяется MicrosoftTestManager? Какие он имеет функциональные возможности?
3. Для чего используется LabManagement 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 развертываться?
7. Кем обслуживаться 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 оформлении 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(требует 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, A или B, проверять правильность работы ПО, или, если программа правильно работает в ситуации A, то, скорее всего, в ситуации B 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 критерий 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 нарочно 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**(integrationtesting)** 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 (требует составления 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 обозреватель 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.

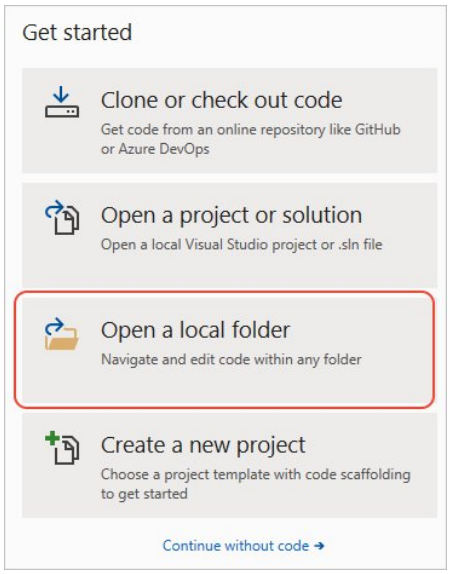
- Для платформы **pytest** параметры тестирования, такие как расположение тестов и шаблоны имен файлов, указываются с помощью стандартного INI-файла конфигурации pytest. Дополнительные сведения см. в справочной документации для pytest.

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

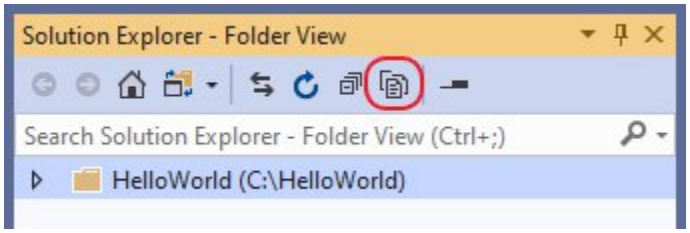
Настройка тестирования для Python без проекта

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** and **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

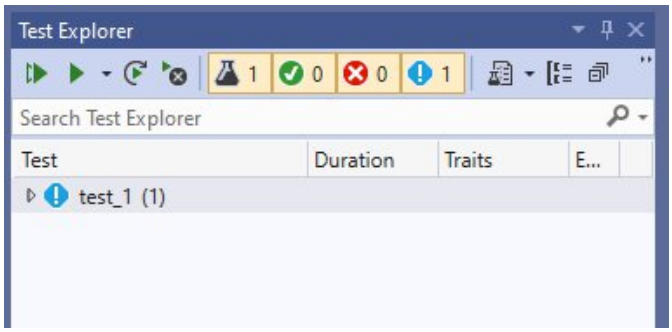
При использовании платформы pytest можно указать расположение тестов и шаблоны имен файлов с помощью стандартного INI-файла конфигурации pytest. По умолчанию используется папка рабочей области или проекта с шаблоном test_*py и *_test.py. Дополнительные сведения см. в справочной документации для pytest.

3. Выбрав платформу, снова щелкните проект правой кнопкой мыши и выберите **Добавить** > **Новый Элемент**, затем **Модульный тест Python** и нажмите кнопку **Добавить**.

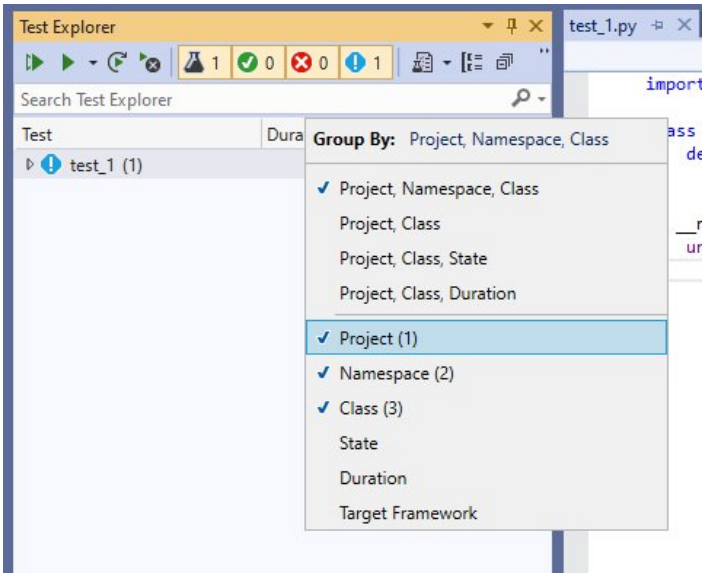
4. Это действие создает файл *test_1.py* с кодом, который импортирует стандартный модуль unittest, производный от тестового класса unittest.TestCase, и вызывает метод unittest.main() при запуске скрипта напрямую:

```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** > **Обозреватель 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).

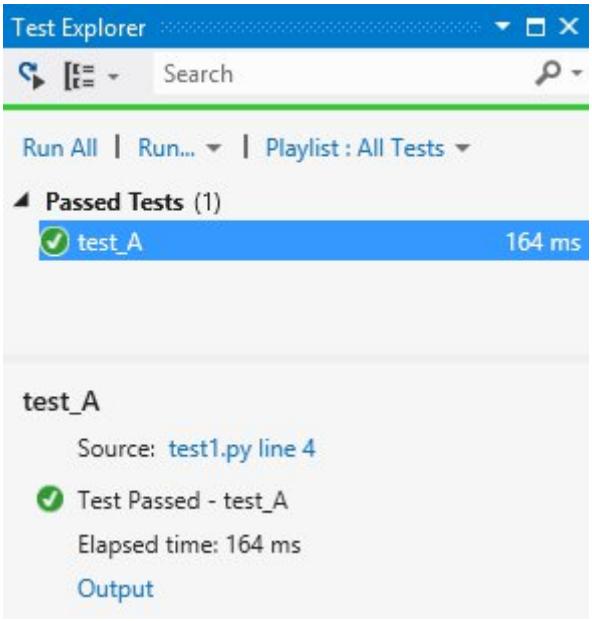
Выполнить тесты

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

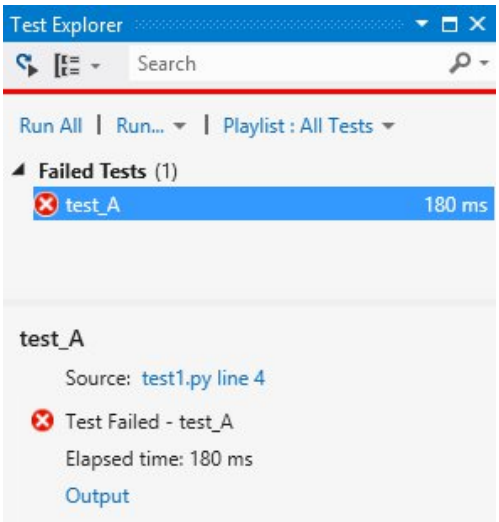
- Команда **Запустить все** явно выполняет все тесты из списка (с учетом фильтров).
- Команда меню **Запуск** позволяет одновременно выполнить все тесты, которые завершились успешно, завершились неудачно или еще не выполнялись.
- Также можно выбрать один или несколько тестов, щелкнуть их правой кнопкой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

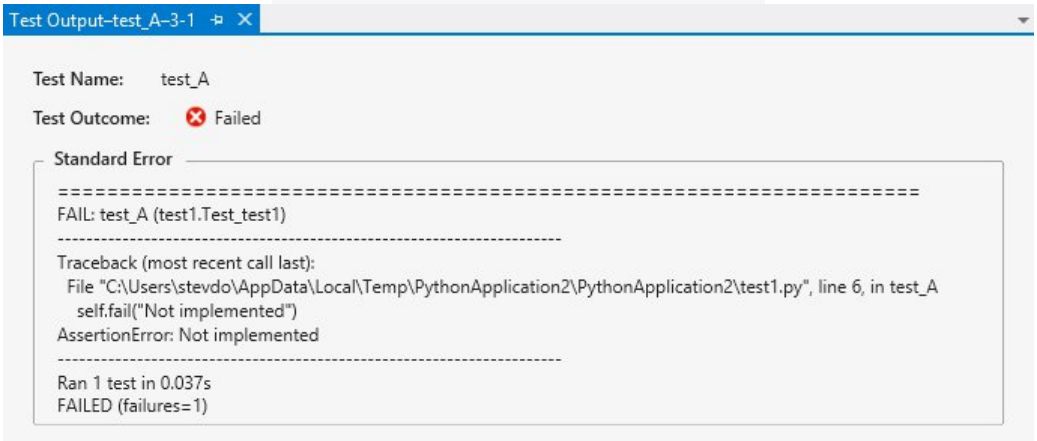
Тесты выполняются в фоновом режиме, а **обозреватель тестов** обновляет их состояние по мере завершения:

- 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:





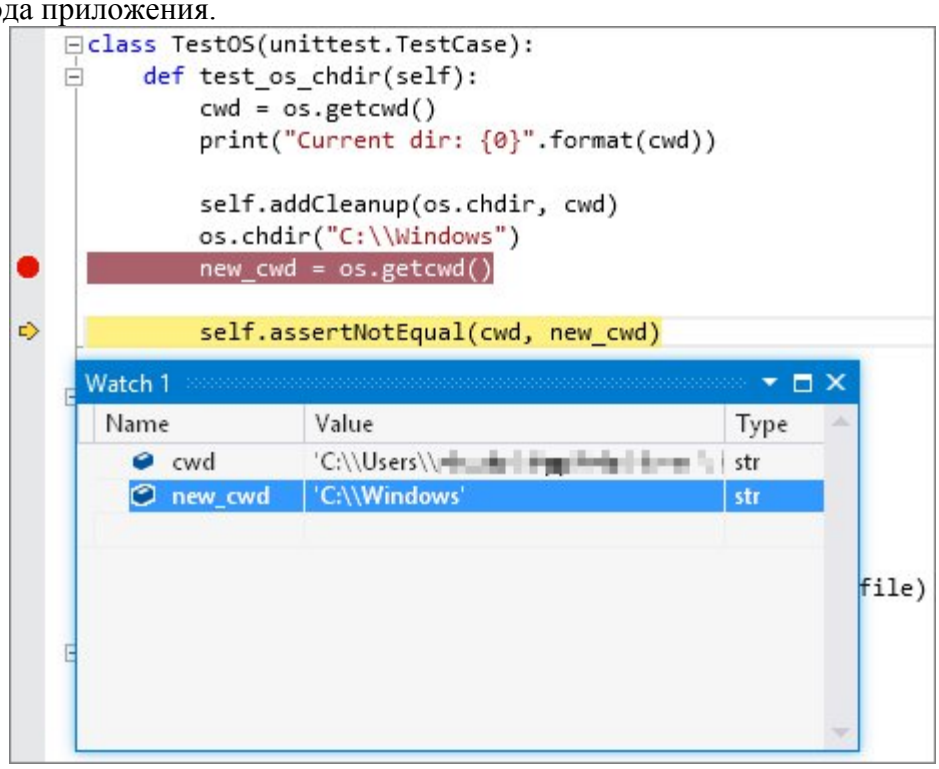
Отладка тестов

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**.

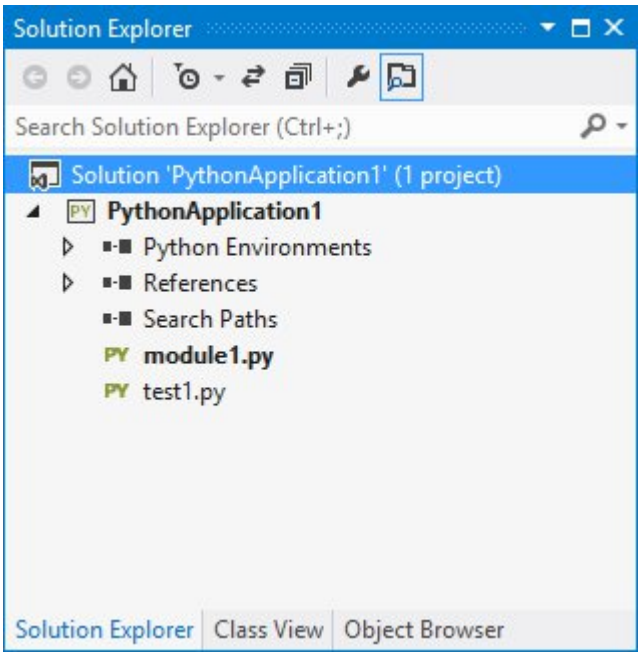
- оценка «*отлично*» выставляется студенту, если задание выполнено в полном объеме, студент хорошо ориентируется в технологии тестирования программного обеспечения, логично и корректно демонстрирует работу всех функций Visual Studio;
- оценка «*хорошо*» выставляется студенту, если он выполнил задание 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 (требует 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 IntellSense. 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.

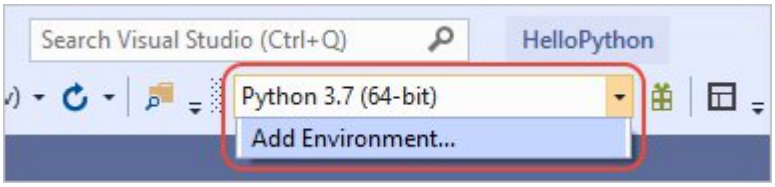
Каждый проект Python имеет один назначенный файл запуска, выделенный полужирным шрифтом в **обозревателе решений**. Это файл, который запускается, когда вы запускаете отладку (**F5** или **Отладка** > **Начать отладку**) или выполняете проект в **интерактивном** окне (**SHIFT+ALTF+5** или **Отладка** > **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

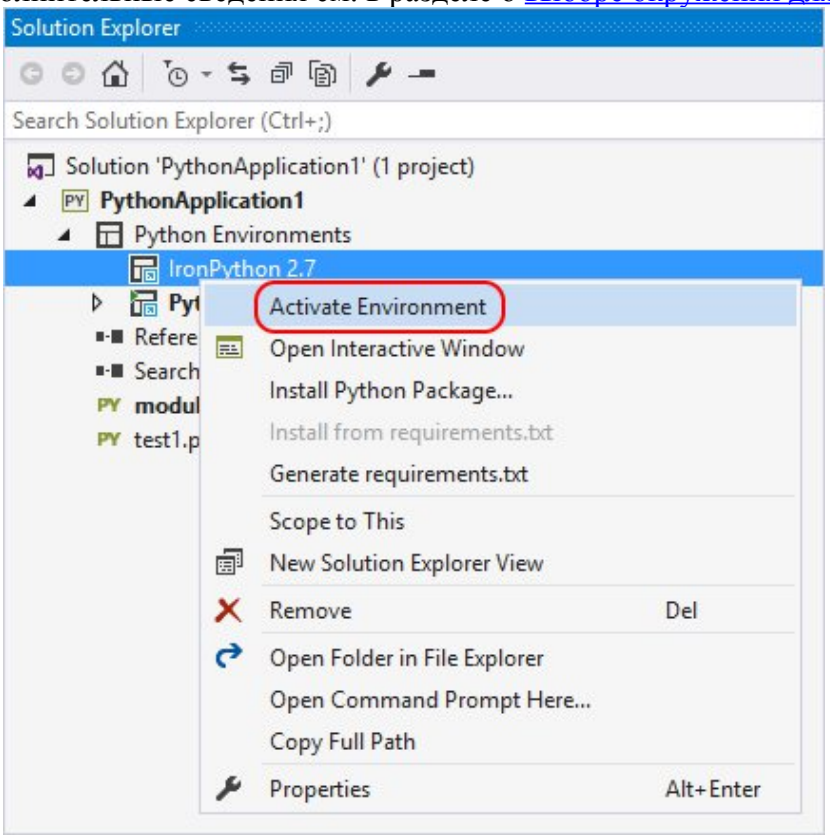
Если удалить выбранный файл запуска из проекта и не выбрать новый, Visual Studio не будет знать, с какого файла Python нужно начинать выполнение проекта. В этом случае в Visual Studio 2017 версии 15.6 и более поздних версий возникает ошибка. В более ранних версиях либо открывается окно вывода с запущенным 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.

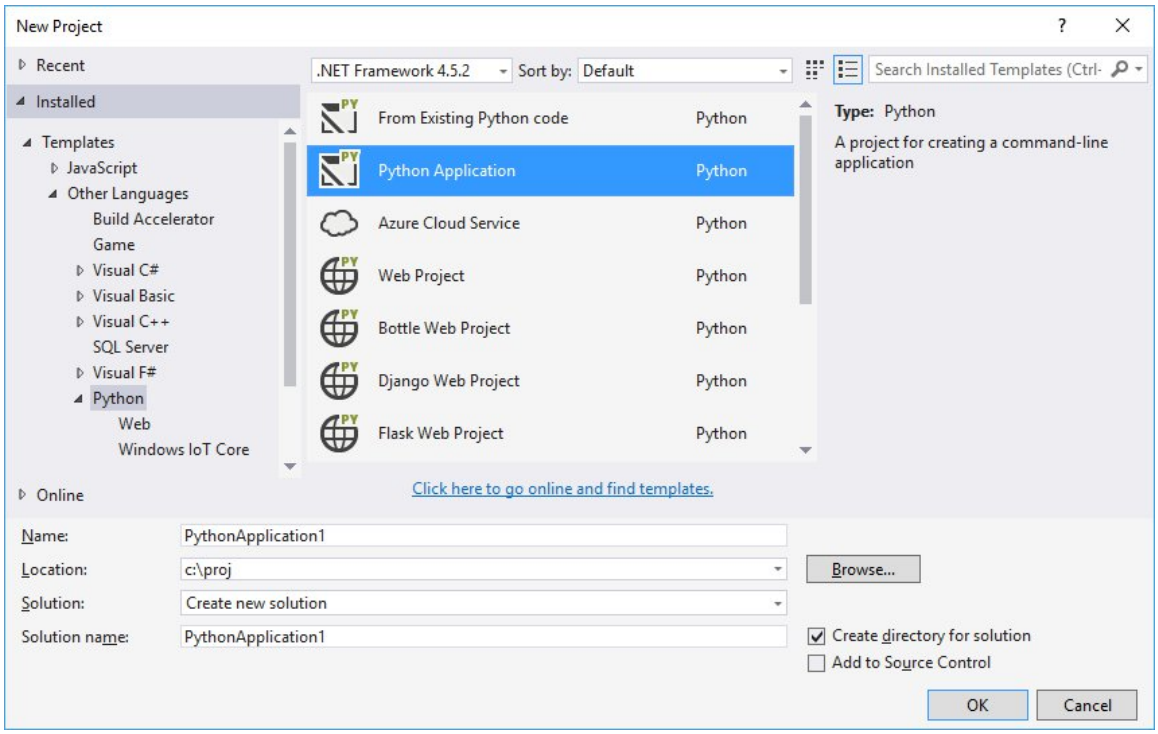


Чтобы изменить активное окружение, в **обозревателе решений** щелкните правой кнопкой мыши нужное окружение и выберите действие **Активировать окружение**. Дополнительные сведения см. в разделе о выборе окружения для проекта.



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 с IronPython	Аналогичен шаблону приложения Python, но по умолчанию использует интерпретатор IronPython, поддерживающий взаимодействие .NET и смешанный режим отладки с использованием языков .NET.
Приложение WPF с 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 с IronPython и 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 с 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 сведения

Шаблон	ОПИСАНИЕ
(Интернет вещей)	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

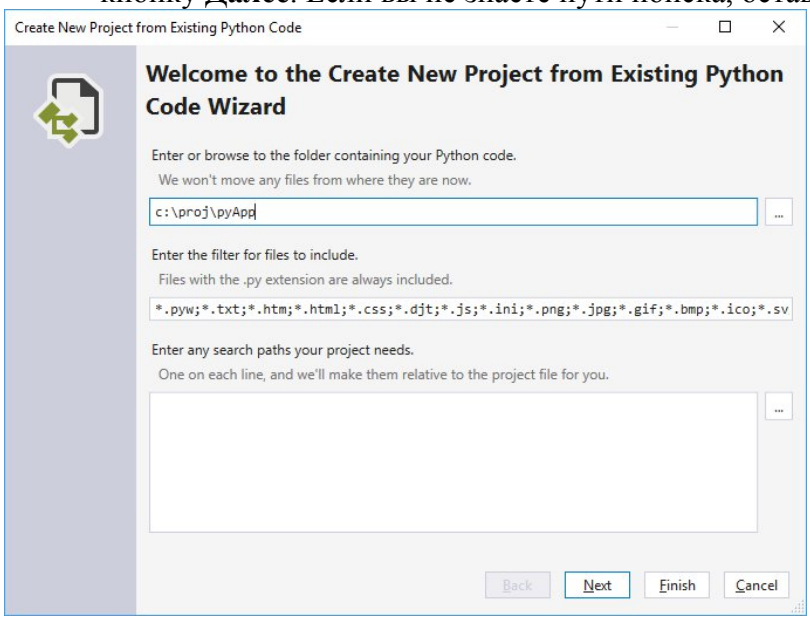
Так как Python является интерпретируемым языком, проекты Python в Visual Studio не создают отдельный исполняемый файл, как это делают проекты, написанные на других компилируемых языках программирования (например, C#). Дополнительные сведения см. в разделе **вопросов и ответов**.

Создание проекта на основе имеющихся файлов

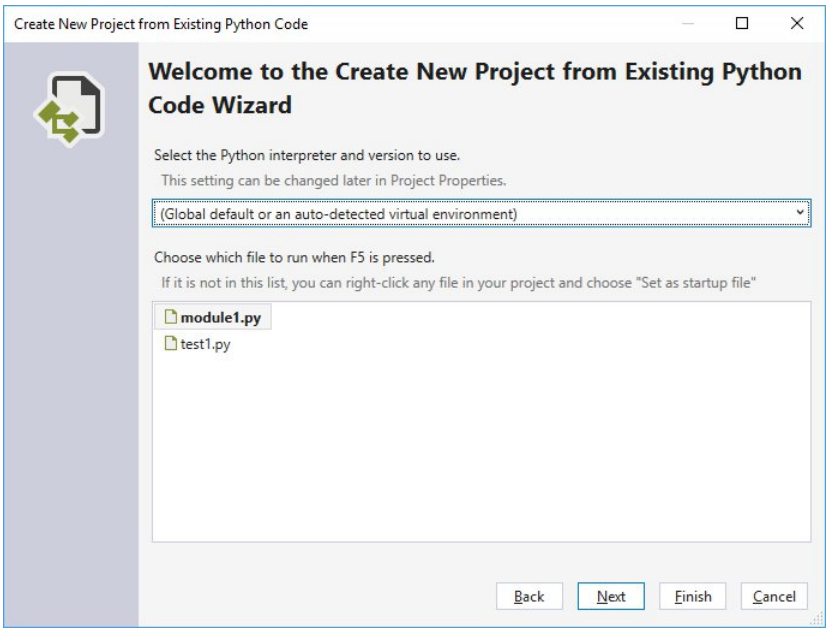
Важно!

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

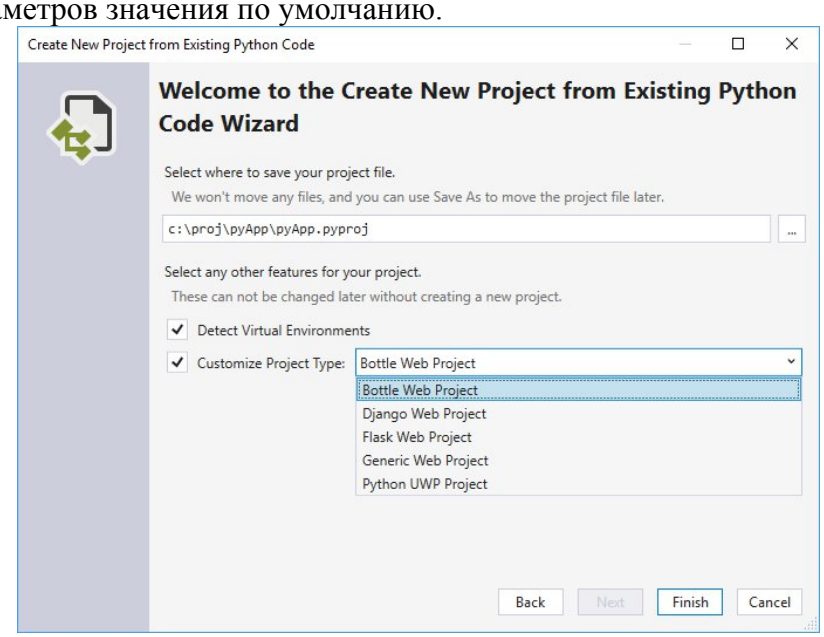
1. Запустите Visual Studio и последовательно выберите **Файл** > **Создать** > **Проект**.
2. В диалоговом окне **Создание проекта** выполните поиск по запросу «Python», выберите шаблон **На основе существующего кода Python**, укажите имя и расположение проекта, а затем нажмите кнопку **ОК**.
3. В появившемся 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.

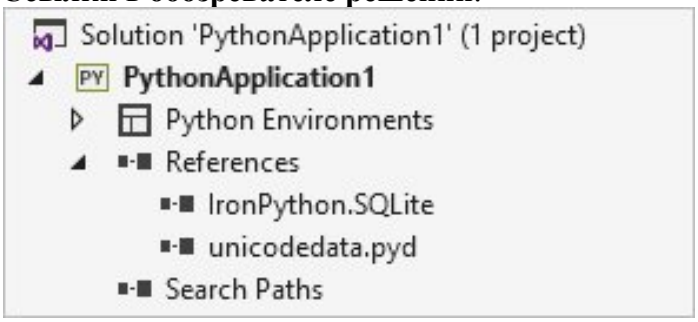
При попытке создать связь с файлом, который уже существует в папке проекта, он добавляется как обычный файл, а не как связанный. Чтобы преобразовать файл в связь, выберите **Файл** > **Сохранить как** и сохраните файл в расположение вне иерархии проекта. Visual Studio автоматически преобразует его в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

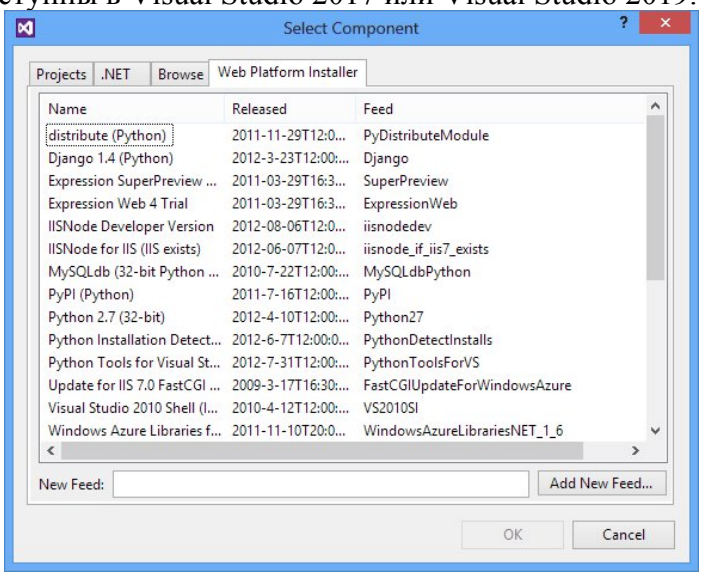
```python
import clr
clr.AddReference('System.Drawing')
clr.AddReference('System.Windows.Forms')

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 управлении 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 работа №2 (требует 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» 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, инструментарии, 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 методолгия, 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 разработки 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.

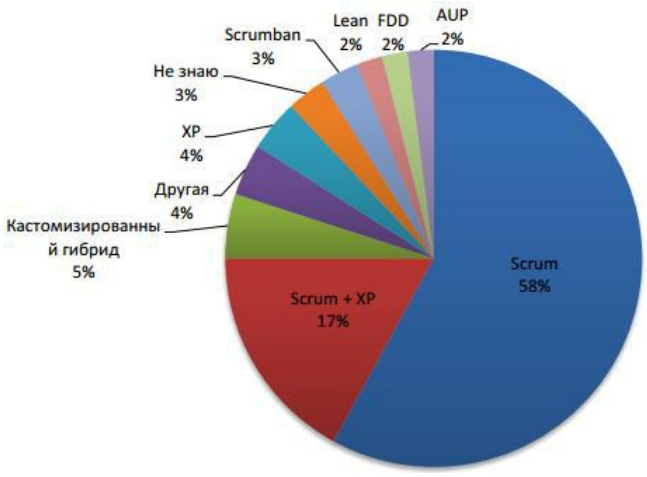


Рис. 1. Популярность гибких методологий

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 (CMM-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).

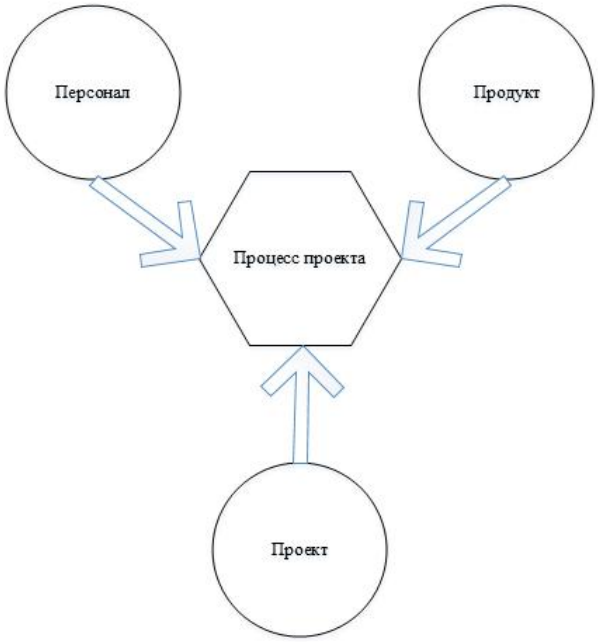


Рис. 2. Закон четырех П

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	
Имеютсяли программные средства реализации методологии, какие?	
Используется ли в настоящее время	
Примеры успешных проектов, реализованных спомощью 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 управлении 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 – 8 СЕМЕСТР

ПК-1 Способен выявлять информационные потребности пользователей и составлять техническое задание на разработку информационной системы
ПК-1.2. Составляет техническое задание на разработку информационной системы
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 жизненные 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 информационные системы
ПК-2.2. Разрабатывает модель информационной системы, в том числе с опорой на современные облачные решения

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?

ПК-2 Способен проектировать информационные системы
ПК-2.3. Составляет технико-экономическое обоснование проектных решений

Обучающийся знает: понятие технико-экономического обоснования и базовые характеристики затрат на разработку программных средств.

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

1. Организация документирования программных средств
2. Требования к документации программных средств
3. Планирование документирования программных средств
4. Состав и содержание документов программных средств
5. Каким способом программное обеспечение развертываться?
6. Кем обслуживаться программное обеспечение в процессе эксплуатации?

ПК-4 Способен управлять процессами создания информационных систем
ПК-4.1. Применяет гибкие технологии управления и цифровые средства контроля жизненного цикла создания информационных систем

Обучающийся знает: суть процесса разработки программного обеспечения и виды деятельности в нем, модели жизненного цикла разработки программного обеспечения; 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 процессами создания информационных систем
ПК-4.2. Применяет актуальные нормативные документы и стандарты при создании информационных систем, используя общедоступные справочно-правовые системы и профессиональные базы данных сферы ИТ

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 разработки информационных систем
ПК-5.1. Проводит оценку качества программной разработки на разных этапах жизненного цикла информационных систем с привлечением современных отраслевых решений ИТ-сферы

Обучающийся знает: международные и отечественные стандарты оценки качества разработки информационных систем, а также методы и способы тестирования программного обеспечения

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

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

ПК-5 Способен обеспечивать качество разработки информационных систем
ПК-5.2. Проводит оценку рисков работы информационных систем в условиях цифровизации бизнеса и применения сквозных цифровых и отраслевых технологий
Обучающийся знает: возможные риски функционирования информационных систем, методы и способы оценки рисков информационных систем
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.

ПК-5 Способен обеспечивать качество разработки информационных систем
ПК-5.3. Обеспечивает эффективную организацию разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества 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.
Оценка достижения обучающимся запланированного результата обучения осуществляется по результатам его участия в устных опросах и ответа на экзамене.

1. Какие шаблоны тестовых проектов имеются вVisualStudio?
2. Для чего применяется MicrosoftTestManager? Какие он имеет функциональные возможности?
3. Для чего используется LabManagement 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?

ОЦЕНКА УМЕНИЙ И НАВЫКОВ НА ЭКЗАМЕНЕ

ПК-1 Способен выявлять информационные потребности пользователей и составлять техническое задание на разработку информационной системы
ПК-1.2. Составляет техническое задание на разработку информационной системы
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 информационные системы
ПК-2.2. Разрабатывает модель информационной системы, в том числе с опорой на современные облачные решения

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

Обучающийся владеет: навыками критического анализа информации.

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

ПК-2 Способен проектировать информационные системы

ПК-2.3. Составляет технико-экономическое обоснование проектных решений

Обучающийся умеет: прогнозировать и оценивать объем ресурсов, необходимых для создания программных продуктов и комплексов, осуществлять выбор проектного решения с опорой на международные стандарты и методы оценки качества разработки программного обеспечения и современные требования к информационным системам.

Обучающийся владеет: навыками проведения технико-экономического обоснования проектного решения и проведения системного анализа прикладной области.

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

ПК-4 Способен управлять процессами создания информационных систем

ПК-4.1. Применяет гибкие технологии управления и цифровые средства контроля жизненного цикла создания информационных систем

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

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

Оценка достижения обучающимся запланированного результата обучения осуществляется выполнения практических работ.

ПК-4 Способен управлять процессами создания информационных систем

ПК-4.2. Применяет актуальные нормативные документы и стандарты при создании информационных систем, используя общедоступные справочно-правовые системы и профессиональные базы данных сферы ИТ

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 разработки информационных систем

ПК-5.1. Проводит оценку качества программной разработки на разных этапах жизненного цикла информационных систем с привлечением современных отраслевых решений ИТ-сферы

Обучающийся умеет: проводить тестирование информационных систем.

Обучающийся владеет: навыками построения плана и протокола тестирования.

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

ПК-5 Способен обеспечивать качество разработки информационных систем

ПК-5.2. Проводит оценку рисков работы информационных систем в условиях цифровизации бизнеса и применения сквозных цифровых и отраслевых технологий

Обучающийся умеет: анализировать издержки и риски при создании информационных систем; разрабатывать план упреждения рисков в процессе создания информационных систем.

Обучающийся владеет: навыками оценки рисков в различных моделях разработки информационных систем.

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

ПК-5 Способен обеспечивать качество разработки информационных систем

ПК-5.3. Обеспечивает эффективную организацию 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.

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 ОЦЕНИВАНИЯ ЗНАНИЙ, УМЕНИЙ, 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					

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
техническое задание на разработку информационной системы ПК-1.2. Составляет техническое задание на разработку информационной системы					
Знать: понятия, особенности и архитектуру программного обеспечения, профессиональные и этические требования профессионального сообщества программистов к деятельности по разработке информационных систем	Не знает	Фрагментарные знания о понятиях, особенностях и архитектуре программного обеспечения, профессиональных и этических требованиях профессионального сообщества программистов к деятельности по разработке информационных систем	Общие, но не структурированные знания о понятиях, особенностях и архитектуре программного обеспечения, профессиональных и этических требованиях профессионального сообщества программистов к деятельности по разработке информационных систем	Сформированные, но содержащие отдельные пробелы знания о понятиях, особенностях и архитектуре программного обеспечения, профессиональных и этических требованиях профессионального сообщества программистов к деятельности по разработке информационных систем	Сформированные систематизированные прочные знания о понятиях, особенностях и архитектуре программного обеспечения, профессиональных и этических требованиях профессионального сообщества 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 самостоятельностью выполнения умения составлять техническую документацию процесса разработки программного обеспечения	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения составлять техническую документацию процесса разработки программного обеспечения	Успешно сформированные, самостоятельно и осознанно выполняемые умения составлять техническую документацию процесса разработки программного обеспечения
Владеть: навыками разработки отдельных технических документов	Не владеет	Фрагментарно сформированные навыки разработки отдельных технических	В целом сформированные, но выполняемые на элементарном	Сформированные, но не устойчивые в сложных ситуациях навыки	Успешно сформированные, устойчивые, технологически и грамотно

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
		документов	уровне навыки разработки отдельных технических документов	разработки отдельных технических документов	выполняемые навыки разработки отдельных технических документов
ПК-2 Способен проектировать информационные системы ПК-2.2. Разрабатывает модель информационной системы, в том числе с опорой на современные облачные решения					
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	Не знает	Фрагментарные знания о процессе и средствах разработки 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	Общие, но не структурированные знания о процессе и средствах разработки 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	Сформированные, но содержащие отдельные пробелы знания о процессе и средствах разработки 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	Сформированные систематизированные прочные знания о процессе и средствах разработки 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	В целом освоенные, но не подкрепляемые знаниями и самостоятельностью выполнения умения определять факторы сложности разработки	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения определять факторы	Успешно сформированные, самостоятельно и осознанно выполняемые умения определять факторы сложности разработки программных систем,

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
		проектировани я	программных систем, выбирать адекватные технологии для их проектировани я	сложности разработки программных систем, выбирать адекватные технологии для их проектирован ия	выбирать адекватные технологии для их проектировани я
Владеть: навыками критического анализа технической информации	Не владеет	Фрагментарно сформированн ые навыки критического анализа технической информации	В целом сформированн ые, но выполняемые на элементарном уровне навыки критического анализа технической информации	Сформирован ные, но не устойчивые в сложных ситуациях навыки критического анализа технической информации	Успешно сформированн ые, устойчивые, технологическ и грамотно выполняемые навыки критического анализа технической информации
ПК-2 Способен проектировать информационные системы П К-2.3. Составляет технико-экономическое обоснование проектных решений					
Знать: - понятие технико-экономическ ого обоснования, - базовые характеристи ки затрат на разработку программных средств	Не знает	Фрагментарны е знания о понятиях технико-экономическог о обоснования и Базовых характеристика х затрат на разработку программных средств	Общие, но не структурирова нные знания о понятиях технико-экономическог о обоснования и Базовых характеристика х затрат на разработку программных средств	Сформирован ные, но содержащие отдельные пробелы знания о понятиях технико-экономическо го обоснования и Базовых характеристи ках затрат на разработку программных средств	Сформированн ые систематизиро ванные прочные знания о понятиях технико-экономическог о обоснования и Базовых характеристик ах затрат на разработку программных средств
Уметь: - прогнозирова ть и оценивать объем ресурсов, необходимых для создания программных	Не умеет	Частично освоенные умения прогнозировать и оценивать объем ресурсов, необходимых для создания	В целом освоенные, но не подкрепляемые знаниями и самостоятельно стью выполнения умения	В целом сформирован ные и самостоятель но выполняемые , но не всегда интеллектуал ьно	Успешно сформированн ые, самостоятельн о и осознанно выполняемые умения прогнозироват ь и оценивать

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
продуктов и комплексов - осуществлять выбор проектного решения с опорой на международные стандарты и методы оценки качества разработки программного обеспечения и современные требования к информационным системам		программных продуктов и комплексов, осуществлять выбор проектного решения с опорой на международные стандарты и методы оценки качества разработки программного обеспечения и современные требования к информационным системам	прогнозировать и оценивать объем ресурсов, необходимых для создания программных продуктов и комплексов, осуществлять выбор проектного решения с опорой на международные стандарты и методы оценки качества разработки программного обеспечения и современные требования к информационным системам	обоснованные умения прогнозировать и оценивать объем ресурсов, необходимых для создания программных продуктов и комплексов, осуществлять выбор проектного решения с опорой на международные стандарты и методы оценки качества разработки программного обеспечения и современные требования к информационным системам	объем ресурсов, необходимых для создания программных продуктов и комплексов, осуществлять выбор проектного решения с опорой на международные стандарты и методы оценки качества разработки программного обеспечения и современные требования к информационным системам
Владеть: - навыками проведения технико-экономического обоснования проектного решения; - навыками проведения системного анализа прикладной области	Не владеет	Фрагментарно сформированные навыки проведения технико-экономического обоснования проектного решения и навыки проведения системного анализа прикладной области	В целом сформированные, но выполняемые на элементарном уровне навыки проведения технико-экономического обоснования проектного решения и навыки проведения системного анализа прикладной области	Сформированные, но не устойчивые в сложных ситуациях навыки проведения технико-экономического обоснования проектного решения и навыки проведения системного анализа прикладной области	Успешно сформированные, устойчивые, технологически и грамотно выполняемые навыки проведения технико-экономического обоснования проектного решения и навыки проведения системного анализа прикладной области
ПК-4 Способен управлять процессами создания информационных систем					

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
ПК-4.1. Применяет гибкие технологии управления и цифровые средства контроля жизненного цикла создания информационных систем					
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
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 самостоятельностью выполнения умения планировать этапы разработки информационной системы с	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения планировать этапы разработки	Успешно сформированные, самостоятельно и осознанно выполняемые умения планировать этапы разработки информационной системы с использованием

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
			использование м инструменталь ных средств	информацион ной системы с использовани ем инструментал ьных средств	инструменталь ных средств
Владеть: - навыками управления программны м проектом с помощью программно-технических средств, - навыками работы с сервисами контроля версий программног о продукта	Не владеет	Фрагментарно сформированн ые навыки управления программным проектом с помощью программно-технических средств и навыки работы с сервисами контроля версий программного продукта	В целом сформированн ые, но выполняемые на элементарном уровне навыки управления программным проектом с помощью программно-технических средств и навыки работы с сервисами контроля версий программного продукта	Сформирован ные, 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 сформированн ые, устойчивые, технологическ 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 процессами создания информационных систем ПК-4.2. Применяет актуальные нормативные документы и стандарты при создании информационных систем, используя общедоступные справочно-правовые системы и профессиональные базы данных сферы ИТ					
Знать: нормативные документы и стандарты в процессе создания информацио нных систем	Не знает	Фрагментарны е знания о нормативных документах и стандартах в процессе создания информационн ых систем	Общие, но не структурирова нные знания о нормативных документах и стандартах в процессе создания информационн ых систем	Сформирован ные, но содержащие отдельные пробелы знания о нормативных документах и стандартах в процессе создания информацион ных систем	Сформированн ые систематизиро ванные прочные знания о нормативных документах и стандартах в процессе создания информационн ых систем
Уметь: применять нормативные документы и стандарты в процессе создания	Не умеет	Частично освоенные умения применять нормативные документы и стандарты в	В целом освоенные, но не подкрепляемые знаниями и самостоятельно стью	В целом сформирован ные и самостоятель но выполняемые , но не всегда	Успешно сформированн ые, самостоятельн о и осознанно выполняемые умения

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
информационных систем		процессе создания информационных систем	выполнения умения применять нормативные документы и стандарты в процессе создания информационных систем	интеллектуально обоснованные умения применять нормативные документы и стандарты в процессе создания информационных систем	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 разработки информационных систем ПК-5.1. Проводит оценку качества программной разработки на разных этапах жизненного цикла информационных систем с привлечением современных отраслевых решений ИТ-сферы					
Знать: - международные и отечественные стандарты оценки качества разработки информационных систем; - методы и способы тестирования программного обеспечения	Не знает	Фрагментарные знания международных и отечественных стандартов оценки качества разработки информационных систем; методы и способы тестирования программного обеспечения	Общие, но не структурированные знания международных и отечественных стандартов оценки качества разработки информационных систем; методы и способы тестирования программного обеспечения	Сформированные, но содержащие отдельные пробелы знания международных и отечественных стандартов оценки качества разработки информационных систем; методы и способы тестирования программного обеспечения	Сформированные систематизированные прочные знания международных и отечественных стандартов оценки качества разработки информационных систем; методы и способы тестирования программного обеспечения

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
Уметь: - проводить тестирование информационных систем	Не умеет	Частично освоенные умения проводить тестирование информационных систем	В целом освоенные, но не подкрепляемые знаниями и самостоятельностью выполнения умения проводить тестирование информационных систем	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения проводить тестирование информационных систем	Успешно сформированные, самостоятельно и осознанно выполняемые умения ан проводить тестирование информационных систем
Владеть: - навыками построения плана и протокола тестирования	Не владеет	Фрагментарно сформированные навыки навыками построения плана и протокола тестирования	В целом сформированные, но выполняемые на элементарном уровне навыки навыками построения плана и протокола тестирования	Сформированные, но не устойчивые в сложных ситуациях навыки навыками построения плана и протокола тестирования	Успешно сформированные, устойчивые, технологически и грамотно выполняемые навыки навыками построения плана и протокола тестирования
ПК-5 Способен обеспечивать качество разработки информационных систем ПК-5.2. Проводит оценку рисков работы информационных систем в условиях цифровизации бизнеса и применения сквозных цифровых и отраслевых технологий					
Знать: - возможные риски функционирования информационных систем - методы и способы оценки рисков информационных систем	Не знает	Фрагментарные знания о возможных рисках функционирования информационных систем, методах и способах оценки рисков информационных систем	Общие, но не структурированные знания о возможных рисках функционирования информационных систем, методах и способах оценки рисков информационных систем	Сформированные, но содержащие отдельные пробелы знания о возможных рисках функционирования информационных систем, методах и способах оценки рисков информационных систем	Сформированные систематизированные прочные знания о возможных рисках функционирования информационных систем, методах и способах оценки рисков информационных систем

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
Уметь: - анализировать издержки и риски при создании информационных систем; - разрабатывать план упреждения рисков в процессе создания информационных систем	Не умеет	Частично освоенные умения анализировать издержки и риски при создании информационных систем, разрабатывать план упреждения рисков в процессе создания информационных систем	В целом освоенные, но не подкрепляемые знаниями и самостоятельностью выполнения умения анализировать издержки и риски при создании информационных систем, разрабатывать план упреждения рисков в процессе создания информационных систем	В целом сформированные и самостоятельно выполняемые, но не всегда интеллектуально обоснованные умения анализировать издержки и риски при создании информационных систем, разрабатывать план упреждения рисков в процессе создания информационных систем	Успешно сформированные, самостоятельно и осознанно выполняемые умения анализировать издержки и риски при создании информационных систем, разрабатывать план упреждения рисков в процессе создания информационных систем
Владеть: навыками оценки рисков в различных моделях информационных систем	Не владеет	Фрагментарно сформированные навыки оценки рисков в различных моделях информационных систем	В целом сформированные, но выполняемые на элементарном уровне навыки оценки рисков в различных моделях информационных систем	Сформированные, но не устойчивые в сложных ситуациях навыки оценки рисков в различных моделях информационных систем	Успешно сформированные, устойчивые, технологически и грамотно выполняемые навыки оценки рисков в различных моделях информационных систем
ПК-5 Способен обеспечивать качество разработки информационных систем ПК-5.3. Обеспечивает эффективную организацию разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта					
Знать: основы эффективной организации разработки информационных систем и интеграцию	Не знает	Фрагментарные знания об основах эффективной организации разработки информационных систем и	Общие, но не структурированные знания об основах эффективной организации разработки информационн	Сформированные, но содержащие отдельные пробелы знания об основах эффективной	Сформированные систематизированные прочные знания об основах эффективной

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта		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	организации разработки информационных систем и интеграцию всех технологических процессов (в том числе на основе облачных решений) для высокого качества программного продукта	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 самостоятельностью выполнения 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 и 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 работоспособн	В целом сформированные, но выполняемые на	Сформированные, но не устойчивые в сложных ситуациях	Успешно сформированные, устойчивые, технологическ

Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Критерии оценивания результатов обучения				
	1	2	3	4	5
информационных систем и эффективность интеграции всех технологических процессов		ости информационных систем и эффективность интеграции всех технологических процессов	элементарном уровне навыки проверки работоспособности информационных систем и эффективность интеграции всех технологических процессов	навыки проверки работоспособности информационных систем и эффективность интеграции всех технологических процессов	и грамотно выполняемые навыки проверки работоспособности информационных систем и эффективность интеграции всех технологических 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, особенностей 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, особенностей 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, особенностей и средств разработки информационных систем и крупных программных комплексов.

5. 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)