МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное учреждение
высшего профессионального образования
"Казанский (Приволжский) федеральный университет"
Институт вычислительной математики и информационных технологий

УТВЕРЖДАЮ

Проректор по образовательной деятельности КФУ

Проф. Таюрский Д.А.

_____ "____" _____ 20___ г.

подписано электронно-цифровой подписью

Программа дисциплины

Технологии и стандарты разработки программного обеспечения Б1.В.ДВ.6

Направление подготовки: 02.03.02 - Фундаментальная информатика и информационные технологии
Профиль подготовки: Системный анализ и информационные технологии
Квалификация выпускника: бакалавр
Форма обучения: очное
Язык обучения: русский

Автор(ы):

Тагиров Р.Р. , Шаймухаметов Р.Р.

Рецензент(ы):

Хабибуллин Р.Ф.

СОГЛАСОВАНО:

Заведующий(ая) кафедрой: Латыпов Р. Х.
Протокол заседания кафедры No ___ от "____" _____ 201__г

Учебно-методическая комиссия Института вычислительной математики и информационных технологий:
Протокол заседания УМК No ____ от "____" _____ 201__г

Регистрационный No 922417

Казань
2017

Содержание

Программу дисциплины разработал(а)(и) старший преподаватель, б/с Тагиров Р.Р. кафедра системного анализа и информационных технологий отделение фундаментальной информатики и информационных технологий , Ravil.Tagirov@kpfu.ru ; Начальник отдела Шаймухаметов Р.Р. Отдел метрологии, сертификации и стандартизации КФУ , Ramil.Shaimukhametov@kpfu.ru

1. Цели освоения дисциплины

Целью изучения дисциплины "Технологии и стандарты разработки программ" является формирование у студентов специальности 01.02.00 Прикладная информатика и информатика фундаментальных теоретических знаний по вопросам методики и практики проектирования сложных программных средств для информационных систем, а также обучение студентов современным программным средствам для проектирования программного обеспечения, основанным на использовании CASE-технологии.

2. Место дисциплины в структуре основной образовательной программы высшего профессионального образования

Данная учебная дисциплина включена в раздел " Б1.В.ДВ.6 Дисциплины (модули)" основной образовательной программы 02.03.02 Фундаментальная информатика и информационные технологии и относится к дисциплинам по выбору. Осваивается на 4 курсе, 7 семестр.

'Технологии и стандарты разработки программного обеспечения ' входит в состав профессиональных дисциплин. Является систематизирующим курсом, позволяющим охватить в рамках единой парадигмы весь процесс разработки программного обеспечения. Читается на 4 курсе, в 8 семестре.

3. Компетенции обучающегося, формируемые в результате освоения дисциплины /модуля

В результате освоения дисциплины формируются следующие компетенции:

Шифр компетенции	Расшифровка приобретаемой компетенции
ОПК-4 (профессиональные компетенции)	способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности
ПК-10 (профессиональные компетенции)	способность реализовывать процессы управления качеством производственной деятельности, связанной с созданием и использованием информационных технологий, осуществлять мониторинг и оценку качества процессов производственной деятельности
ПК-11 (профессиональные компетенции)	способность составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы
ПК-8 (профессиональные компетенции)	способность применять на практике международные и профессиональные стандарты информационных технологий, современные парадигмы и методологии, инструментальные и вычислительные средства
ПК-9 (профессиональные компетенции)	способностью составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы

В результате освоения дисциплины студент:

1. должен знать:

содержание действующих российских и международных стандартов в области создания программных средств,содержание действующих российских стандартов документирования программных средств, современное состояние развития CASE-средств и промышленных технологий проектирования ПО, современные методы проектирования ПО, принципы организации и методики тестирования при испытании сложных ПС и определения их надежности, методы управления разработкой сложных программных систем

2. должен уметь:

составлять модель жизненного цикла для проектирования ПС,
формировать цели, задачи и требования к проектируемому ПС,
на основе построения и анализа моделей AS-IS и TO-BE,
строить семейство моделей проектируемой программной системы на основе выбранного метода проектирования,
применять инструментальные CASE-средства для разработки программного обеспечения,
выбирать и применять методы тестирования ПС,
составлять документацию, сопровождающую проектирование ПС на всех его этапах

3. должен владеть:

-навыками создания документации по программному проекту
-навыками разработки ПО различного уровня сложности

4. должен демонстрировать способность и готовность:

-применять полученные знания и навыки в своей будущей профессиональной деятельности

4. Структура и содержание дисциплины/ модуля

Общая трудоемкость дисциплины составляет 2 зачетных(ые) единиц(ы) 72 часа(ов).

Форма промежуточного контроля дисциплины зачет в 7 семестре.

Суммарно по дисциплине можно получить 100 баллов, из них текущая работа оценивается в 50 баллов, итоговая форма контроля - в 50 баллов. Минимальное количество для допуска к зачету 28 баллов.

86 баллов и более - "отлично" (отл.);

71-85 баллов - "хорошо" (хор.);

55-70 баллов - "удовлетворительно" (удов.);

54 балла и менее - "неудовлетворительно" (неуд.).

4.1 Структура и содержание аудиторной работы по дисциплине/ модулю

Тематический план дисциплины/модуля

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
1.	Тема 1. Программное обеспечение ЭВМ.	7		0	0	2	Письменное домашнее задание

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
2.	Тема 2. Пакеты прикладных программ.	7		0	0	2	Письменное домашнее задание
3.	Тема 3. Программные средства.	7		0	0	2	Письменное домашнее задание
4.	Тема 4. Жизненный цикл программного обеспечения.	7		0	0	2	Письменное домашнее задание
5.	Тема 5. Модели жизненного цикла программного обеспечения.	7		0	0	2	Письменное домашнее задание
6.	Тема 6. Разработка требований и внешнее проектирование ПО.	7		0	0	2	Письменное домашнее задание
7.	Тема 7. Структурный подход к проектированию программного обеспечения.	7		0	0	2	Письменное домашнее задание
8.	Тема 8. Проектирование и программирование модулей.	7		0	0	2	Письменное домашнее задание
9.	Тема 9. Объектно-ориентированный подход к проектированию программного обеспечения.	7		0	0	2	Контрольная работа Письменное домашнее задание
10.	Тема 10. Проектирование и разработка интерфейса ПО.	7		0	0	2	Письменное домашнее задание
11.	Тема 11. Тестирование, отладка и сборка ПО.	7		0	0	2	Письменное домашнее задание
12.	Тема 12. Сопровождение ПО на стадии эксплуатации.	7		0	0	2	Письменное домашнее задание

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
13.	Тема 13. Управление разработкой ПО.	7		0	0	4	Письменное домашнее задание
14.	Тема 14. Документация ПО.	7		0	0	4	Письменное домашнее задание
15.	Тема 15. Разработка и стандартизация информационных технологий.	7		0	0	4	Контрольная работа Письменное домашнее задание
.	Тема . Итоговая форма контроля	7		0	0	0	Зачет
	Итого			0	0	36	

4.2 Содержание дисциплины

Тема 1. Программное обеспечение ЭВМ.

лабораторная работа (2 часа(ов)):

Технология программирования (ТП) - технология разработки программного средства (ПС), включающая все процессы, начиная с момента зарождения идеи этого средства. Результатом применения ТП является программа, действующая в заданной вычислительной среде, хорошо отлаженная и документированная, доступная для понимания и развития в процессе сопровождения. Процесс разработки ПС и методы оценивания продуктов стандартизованы (ISO/IEC 12207, 9126 и др.). Все это способствует повышению эффективности проектирования, разработки, тестирования и оценки качества ПС. Архитектура ПС - это представление ПС как системы, состоящей из совокупности взаимодействующих подсистем. В качестве таких подсистем выступают отдельные программы. Разработка архитектуры является первым этапом упрощения создаваемого ПС путем выделения независимых компонент. Основные задачи разработки архитектуры ПС [1]: 1. выделение программных подсистем и отображение на них внешних функций (заданных во внешнем описании) ПС; 2. определение способов взаимодействия между выделенными программными подсистемами. С учетом принимаемых на этом этапе решений производится дальнейшая конкретизация функциональной спецификации. Проектирование ПС ? это процесс, следующий за этапами анализа и формирования требований. Модели анализа поставляют этапу проектирования исходные сведения для работы. Информационная модель описывает информацию, которую должно обрабатывать ПС. Функциональная модель определяет перечень функций обработки. Поведенческая модель закрепляет динамику системы. На выходе этапа проектирования ? разработка данных, разработка архитектуры и процедурная разработка ПС

Тема 2. Пакеты прикладных программ.

лабораторная работа (2 часа(ов)):

В качестве модульной структуры программы принято использовать древовидную структуру. В узлах такого дерева размещаются программные модули, а направленные дуги показывают подчиненность модулей. В процессе разработки программы ее модульная структура может формироваться и использоваться по-разному для определения порядка программирования и отладки модулей, указанных в этой структуре. Обычно рассматриваются два метода : метод восходящей разработки; метод нисходящей разработки. Метод восходящей разработки заключается в следующем. Сначала строится модульная структура программы в виде дерева. Затем поочередно программируются модули программы, начиная с модулей самого нижнего уровня, в таком порядке, чтобы для каждого программируемого модуля были уже запрограммированы все модули, к которым он может обращаться. После того, как все модули программы запрограммированы, производится их тестирование и отладка в порядке, в каком велось их программирование. Технологией программирования не рекомендуется восходящий порядок разработки программы по следующим причинам. Каждая программа подчиняется некоторым внутренним для нее, но глобальным для ее модулей соображениям. При восходящей разработке эта глобальная информация для модулей нижних уровней еще не ясна в полном объеме. Часто приходится перепрограммировать модули, когда уточняется эта информация. При восходящем тестировании для каждого модуля приходится создавать ведущую программу, которая подготавливает для тестируемого модуля необходимое состояние информационной среды и производит требуемое обращение к нему. Это приводит к большому объему отладочного программирования. Метод нисходящей разработки заключается в следующем. Как и в предыдущем методе сначала строится модульная структура программы в виде дерева. Затем поочередно программируются модули программы, начиная с верхнего уровня. Переход к программированию следующего модуля происходит в том случае, если запрограммирован модуль, который к нему обращается. После того, как все модули программы запрограммированы, производится их поочередное тестирование и отладка в таком же порядке. Первым тестируется головной модуль программы при том состоянии информационной среды, при котором начинает выполняться эта программа. Те модули, к которым может обращаться головной, заменяются их имитаторами. Имитатор модуля представляется программой, которая сигнализирует о факте обращения к имитируемому модулю. После завершения тестирования и отладки головного и любого последующего модуля производится переход к тестированию модулей, которые представлены имитаторами. Для этого имитатор заменяется самим этим модулем и добавляются имитаторы тех модулей, к которым он может обращаться. При этом каждый такой модуль будет тестироваться при тех состояниях информационной среды, которые возникают к моменту обращения к этому модулю при выполнении тестируемой программы. Таким образом, большой объем отладочного программирования при восходящем тестировании заменяется программированием простых имитаторов. Особенностью классических методов восходящей и нисходящей разработок является требование, чтобы модульная структура программы была разработана до начала программирования модулей. Это требование соответствует водопадному подходу к разработке ПС. Однако, не всегда до программирования модулей можно точно и содержательно разработать структуру программы. Конструктивный и архитектурный подходы к разработке программ предлагают формирование модульной структуры в процессе программирования модулей. Конструктивный подход к разработке программы представляет собой модификацию нисходящей разработки, при которой модульная древовидная структура программы формируется в процессе программирования модулей. Разработка программы при конструктивном подходе начинается с программирования головного модуля исходя из спецификации программы в целом. Если эта программа большая, выделяются подзадачи.

Тема 3. Программные средства.

лабораторная работа (2 часа(ов)):

Программу для ее упрощения разрабатывают по частям, которые называются программными модулями. Такой метод разработки программ называют модульным программированием. Программный модуль ? фрагмент описания процесса, оформляемый как самостоятельный программный продукт, пригодный для использования в описаниях разных процессов. Программный модуль программируется, компилируется и отлаживается отдельно; может включаться в состав разных программ; является средством борьбы со сложностью программ; является средством борьбы с дублированием в программировании. Основными характеристиками программного модуля являются [1, 7]: размер; связность; сцепление с другими модулями; рутинность. Размер модуля измеряется числом содержащихся в нем операторов или строк. Модуль не должен быть слишком маленьким или слишком большим. Маленькие модули приводят к громоздкой модульной структуре программы. Большие модули неудобны для изучения и изменений, могут существенно увеличить суммарное время повторных трансляций программы при ее отладке. Отладка модуля размером в одну страницу может быть в разы проще отладки модуля размером в одну страницу и еще 4-5 строк на другой странице. Это связано с принципами организации человеческой памяти. Есть сверхоперативная память, связанная, в основном, со зрением. Эта память имеет очень быстрый доступ, но очень мала ? 7-9 позиций. Существенно больше оперативная память, в которой и происходит вся основная мыслительная деятельность, но данные в ней не могут храниться долго. Наконец, самая большая ? долговременная память. Человеку непросто заложить туда данные, но хранятся они долго. С устройством памяти связан принцип центрального зрения. Человек хорошо воспринимает какую-то точку и то, что ее окружает. Если при отладке программы автор должен обозревать больше, чем одну небольшую страницу текста, он не может полноценно воспринять программу - листать вредно. Связность модуля - мера зависимости его частей, внутренняя характеристика. Чем выше связность модуля, тем больше связей он скрывает от внешней части программы и больший вклад в упрощение программы вносит. Для оценки степени связности модуля используется семь типов связности

Тема 4. Жизненный цикл программного обеспечения.

лабораторная работа (2 часа(ов)):

Под жизненным циклом программного средства (ЖЦПС) понимают весь период его разработки и эксплуатации, начиная от момента возникновения замысла ПС и кончая прекращением его использования. В настоящее время можно выделить пять основных подходов к организации процесса создания и использования ПС. Водопадный подход состоит из цепочки этапов. На каждом этапе создаются документы, используемые на последующем этапе. В исходном документе фиксируются требования к ПС. В конце этой цепочки создаются программы, включаемые в ПС. ? Исследовательское программирование предполагает быструю реализацию рабочих версий программ ПС, выполняющих в первом приближении требуемые функции. После экспериментального применения реализованных программ производится их модификация. Этот процесс повторяется до тех пор, пока ПС не будет достаточно приемлемо для пользователей. Такой подход применялся на ранних этапах развития программирования (интуитивная технология). В настоящее время этот подход применяется для разработки таких ПС, для которых пользователи не могут точно сформулировать требования (например, для разработки систем искусственного интеллекта). ? Прототипирование. Этот подход моделирует начальную фазу исследовательского программирования вплоть до создания рабочих версий программ, предназначенных для проведения экспериментов с целью установить требования к ПС. С самого начала разработчики пытаются выделить основные требования заказчика и реализовать их в виде работающего прототипа системы. Цикл разработки и показа прототипа повторяется несколько раз.

Тема 5. Модели жизненного цикла программного обеспечения.

лабораторная работа (2 часа(ов)):

Обобщением модели создания прототипов является спиральная модель, в которой разработка приложения выглядит как серия последовательных итераций. При большом числе итераций разработка по этой модели нуждается в автоматизации всех процессов, иначе она становится неэффективной. Формальные преобразования. Этот подход включает разработку формальных спецификаций ПС и превращение их в программы путем корректных преобразований. На этом подходе базируется компьютерная технология (CASE-технология) разработки ПС. Сборочное программирование. Этот подход предполагает, что ПС конструируется из компонент, которые уже существуют. Должна быть библиотека таких компонент, каждая из которых может многократно использоваться в разных ПС. Процесс разработки ПС при данном подходе состоит скорее из сборки программ из компонент, чем из их программирования Основное назначение моделей ЖЦ ПС Планирование и распределение работ между разработчиками, управление проектом. Обеспечение взаимодействия между разработчиками проекта и заказчиком. Контроль работ, оценивание промежуточных результатов заданным требованиям. Согласование промежуточных результатов с заказчиком. Проверка правильности конечного продукта путем его тестирования на запланированных и согласованных с заказчиком наборах тестов. Оценивание соответствия характеристик качества полученного продукта заданным требованиям. Определение направлений усовершенствования или модернизации продукта. На сегодня основой формирования новой модели ЖЦ для конкретной прикладной системы является международный стандарт ISO/IEC 12207 ?Информационная технология. Процессы жизненного цикла программных средств?, который задает полный набор процессов, охватывающий все возможные виды работ и задач, связанных с построением ПС, начиная с анализа предметной области и кончая изготовлением соответствующего продукта. Данный стандарт содержит основные и вспомогательные процессы

Тема 6. Разработка требований и внешнее проектирование ПО.

лабораторная работа (2 часа(ов)):

Под моделью ПО в общем случае понимается формализованное описание системы ПО на определенном уровне абстракции. Каждая модель определяет конкретный аспект системы, использует набор диаграмм и документов заданного формата, а также отражает точку зрения и является объектом деятельности различных людей с конкретными интересами, ролями или задачами. Графические (визуальные) модели представляют собой средства для визуализации, описания, проектирования и документирования архитектуры системы. Разработка модели системы ПО промышленного характера в такой же мере необходима, как и наличие проекта при строительстве большого здания. Это утверждение справедливо как в случае разработки новой системы, так и при адаптации типовых продуктов класса R/3 или BAAN, в составе которых также имеются собственные средства моделирования. Хорошие модели являются основой взаимодействия участников проекта и гарантируют корректность архитектуры. Поскольку сложность систем повышается, важно располагать хорошими методами моделирования. Хотя имеется много других факторов, от которых зависит успех проекта, но наличие строгого стандарта языка моделирования является весьма существенным. Состав моделей, используемых в каждом конкретном проекте, и степень их детальности в общем случае зависят от следующих факторов: сложности проектируемой системы; необходимой полноты ее описания; знаний и навыков участников проекта; времени, отведенного на проектирование. Визуальное моделирование оказало большое влияние на развитие ТС ПО вообще и CASE-средств в частности. Понятие CASE (Computer Aided Software Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение этого понятия, ограниченное только задачами автоматизации разработки ПО, в настоящее время приобрело новый смысл, охватывающий большинство процессов жизненного цикла ПО. CASE-технология представляет собой совокупность методов проектирования ПО, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех стадиях разработки и сопровождения ПО и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методах структурного или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

Тема 7. Структурный подход к проектированию программного обеспечения.

лабораторная работа (2 часа(ов)):

Структурный подход к проектированию программного обеспечения. Характеристика и основные принципы структурного подхода. SADT (Structured Analysis and Design Technique), DFD (Data Flow Diagrams) и ERD (Entity-Relationship Diagrams) модели структурного подхода. Концепции функциональной модели SADT. Состав функциональной модели. Построение иерархии диаграмм моделей стандарта IDEF0. Типы связей между функциями.

Тема 8. Проектирование и программирование модулей.

лабораторная работа (2 часа(ов)):

Восходящее проектирование (или проектирование ?снизу вверх?) основано на выделении нескольких достаточно крупных модулей, реализующих некоторые функции в общей программе. При выделении модулей опираются на доступность реализуемых функций для понимания, простоту структурирования данных, существование готовых программ и модулей для реализации заданных функций, возможности переделки существующих программ для новых целей; имеет значение и размер будущего модуля. Каждый модуль при восходящем проектировании автономно программируется, тестируется и отлаживается. После этого отдельные модули объединяются в подсистемы с помощью управляющего модуля, в котором определяется последовательность вызовов модулей, ввод-вывод и контроль данных и результатов. В свою очередь, подсистемы затем объединяются в более сложные системы и в общий программный комплекс, который подвергается комплексной отладке с проверкой правильности межмодульных связей. Рассмотренный подход можно рекомендовать при разработке не очень сложных программ. Если размеры подпрограмм невелики, то целесообразно выделить подпрограммы и начать программирование с их составления. Основные недостатки восходящего проектирования программы проявляются в сложности объединения модулей в единую систему, в трудности выявления и исправления ошибок, допущенных на ранних стадиях разработки модулей. Кроме того, отдельные модули могут создаваться без общего представления о структуре всей системы, что затрудняет их объединение. Для создания сложных программ можно рекомендовать нисходящее проектирование, основанное на выделении в решаемой задаче иерархии уровней обобщения. Схема иерархии уровней обобщения позволяет программисту сначала сконцентрировать внимание на том, что нужно сделать, и лишь затем ? на том, как это сделать. Ведущая программа записывается как программа верхнего уровня, управляющая вызовами модулей более низкого уровня. Каждый из модулей более низкого уровня, в свою очередь, управляет вызовами модулей еще более низкого уровня. Такой способ проектирования позволяет создавать сложные и громоздкие программы из небольших простых модулей: размер задачи отражается только в числе модулей и уровней обобщений. При нисходящем проектировании появляется возможность использовать вертикальное управление в схеме иерархии с использованием таких правил: модуль возвращает управление вызвавшему; модуль вызывает только модули более низкого уровня; принятие основных решений возлагается на модули максимально высокого уровня.

Тема 9. Объектно-ориентированный подход к проектированию программного обеспечения.

лабораторная работа (2 часа(ов)):

Концептуальной основой объектно-ориентированного анализа и проектирования ПО (ООАП) является объектная модель. Ее основные принципы (абстрагирование, инкапсуляция, модульность и иерархия) и понятия (объект, класс, атрибут, операция, интерфейс и др.) наиболее четко сформулированы Гради Бучем в его фундаментальной книге и последующих работах. Большинство современных методов ООАП основаны на использовании языка UML. Унифицированный язык моделирования UML (Unified Modeling Language) представляет собой язык для определения, представления, проектирования и документирования программных систем, организационно-экономических систем, технических систем и других систем различной природы. UML содержит стандартный набор диаграмм и нотаций самых разнообразных видов. UML - это преемник того поколения методов ООАП, которые появились в конце 1980-х и начале 1990-х годов. Создание UML фактически началось в конце 1994 г., когда Гради Буч и Джеймс Рамбо начали работу по объединению их методов Booch и OMT (Object Modeling Technique) под эгидой компании Rational Software. К концу 1995 г. они создали первую спецификацию объединенного метода, названного ими Unified Method, версия 0.8. Тогда же в 1995 г. к ним присоединился создатель метода OOSE (Object-Oriented Software Engineering) Ивар Якобсон. Таким образом, UML является прямым объединением и унификацией методов Буча, Рамбо и Якобсона, однако дополняет их новыми возможностями. Главными в разработке UML были следующие цели: предоставить пользователям готовый к использованию выразительный язык визуального моделирования, позволяющий им разрабатывать осмысленные модели и обмениваться ими; предусмотреть механизмы расширяемости и специализации для расширения базовых концепций; обеспечить независимость от конкретных языков программирования и процессов разработки. обеспечить формальную основу для понимания этого языка моделирования (язык должен быть одновременно точным и доступным для понимания, без лишнего формализма); стимулировать рост рынка объектно-ориентированных инструментальных средств; интегрировать лучший практический опыт.

Тема 10. Проектирование и разработка интерфейса ПО.

лабораторная работа (2 часа(ов)):

Интерфейсы являются основой взаимодействия всех современных информационных систем. Если интерфейс какого-либо объекта (персонального компьютера, программы, функции) не изменяется (стабилен, стандартизирован), это даёт возможность модифицировать сам объект, не перестраивая принципы его взаимодействия с другими объектами. Например, научившись работать с одной программой под Windows, пользователь с легкостью освоит и другие ? потому, что они имеют одинаковый интерфейс. В вычислительной системе взаимодействие может осуществляться на пользовательском, программном и аппаратном уровнях. В соответствии с этой классификацией можно выделить: Интерфейс пользователя ? совокупность средств, при помощи которых пользователь общается с различными устройствами. Интерфейс командной строки: инструкции компьютеру даются путём ввода с клавиатуры текстовых строк (команд). Графический интерфейс пользователя: программные функции представляются графическими элементами экрана. Диалоговый интерфейс Естественно-языковой интерфейс: пользователь ?разговаривает? с программой на родном ему языке. Физический интерфейс Способ взаимодействия физических устройств. Чаще всего речь идёт о компьютерных портах. Сетевой интерфейс Шлюз (телекоммуникации) ? устройство, соединяющее локальную сеть с более крупной, например, Интернетом Шина (компьютер) Нейро-компьютерный интерфейс (англ. brain-computer interface): отвечает за обмен между нейронами и электронным устройством при помощи специальных имплантированных электродов. Интерфейсы в программировании Интерфейс функции Интерфейс программирования приложений (API): набор стандартных библиотечных методов, который программист может использовать для доступа к функциональности другой программы. Вызов удалённых процедур COM-интерфейс Интерфейс (ООП)

Тема 11. Тестирование, отладка и сборка ПО.

лабораторная работа (2 часа(ов)):

Учитывая разнообразие источников ошибок, при составлении плана тестирования классифицируют ошибки на два типа: 1 ? синтаксические; 2 ? семантические (смысловые). Синтаксические ошибки ? это ошибки в записи конструкций языка программирования (чисел, переменных, функций, выражений, операторов, меток, подпрограмм). Семантические ошибки ? это ошибки, связанные с неправильным содержанием действий и использованием недопустимых значений величин. Обнаружение большинства синтаксических ошибок автоматизировано в основных системах программирования. Поиск же семантических ошибок гораздо менее формализован; часть их проявляется при исполнении программы в нарушениях процесса автоматических вычислений и индицируется либо выдачей диагностических сообщений рабочей программы, либо отсутствием печати результатов из-за бесконечного повторения одной и той же части программы (зацикливания), либо появлением непредусмотренной формы или содержания печати результатов. В план тестирования обычно входят следующие этапы: Сравнение программы со схемой алгоритма. Визуальный контроль программы на экране дисплея или визуальное изучение распечатки программы и сравнение ее с оригиналом на программном бланке. Первые два этапа тестирования способны устранить больше количество ошибок, как синтаксических (что не так важно), так и семантических (что очень важно, так как позволяет исключить их трудоемкий поиск в процессе дальнейшей отладки). Трансляция программы на машинных язык. На этом этапе выявляются синтаксические ошибки. Компиляторы с языков Си, Паскаль выдают диагностическое сообщение о синтаксических ошибках в листинге программы (листингом называется выходной документ транслятора, сопровождающий оттранслированную программу на машинном языке ? объектный модуль). Редактирование внешних связей и компоновка программы. На этапе редактирования внешних связей программных модуле программа-редактор внешних связей, или компоновщик задач, обнаруживает такие синтаксические ошибки, как несоответствие числа параметров в описании подпрограммы и обращении к ней, вызов несуществующей стандартной программы. например, 51 H вместо 51 N, различные длины общего блока памяти в вызывающем и вызываемом модуле и ряд других ошибок. Выполнение программы. После устранения обнаруженных транслятором и редактором внешних связей (компоновщиком задач) синтаксических ошибок переходят к следующему этапу ? выполнению программы на ЭВМ на машинном языке: программа загружается в оперативную память, в соответствие с программой вводятся исходные данные и начинается счет. Проявление ошибки в процессе вода исходных данных или в процессе счета приводит к прерыванию счета и выдаче диагностического сообщения рабочей программы. Проявление ошибки дает повод для выполнения отладочных действий; отсутствие же сообщений об ошибках не означает их отсутствия в программе. План тестирования включает при этом проверку правильности полученных результатов для каких-либо допустимых значений исходных данных. Тестирование программы. Если программа выполняется успешно, желательно завершить ее испытания тестированием при задании исходных данных, принимающих предельные для программы значения. а также выходящие за допустимые пределы значения на входе. Контрольные примеры (тесты) ? это специально подобранные задачи, результаты которых заранее известны или могут быть определены без существенных затрат.

Тема 12. Сопровождение ПО на стадии эксплуатации.

лабораторная работа (2 часа(ов)):

Фаза эксплуатация и сопровождение - практическое использование программного изделия. Процедуры сопровождения регламентируют соответствующими стандартами для снижения затрат на этот вид деятельности. Цель сопровождения ? обеспечить удовлетворение реальных потребностей пользователя. Деятельность - работы по внесению изменений в программы и документацию для развития и совершенствования функциональных возможностей программного изделия и повышения его качества, по поддерживанию изделия в рабочем состоянии и по повышению эффективности его использования. Сопровождение программного изделия в результате всегда дает изменение программного продукта. Штат, занятый сопровождением, должен полностью понимать программный продукт, в который необходимо вносить изменения. В некоторых случаях требуется обучение специалистов по сопровождению. В процессе эксплуатации и сопровождения создается Документ, отражающий историю развития проекта. На ранних стадиях эксплуатации существует гарантийный период, когда разработчик сохраняет ответственность за исправление ошибок в программном продукте. Окончание гарантийного периода фиксируется окончательной приемкой, критерием которой служит успешное выполнение всех приемных тестов и подтверждение выполнения всех требований пользователя. Момент окончательной приемки соответствует формальной передаче программного изделия от разработчика к пользователю (обычно организации). Сопровождение программного обеспечения связано с внесением изменений в течение всего времени использования программного изделия. Причины, определяющие необходимость внесения изменений в изделие: - наличие ошибок, - изменение требования пользователя, - появление более совершенных общесистемных программных средств или технических устройств, - изменение организационной структуры, условий и методов работы пользователя.

Тема 13. Управление разработкой ПО.

лабораторная работа (4 часа(ов)):

Структура стандарта ГОСТ ISO/IEC 12207 Первый раздел описывает область применения данного стандарта. 1. Назначение. Устанавливает общую структуру процессов ЖЦ ПС, на которую можно ориентироваться в программной индустрии. Определяет процессы, работы и задачи, которые используются: при приобретении системы, содержащей программные средства, или отдельно поставляемого программного продукта; при оказании программной услуги, а также при поставке, разработке, эксплуатации и сопровождении программных продуктов. 2. Область распространения. Применяется при приобретении систем, программных продуктов и оказании соответствующих услуг; а также при поставке, разработке, эксплуатации и сопровождении программных продуктов и программных компонентов программно-аппаратных средств. 3. Адаптация. Определяет набор процессов, работ и задач, предназначенных для адаптации к условиям конкретных программных проектов. Процесс адаптации заключается в исключении неприменяемых в условиях конкретного проекта процессов, работ и задач. 4. Соответствие. Соответствие стандарту определяется как выполнение всех процессов, работ и задач, выбранных из стандарта в процессе адаптации, для конкретного программного проекта. Выполнение процесса или работы считается завершенным, когда выполнены все требуемые для них задачи в соответствии с предварительно установленными в договоре требованиями. 5. Ограничения. Описывает архитектуру процессов жизненного цикла программных средств, но не определяет детали реализации или выполнения работ и задач, входящих в данные процессы Под качеством ПС принято понимать совокупность характеристик, относящуюся к его способности удовлетворять установленным потребностям. Общепринятой моделью, лежащей в основе оценки качества ПС, является модель, регламентированная в стандарте ISO/IEC 9126-1:2001 ?Информационная технология. Оценка программного продукта. Характеристики качества и руководства по их применению?. В соответствии с данным стандартом модель качества ПС представляет собой иерархическую структуру, состоящую из трех уровней. 1. Характеристики качества (цели) - то, что мы хотим видеть в ПС. 2. Атрибуты качества - свойства ПС, показывающие приближение к цели. 3. Метрики - количественные характеристики степени наличия атрибутов. Верхний уровень данной модели представлен шестью основными характеристиками качества ПС. Это функциональность, надежность, практичность, эффективность, сопровождаемость и мобильность. Функциональность это способность ПС выполнять набор функций, удовлетворяющих заданным потребностям пользователей. Набор указанных функций определяется во внешнем описании ПС. Надежность - это способность ПС безотказно выполнять определённые функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью. Надёжное ПС не исключает наличия в нём ошибок, важно, чтобы эти ошибки при практическом применении этого ПС в заданных условиях проявлялись достаточно редко. Легкость применения это характеристики ПС, которые позволяют минимизировать усилия пользователя по подготовке исходных данных, применению ПС и оценке полученных результатов. Эффективность это отношение уровня услуг, предоставляемых ПС пользователю при заданных условиях, к объему используемых ресурсов. Сопровождаемость это характеристики ПС, которые позволяют минимизировать усилия по внесению изменений для устранения в нем ошибок и по его модификации в соответствии с изменяющимися потребностями пользователей. Мобильность это способность ПС быть перенесенным из одной среды (окружения) в другую, в частности, с одного компьютера на другой. Функциональность и надежность являются обязательными характеристиками качества ПС. Разработка спецификации качества сводится к построению модели качества ПС. В этой модели должен быть перечень всех свойств, которыми требуется обеспечить ПС, и которые в совокупности образуют приемлемое качество ПС.

Тема 14. Документация ПО.

лабораторная работа (4 часа(ов)):

Документация на программное обеспечение ? это документы, сопровождающие программное обеспечение (ПО) ? программу или программный продукт. Эти документы описывают то, как работает программа и/или то, как её использовать. Документирование это важная часть в разработке программного обеспечения, но часто ей уделяется недостаточно внимания. Существует четыре основных типа документации на ПО: архитектурная/проектная ? обзор программного обеспечения, включающий описание рабочей среды и принципов, которые должны быть использованы при создании ПО техническая ? документация на код, алгоритмы, интерфейсы, API пользовательская ? руководства для конечных пользователей, администраторов системы и другого персонала маркетинговая Архитектурная/проектная документация Проектная документация обычно описывает продукт в общих чертах. Не описывая того, как что-либо будет использоваться, она скорее отвечает на вопрос ?почему именно так?? Например, в проектном документе программист может описать обоснование того, почему структуры данных организованы именно таким образом. Описываются причины, почему какой-либо класс сконструирован определённым образом, выделяются паттерны, в некоторых случаях даже даются идеи как можно будет выполнить улучшения в дальнейшем. Ничего из этого не входит в техническую или пользовательскую документацию, но всё это действительно важно для проекта. Техническая документация Это именно то, что подразумевают под термином документация большинство программистов. При создании программы, одного лишь кода, как правило, недостаточно. Должен быть предоставлен некоторый текст, описывающий различные аспекты того, что именно делает код. Такая документация часто включается непосредственно в исходный код или предоставляется вместе с ним. Подобная документация имеет сильно выраженный технических характер и в основном используется для определения и описания API, структур данных и алгоритмов. Часто при составлении технической документации используются автоматизированные средства ? генераторы документации. Они получают информацию из специальным образом оформленных комментариев в исходном коде, и создают справочные руководства в каком-либо формате, например, в виде текста или HTML. Использование генераторов документации и документирующих комментариев многими программистами признаётся удобным средством, по различным причинам. В частности, при таком подходе документация является частью исходного кода, и одни и те же инструменты могут использоваться для сборки программы и одновременной сборки документации к ней. Это также упрощает поддержку документации в актуальном состоянии.

Тема 15. Разработка и стандартизация информационных технологий.

лабораторная работа (4 часа(ов)):

Стандарт в ИТ определяют как общепринятые требования, предъявляемые к техническому, программному, информационному и иному обеспечению, которые обеспечивают возможность стыковки и совместной работы систем. Различают: стандарты де-юре (объявленные и принятые официально); стандарты де-факто (не оформленные в виде документа, но применяемые на практике). В области традиционного ?материального? производства давно сложилась система поддержки и согласования стандартов, а в области информационных технологий многое ещё предстоит сделать. Популярное программное обеспечение не знает границ территорий и достаточно быстро распространяется по всему миру. Поэтому на национальном, межкорпоративном и международном уровнях всё чаще требуется использование общих (унифицированных) международных стандартов. Важно отметить активное использование Интернета при разработке стандартов, в которой принимают участие многие организации и специалисты их различных стран. Это телеконференции с дискуссиями по наиболее важным вопросам; электронное голосование по утверждению проектов стандартов на разных стадиях разработки вплоть до статуса международного стандарта; организация очных семинаров и конференций; организация полного электронного архива, доступного по сети. Развитие информационных технологий связано с национальными и международными стандартами. Международные стандарты создаются на основе шести принципов, определенных Всемирной торговой организацией (ВТО): открытость, прозрачность, непредвзятость и соблюдение консенсуса, эффективность и целесообразность, согласованность и нацеленность на развитие. В России создаётся отечественная нормативная база в области информационных технологий. Для стандартизации информационных технологий, информационно-телекоммуникационных систем и проектирования информационных систем в стране создаются национальные стандарты и другие нормативные документы. Они определяют фундаментальные общие процедуры, положения и требования, которые могут быть использованы в различных предметных областях деятельности. Существуют специализированные организации: ВНИИстандарт, Гостехкомиссии России и др.

4.3 Структура и содержание самостоятельной работы дисциплины (модуля)

N	Раздел Дисциплины	Семестр	Неделя семестра	Виды самостоятельной работы студентов	Трудоемкость (в часах)	Формы контроля самостоятельной работы
1.	Тема 1. Программное обеспечение ЭВМ.	7		подготовка домашнего задания	2	домашнее задание
2.	Тема 2. Пакеты прикладных программ.	7		подготовка домашнего задания	2	домашнее задание
3.	Тема 3. Программные средства.	7		подготовка домашнего задания	2	домашнее задание
4.	Тема 4. Жизненный цикл программного обеспечения.	7		подготовка домашнего задания	2	домашнее задание
5.	Тема 5. Модели жизненного цикла программного обеспечения.	7		подготовка домашнего задания	2	домашнее задание
6.	Тема 6. Разработка требований и внешнее проектирование ПО.	7		подготовка домашнего задания	3	домашнее задание

N	Раздел Дисциплины	Семестр	Неделя семестра	Виды самостоятельной работы студентов	Трудоемкость (в часах)	Формы контроля самостоятельной работы
7.	Тема 7. Структурный подход к проектированию программного обеспечения.	7		подготовка домашнего задания	3	домашнее задание
8.	Тема 8. Проектирование и программирование модулей.	7		подготовка домашнего задания	2	домашнее задание
9.	Тема 9. Объектно-ориентированный подход к проектированию программного обеспечения.	7		подготовка домашнего задания	1	домашнее задание
				подготовка к контрольной работе	1	контрольная работа
10.	Тема 10. Проектирование и разработка интерфейса ПО.	7		подготовка домашнего задания	2	домашнее задание
11.	Тема 11. Тестирование, отладка и сборка ПО.	7		подготовка домашнего задания	2	домашнее задание
12.	Тема 12. Сопровождение ПО на стадии эксплуатации.	7		подготовка домашнего задания	2	домашнее задание
13.	Тема 13. Управление разработкой ПО.	7		подготовка домашнего задания	2	домашнее задание
14.	Тема 14. Документация ПО.	7		подготовка домашнего задания	2	домашнее задание
15.	Тема 15. Разработка и стандартизация информационных технологий.	7		подготовка домашнего задания	3	домашнее задание
				подготовка к контрольной работе	3	контрольная работа
	Итого				36	

5. Образовательные технологии, включая интерактивные формы обучения

Обучение происходит в форме лекционных и лабораторных занятий, а также самостоятельной работы студентов.

Теоретический материал излагается на лекциях. Причем конспект лекций, который остается у студента в результате прослушивания лекции не может заменить учебник. Его цель-формулировка основных утверждений и определений. Прослушав лекцию, полезно ознакомиться с более подробным изложением материала в учебнике. Список литературы разделен на две категории: необходимый для сдачи зачета минимум и дополнительная литература.

Изучение курса подразумевает не только овладение теоретическим материалом, но и получение практических навыков для более глубокого понимания разделов на основе решения задач и упражнений, иллюстрирующих доказываемые теоретические положения, а также развитие абстрактного мышления и способности самостоятельно доказывать утверждения.

Самостоятельная работа предполагает выполнение домашних работ. Практические задания, выполненные в аудитории, предназначены для указания общих методов решения задач определенного типа. Закрепить навыки можно лишь в результате самостоятельной работы.

Кроме того, самостоятельная работа включает подготовку к зачету. При подготовке к сдаче зачета весь объем работы рекомендуется распределять равномерно по дням, отведенным для подготовки к зачету, контролировать каждый день выполнения работы. Лучше, если можно перевыполнить план. Тогда будет резерв времени.

6. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины и учебно-методическое обеспечение самостоятельной работы студентов

Тема 1. Программное обеспечение ЭВМ.

домашнее задание , примерные вопросы:

Программное обеспечение (ПО) и его классификация. Системное и прикладное ПО. Режимы работы и функции операционной системы. Система программирования. Система контроля и диагностики. Прикладные программы и пакеты прикладных программ. История развития прикладного программного обеспечения. Классификация прикладного программного обеспечения.

Тема 2. Пакеты прикладных программ.

домашнее задание , примерные вопросы:

Определение пакетов прикладных программ (ППП). Классификация ППП. Составные части ППП. Модульный принцип формирования пакета. Функции отдельных модулей пакета. Модель предметной области пакета. Статическая и динамическая модели предметной области

Тема 3. Программные средства.

домашнее задание , примерные вопросы:

Понятие программного средства (ПС), программного продукта (ПП) и программного изделия (ПИ). Основные требования, предъявляемые к ПИ как к продукции производственно-технического назначения. Информатика как отрасль производства программных изделий. Развитие отрасли производства программных изделий в России. Понятие рынка программных средств. Маркетинг программных продуктов.

Тема 4. Жизненный цикл программного обеспечения.

домашнее задание , примерные вопросы:

Понятие жизненного цикла (ЖЦ) программного обеспечения. Определение ЖЦ международным стандартом ISO/IEC 12207:1995. Основные процессы ЖЦ ПО. Вспомогательные процессы ЖЦ ПО. Организационные процессы ЖЦ ПО. Взаимосвязь между процессами ЖЦ ПО.

Тема 5. Модели жизненного цикла программного обеспечения.

домашнее задание , примерные вопросы:

Понятие модели и стадии ЖЦ ПО. Характеристика стадий создания ПО. Каскадная и спиральная модели ЖЦ. Подход RAD (Rapid Application Development) к разработке ПО.

Тема 6. Разработка требований и внешнее проектирование ПО.

домашнее задание , примерные вопросы:

Анализ и разработка требований к ПО. Определение целей создания ПО. Разработка внешний спецификаций проекта. Использование программной инженерии при разработки ПО. Понятие CASE ? технологии. Обзор CASE-средств для проектирования ПО.

Тема 7. Структурный подход к проектированию программного обеспечения.

домашнее задание , примерные вопросы:

Характеристика и основные принципы структурного подхода. SADT (Structured Analysis and Design Technique), DFD (Data Flow Diagrams) и ERD (Entity-Relationship Diagrams) модели структурного подхода. Концепции функциональной модели SADT. Состав функциональной модели. Построение иерархии диаграмм моделей стандарта IDEF0. Типы связей между функциями.

Тема 8. Проектирование и программирование модулей.

домашнее задание , примерные вопросы:

Модульный принцип построения и проектирования ПО. Проектирование и кодирование логики модулей. Требования к структуре модуля и взаимодействию модулей между собой. Связность модуля. Сцепление модулей. Этапы программирования. Пошаговая детализация и структурное программирование. Стиль программирования.

Тема 9. Объектно-ориентированный подход к проектированию программного обеспечения.

домашнее задание , примерные вопросы:

Определение и описание архитектуры программного обеспечения. Базовые средства по созданию архитектуры ПО. Способы формального представления знаний. Основы устройства и использование экспертных систем в разработке адаптируемого программного обеспечения. Основные направления интеллектуализации ПО.

контрольная работа , примерные вопросы:

Составление обзора основных стандартов обеспечения качества ПО.

Тема 10. Проектирование и разработка интерфейса ПО.

домашнее задание , примерные вопросы:

Влияние эргономики на удобство работы на компьютере. Психологическая эргономика. Интерфейс программного средства. Принципы проектирования интерфейса. Состав интерфейса системы: процесс ввода/вывода и процесс диалога. Критерии хорошего диалога. Организация управления ПС с входным языком командного типа, с языком командного типа. Организация диалога типа вопрос-ответ и на основе командных форм. Использование смешанной структуры диалога.

Тема 11. Тестирование, отладка и сборка ПО.

домашнее задание , примерные вопросы:

Определение и принципы тестирования ПО. Категории ошибок. Тестирование и отладка программ. Аксиомы тестирования. Средства тестирования. Анализ рисков как средство тестирования. Процесс тестирования. Методы тестирования программ. Методы проектирования тестовых наборов данных. Сборка программ при тестировании.

Тема 12. Сопровождение ПО на стадии эксплуатации.

домашнее задание , примерные вопросы:

Проблемы внедрения, эксплуатации и сопровождения ПО.

Тема 13. Управление разработкой ПО.

домашнее задание , примерные вопросы:

Стандартизация и метрология в разработке программного обеспечения. Понятие качественного ПС и связанные с ним характеристики. Стандартизация показателей качества ПС. Характеристики качества базового международного стандарта ISO 9126:1991. Надежность ПО. Основные количественные показатели надежности. Классификация моделей надежности.

Тема 14. Документация ПО.

домашнее задание , примерные вопросы:

Принципы и стандарты документирования программного обеспечения. Представление стандартов ЕСПД. Документирование стадий разработки, этапов и содержания работ. Типовая структура и содержание эксплуатационных документов пользователей ПО. Типовая структура и содержание технологических документов для разработчиков ПО. Средства документирования.

Тема 15. Разработка и стандартизация информационных технологий.

домашнее задание , примерные вопросы:

Этапы разработки технологических процессов. Параметры технологических процессов. Критерии качества технологических процессов. Критерии оптимизации информационных технологий. Средства проектирования технологических процессов.

контрольная работа , примерные вопросы:

Анализ соответствия выбранному стандарту ПО (на конкретном примере).

Тема . Итоговая форма контроля

Примерные вопросы к зачету:

Вопросы к зачету:

1. Критерии качества программного средства. Определение качества ПО в стандарте ISO 9126. Многоуровневая модель качества ПО. Оценочные характеристики качества программного продукта

2. Жизненный цикл программного продукта, фазы жизненного цикла. Этапы классического жизненного цикла, их содержание.

3. Фаза разработки, этапы процесса разработки. Стратегии конструирования ПО: линейная, инкрементная, эволюционная.

4. Стандарт ISO/IEC 12207-95: основные определения - система, модель жизненного цикла, квалификационные требования. Основные процессы, их содержание, работы и задачи процесса разработки.

5. Стандарт ISO/IEC 15504 (SPICE): оценка возможностей разработчика. Связь этого стандарта с моделью зрелости предприятия SEI CMM.

6. Прогностические модели процесса разработки: каскадная, RAD, спиральная.

7. Адаптивные модели процесса разработки: экстремальное программирование, Scrum.

8. Руководство программным проектом. Предварительные оценки проекта. Системный анализ и анализ требований. Анализ рисков. Планирование процесса разработки. Типовая структура распределения работ.

9. Контроль процесса разработки. Размерно- и функционально-ориентированные метрики. Метрические характеристики объектно-ориентированных систем.

10. Структурный и объектно-ориентированный подходы к разработке ПО. Их сравнительный анализ. Сущность объектного подхода к разработке программных средств.

11. Анализ предметной области: цели и задачи. Модели предметной области. Формальные определения. Классификация моделей. Методология IDEF0, синтаксис IDEF0-моделей.

12. Диаграммы потоков данных (DFD-диаграммы) и диаграммы потоков работ (IDEF3-диаграммы), их использование при моделировании предметной области.

13. Объектно-ориентированный анализ предметной области. Методика определения границ системы и ключевых абстракций. Пример проведения анализа. Функциональные и не-функциональные требования к системе.

14. Функциональные требования к системе. Способ их представления в виде UML-диаграммы. Пример диаграммы с использованием отношений "расширяет" и "включает". Понятие прецедента и сценария.

15. Концептуальная модель системы: концептуальные классы, системные события и системные операции. Способ их представления в виде UML-диаграмм. Пример концептуального описания прецедента.

16. Диаграммы взаимодействия как элементы концептуальной модели. Синтаксис диаграмм взаимодействия.

17. Проектирование программных средств. Цели и задачи этапа проектирования. Понятие модели проектирования, ее отличия от концептуальной модели. Стадии проектирования, их краткая характеристика.

18. Задачи, решаемые на стадии эскизного проектирования. Понятие архитектуры ПС. Проблема выбора архитектуры. Влияние архитектуры на качественные характеристики ПС.

19. Понятие модуля и модульного программирования. Преимущества модульного подхода к разработке ПО. Модули как средство физического структурирования ПО. Свойства модулей.

20. Задачи, решаемые на стадии детального проектирования. Цели и задачи проектирования пользовательского интерфейса.

21. Понятие шаблона. Классификация шаблонов. Стандарт описания шаблонов.

22. Идентификация методов программных классов. Диаграммы классов, способы отображения отношений ассоциации и зависимости. Пример диаграммы классов.

23. Тестирование и отладка программного средства. Стадии тестирования и их характеристика. Основные принципы тестирования. Тесты и тестовые наборы. Понятие тестового покрытия.

24. Отладочное тестирование. Соотношение структурного и функционального подходов. Примеры реализации.

25. Интеграционное тестирование. Виды интеграционного тестирования. Критерии полноты тестовых наборов. Регрессионное тестирование. Критерии завершения отладочного тестирования.

26. Системное тестирование. Виды системного тестирования. Критерии полноты тестовых наборов.

27. Особенности объектно-ориентированного тестирования. Расширение области применения тестирования. Критерии тестирования моделей. Тестирование классов. Тестирование кластеров и потоковое тестирование.

28. Понятие автоматизированного тестирования. Автотесты. Достоинства и недостатки автоматизированного тестирования. Средства автоматизированного тестирования.

29. Утилита модульного тестирования NUnit. Средства описания тестов. Утверждения, параметры утверждений.

30. Понятие версии программного продукта и системы контроля версий. Модели версионирования, их сравнение.

31. Система Subversion, ее архитектура. Хранилище, его структура, правки. Команды SVN для работы с хранилищем. Понятия рабочей копии и служебного каталога. Сценарий объединения правок. Конфликты и способы их разрешения.

32. Понятие сборки, манифест сборки. Сборка приложения, системы автоматизации сборки.

33. Утилита NAnt, файл сборки и его структура. Цели, зависимость целей, описание целей.

34. Документирование процесса разработки. Типы документов управления.

35. Документирование программного продукта. Документация сопровождения, ее назначение и состав. Пользовательская документация, ее назначение и состав.

7.1. Основная литература:

1. Архитектура и проектирование программных систем: Монография / С.В. Назаров. - М.: НИЦ Инфра-М, 2013. - 351 с. URL: http://znanium.com/bookread.php?book=353187

2. Управление качеством программного обеспечения: Учебник / Б.В. Черников. - М.: ИД ФОРУМ: ИНФРА-М, 2012. - 240 с. URL: http://znanium.com/bookread.php?book=256901

3. Основы построения автоматизированных информационных систем: Учебник / В.А. Гвоздева, И.Ю. Лаврентьева. - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 320 с. URL: http://znanium.com/bookread.php?book=392285

7.2. Дополнительная литература:

1. Гагарина Л.Г. Технология разработки программного обеспечения: Учеб. пос. / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул; Под ред. проф. Л.Г.Гагариной - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 400 с. ЭБС "Знаниум": http://znanium.com/bookread.php?book=389963

2.Ченцов С. В. Многоэтапный анализ архитектурной надежности и синтез отказоустойчивого программного обеспечения сложных систем [Электронный ресурс] : монография / А. С. Кузнецов, С. В. Ченцов, Р. Ю. Царев. - Красноярск: Сиб. федер. ун-т, 2013. - 143 с. - ЭБС "Знаниум": http://znanium.com/bookread.php?book=492347

3.Царев, Р. Ю. Мультиверсионное программное обеспечение. Алгоритмы голосования и оценка надёжности [Электронный ресурс] : монография / Р. Ю. Царев, А. В. Штарик, Е. Н. Штарик. - Красноярск: Сиб. федер. ун-т, 2013. - 120 с. - ЭБС "Знаниум": http://znanium.com/bookread.php?book=492377

4. Антамошкин, О. А. Программная инженерия. Теория и практика [Электронный ресурс] : учебник / О. А. Антамошкин. - Красноярск: Сиб. Федер. ун-т, 2012. - 247 с. - ЭБС "Знаниум": http://znanium.com/bookread.php?book=492527

7.3. Интернет-ресурсы:

Википедия - http://ru.wikipedia.org

Интернет-журнал по ИТ - http://www.rsdn.ru

Интернет-издание о высоких технологиях - http://www.cnews.ru/

Интернет-портал образовательных ресурсов по ИТ - http://www.intuit.ru

Компьютерная энциклопедия - http://www.computer-encyclopedia.ru

8. Материально-техническое обеспечение дисциплины(модуля)

Освоение дисциплины "Технологии и стандарты разработки программного обеспечения" предполагает использование следующего материально-технического обеспечения:

Учебно-методическая литература для данной дисциплины имеется в наличии в электронно-библиотечной системе "ZNANIUM.COM", доступ к которой предоставлен студентам. ЭБС "ZNANIUM.COM" содержит произведения крупнейших российских учёных, руководителей государственных органов, преподавателей ведущих вузов страны, высококвалифицированных специалистов в различных сферах бизнеса. Фонд библиотеки сформирован с учетом всех изменений образовательных стандартов и включает учебники, учебные пособия, УМК, монографии, авторефераты, диссертации, энциклопедии, словари и справочники, законодательно-нормативные документы, специальные периодические издания и издания, выпускаемые издательствами вузов. В настоящее время ЭБС ZNANIUM.COM соответствует всем требованиям федеральных государственных образовательных стандартов высшего профессионального образования (ФГОС ВПО) нового поколения.

лабораторные занятия по дисциплине проводятся в аудитории, оснащенной доской и мелом (маркером)

Программа составлена в соответствии с требованиями ФГОС ВПО и учебным планом по направлению 02.03.02 "Фундаментальная информатика и информационные технологии" и профилю подготовки Системный анализ и информационные технологии .

Автор(ы):
Тагиров Р.Р. _____
Шаймухаметов Р.Р. _____
"__" _____ 201 __ г.

Рецензент(ы):
Хабибуллин Р.Ф. _____
"__" _____ 201 __ г.