МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

«СОГЛАСОВАНО»

Руководитель ОП

д.ф.-м.н., профессор, академик РАН, Гузев М.А.

_____
(подпись) (Ф.И.О. рук. ОП)

«9» июля 2018 г.

«УТВЕРЖДАЮ»

Заведующая (ий) кафедрой

информатики, математического и компьютерного моделирования
(название кафедры)

_____ Чеботарев А.Ю.
(подпись) (Ф.И.О. зав. каф.)

«9» июля 2018 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ (РПУД)

Программная инженерия

Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа «Бакалавриат»
Профиль: «Прикладная информатика в компьютерном дизайне»
Форма подготовки: очная

курс 2 семестр 4
лекции 18 час.
практические занятия _____ час.
лабораторные работы 54 час.
в том числе с использованием МАО лек. _____ / пр. _____ / лаб. _____ час.
всего часов аудиторной нагрузки 72 час.
в том числе с использованием МАО _____ час.
самостоятельная работа 36 час.
зачет 4 Семестр
экзамен _____ Семестр

Рабочая программа составлена в соответствии с требованиями образовательного стандарта, самостоятельно установленного ДВФУ, принятого решением Ученого совета Дальневосточного федерального университета, протокол от 28.01.2016 № 01-16, и введенного в действие приказом ректора ДВФУ от 18.02.2016 № 12-13-235.
Рабочая программа обсуждена на заседании кафедры информатики, математического и компьютерного моделирования, протокол №22 «23» июня 2017 г..

Заведующий (ая)кафедрой А.Ю.Чеботарёв, д.ф-м.н., профессор
Составитель (ли): ст. преподаватель Кленина Н.В.

Оборотная сторона титульного листа РПУД

I. Рабочая программа пересмотрена на заседании кафедры:
Протокол от«_____»_____201 г. № _____

Заведующий (ая) кафедрой_____ _____
(подпись) (И.О. Фамилия)

II. Рабочая программа пересмотрена на заседании кафедры:
Протокол от«_____»_____201 г. № _____

Заведующий (ая) кафедрой_____ _____
(подпись) (И.О. Фамилия)

АННОТАЦИЯ

учебно-методического комплекса дисциплины

«Программная инженерия»

Рабочая учебная программа дисциплины «Программная инженерия» разработана для студентов 2 курса по направлению 09.03.03 «Прикладная информатика» в соответствие с требованиями ФГОС ВПО по данному направлению и положением об учебно-методических комплексах дисциплин образовательных программ высшего профессионального образования (утверждено приказом и.о. ректора ДВФУ от 17.04.2015 № 12-13-87)

Дисциплина «Программная инженерия» входит в базовую часть Математического и естественнонаучного цикла.

Общая трудоемкость освоения дисциплины составляет 3 зачетных единиц, 144 часа. Учебным планом предусмотрены лекционные занятия (18 часов), практическая работа (54 часа), самостоятельная работа студента (36 часа). Дисциплина реализуется на 2 курсе в 2 семестре.

Содержание дисциплины охватывает знания о теоретических основах инженерии программных продуктов, включая основные этапы проектирования и реализации программных продуктов, в том числе анализ предметной области, обзор существующих решений, постановка задачи, набор требований и проектных решений, основы построения интерфейса, описание данных и алгоритмов, методология тестирования.

В рамках дисциплины рассматривается следующий круг вопросов:

— создание основы знаний использования современных средств проектирования, реализации и сопровождения прикладных программ, необходимой при изучении студентами, общепрофессиональных и специальных дисциплин;

— освоение предусмотренного программой теоретического материала и приобретение практических навыков использования технологий создания информационных систем на базе современных ПК, а также навыков программирования.

Изучаемая дисциплина формирует основные компетенции специалиста в области технологий программирования. Целью изучения дисциплины «Программная инженерия» является формирование системного подхода к спецификации, проектированию и созданию программного продукта.

Дисциплина направлена на формирование общекультурных и профессиональных компетенций выпускника.

Уделяется внимание вопросам практического применения аспектов теории проектирования программных продуктов.

1. Цели освоения дисциплины

Целями освоения дисциплины «Программная инженерия» являются:

– достижение понимания студентами сущности системного подхода к автоматизации и информатизации решения прикладных задач, к построению программных систем на основе современных информационно-коммуникационных технологий;

– освоение студентами современных технологий разработки проектов автоматизации и информатизации прикладных процессов;

– выработка умений: моделирование прикладных и информационных процессов; подготовка обзоров; научных докладов;

– формирование навыков моделирования прикладных и информационных процессов; формирования требований к информатизации и автоматизации прикладных процессов;

– формирование и развитие способностей к формализации и структуризации информации, самостоятельно приобретать и использовать в

практической деятельности новые знания и умения; стремления к саморазвитию;

– выработка умений: использовать, обобщать и анализировать информацию, ставить цели и находить пути их достижения в условиях формирования и развития информационного общества; применять системный подход и математические методы в формализации решения прикладных задач;

– формирование и развитие способностей к суждениям, способностей логически верно, аргументированно и ясно строить устную и письменную речь; самостоятельно приобретать и использовать в практической деятельности новые знания и умения; стремления к саморазвитию;

В результате освоения данной дисциплины бакалавр приобретает знания, умения и навыки, обеспечивающие достижение целей основной образовательной программы «Прикладная информатика».

Дисциплина должна:

- познакомить студентов с теоретическими основами технологии разработки программных продуктов;
- научить студентов самостоятельно осваивать дополнительные инструментальные средства программирования;
- научить студентов использовать широко распространенные алгоритмы и алгоритмические методы;
- научить студентов разрабатывать прикладные программы, использующие языки программирования высокого уровня
- научить студентов документированию программного проекта;
- содействовать студентам в приобретении опыт публичных выступлений.

2. Место дисциплины в структуре ОП бакалавриата

Дисциплина «Программная инженерия» относится к циклу Цикл профессиональных дисциплин ОП (базовая часть Математического и естественнонаучного цикла).

Логическая взаимосвязь с Гуманитарным, социальным и экономическим циклом в разделах: структура научного познания, его сущность, методы и формы.

Содержательно-методическая взаимосвязь с циклами:

«Информационные ресурсы и системы», в разделах: назначение и виды ИКТ; информационная безопасность; технологии сбора, накопления, обработки, передачи и распространения информации; модели данных;

«Математический и естественнонаучный цикл», в разделах: информация, знания; информационные процессы, информационные системы и, технологии.

Близкая по содержанию дисциплина – «Информатика и программирование».

Предшествующая дисциплина: «Информатика и ИКТ», школьный курс.

Требования к «входным» знаниям, приобретенным в результате освоения предшествующих дисциплин:

- владение основами компьютерных и информационных технологий;
- знание основ алгоритмизации;
- начальные умения формализации и моделирования.

Теоретические дисциплины и практики, для которых освоение данной дисциплины необходимо как предшествующее: «Проектирование ИС», «Вычислительные системы, сети и телекоммуникации», «Операционные системы».

Для изучения дисциплины студент должен:

Знать:

– основы алгоритмизации и программирования;

– базовые инструменты проектирования и структурирования программных продуктов.

Уметь:

– программировать на одном из алгоритмических языков;

– осуществлять сбор и анализ информации;

– строить простые оконные приложения;

– решать простые задачи на алгоритмизацию.

Владеть:

– навыками работы в интегрированной среде;

– методами алгоритмизации и программирования;

– навыками отладки приложений.

В результате изучения данной дисциплины у обучающихся формируются следующие общекультурные/ общепрофессиональные/ профессиональные компетенции (элементы компетенций).

Код и формулировка компетенции	**Этапы формирования компетенции**	
ПК-8 способен проводить обследование организаций, выявлять информационные потребности пользователей, формировать требования к информационной системе, участвовать в реинжиниринге прикладных и информационных процессов	Знает	Методы исследования предметной области в ее проблематизации
	Умеет	Выполнять обзор и анализ существующих решений под минимальным руководством преподавателя, с использованием методических указаний. Самостоятельно оценивать надежность и работоспособность информационных систем средней сложности.
	Владеет	Навыками формирования требований к информационной системе, методами проектирования и реализации ИС; навыками тестирования и отладки приложений.

ПК-13 способен принимать участие во внедрении, адаптации и настройке прикладных ИС	Знает	Основные этапы выявления проблем в реализации ИС и задач в области проектирования. Требования к построению программных систем на основе современных информационно-коммуникационных технологий.
	Умеет	Формировать технические, функциональные и прочие требования к разрабатываемой системе. Выполнять обзор существующих решений. Обосновывать необходимость и целесообразность адаптации и настройки проекта.
	Владеет	Навыками работы с современными информационно-коммуникационными средствами.
ПК-16 способен оценивать и выбирать современные операционные среды и информационно-коммуникационные технологии для информатизации и автоматизации решения прикладных задач и создания ИС	Знает	Средства получения информации о предметной области в глобальных компьютерных сетях, основные принципы доступа к библиотечным ресурсам.
	Умеет	Самостоятельно осваивать отдельные темы дисциплины под минимальным руководством преподавателя, с использованием методических указаний. Самостоятельно оценивать надежность и работоспособность информационных систем средней сложности.
	Владеет	Навыками подбора среды разработки и работы в ней; методами проектирования, алгоритмизации и программирования; навыками тестирования и отладки приложений.

I. СТРУКТУРА И СОДЕРЖАНИЕ ТЕОРЕТИЧЕСКОЙ ЧАСТИ КУРСА

Модуль 1. Основные принципы проектирования программных продуктов (10 час.)

Тема 1.1. Введение в предмет «Программная инженерия». (4 час.)

• Дисциплина программной инженерии. История возникновения, тенденции и проблемы развития.

• Связь и сравнение с другими дисциплинами.

• Основные цели и назначение требований технологии проектирования программных продуктов.

Тема 1.2. Этапы проектирования программных продуктов (6 час.)

• Анализ предметной области. Проблематизация ПО. Анализ системных требований. Определение терминологии проекта.

• Обзор существующих решений. Постановка задачи.

• Описание данных проекта. Структура данных, формат данных, источники данных. Формальное описание данных. Потоки управления данными. Организация взаимодействия с пользователем. Организация редактирования данных.

• Основы проектирования интерфейса. Понятие и основные принципы удобства для пользователя. Обеспечение гибкости, ясности, надежности, простоты обучения, переносимости программной системы.

• Технология документирования этапов проектирования программных продуктов.

• Введение в объектно-ориентированные технологии.

Модуль 2. Основные принципы проектирования программных продуктов (8 час.)

Тема 2.1.Конструирование программного обеспечения. (2 час.)

• Выбор инструментальных средств реализации программного продукта.

• Разработка архитектуры программной системы. Технология документирования архитектуры программной системы.

• Алгоритмы и математические методы программной системы: разработка и описание

Тема 2.2. Реализация программной системы. (4 час.)

• Управление разработкой программного обеспечения. План реализации программной системы. Принцип модульности и автономности.

• Технология тестирования. Проект тестирования программной системы.

• Оценка производительности и эффективности программной системы.

• Документирование этапа реализации программной системы.

Тема 2.3 Эксплуатация и сопровождение программной системы. (2/0 час.)

• Средства и технологии поддержки пользователей.

• Оценка качества программного обеспечения.

• Обслуживание программной системы.

• Управление конфигурацией системы .

II. СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ КУРСА

Практическая работа №1. Постановка задачи (0/9 час.)

• Формирование состава проектных групп.

• Выбор темы проекта.

• Анализ и проблематизация предметной области.

• Обзор и анализ существующих решений.

• Формулировка неформальной постановки задачи.

• Построение диаграммы вариантов использования.

• Документирование результатов работы.

Практическая работа №2. Проектные решения. (0/9 час.)

• Обзор и разработка необходимых математических методов решения и алгоритмов.

• Обоснование выбора средств реализации.

• Формирование общих требований к программному продукту.

• Формирование функциональных требований программной системы.

• Документирование результатов работы.

Практическая работа №3. Разработка проекта программной системы (0/9 час.)

• Проектирование интерфейса программной системы.

• Построение диаграммы состояний.

• Разработка архитектуры программной системы. Компоненты системы.

• Документирование результатов работы.

Практическая работа №4. Структуры данных программной системы (0/9 час.)

• Описание внутренних структур данных.

• Построение диаграммы сущностей.

• Описание формата файлов.

• Описание вешнего формата данных.

Практическая работа №5. Реализация программной системы (0/9час.)

• Разработка плана реализации программной системы.

• Автономная реализация компонент системы.

• Разработка стратегия тестирования программной системы.

• Автономная и комплексная отладка программной системы.

• Экспериментальное тестирование программной системы.

• Технические характеристики системы.

Практическая работа №6. Презентация программного продукта. (0/9 час.)

• Оформление отчета о разработке программной системы.

• Разработка и оформление пользовательской инструкции.

• Разработка устного доклада о разработке программной системы.

• Разработка и создание презентации к устному докладу.

• Защита проекта. Оценка проектов других проектных групп.

III. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Учебно-методическое обеспечение самостоятельной работы обучающихся по дисциплине «Программная инженерия» представлено в Приложении 1 и включает в себя:

план-график выполнения самостоятельной работы по дисциплине, в том числе примерные нормы времени на выполнение по каждому заданию;

характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению;

требования к представлению и оформлению результатов самостоятельной работы;

критерии оценки выполнения самостоятельной работы.

IV. КОНТРОЛЬ ДОСТИЖЕНИЯ ЦЕЛЕЙ КУРСА

Типовые контрольные задания, методические материалы, определяющие процедуры оценивания знаний, умений и навыков и (или) опыта деятельности, а также критерии и показатели, необходимые для оценки знаний, умений, навыков и характеризующие этапы формирования компетенций в процессе освоения образовательной программы, представлены в Приложении 2.

V. СПИСОК УЧЕБНОЙ ЛИТЕРАТУРЫ И ИНФОРМАЦИОННО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Основная литература

(электронные и печатные издания)

1. Основы алгоритмизации и программирования: Учебное пособие / В.Д. Колдаев; Под ред. Л.Г. Гагариной. - М.: ИД ФОРУМ: ИНФРА-М, 2015. - 416 с.: ил.; 60x90 1/16. - (Профессиональное образование). (переплет) ISBN 978-5-8199-0279-0 - Режим доступа: http://znanium.com/catalog/product/484837

2. Киселева Т.В. Программная инженерия. Часть 1 [Электронный ресурс] : учебное пособие / Т.В. Киселева. — Электрон. текстовые данные. — Ставрополь: Северо-Кавказский федеральный университет, 2017. — 137 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/69425.html

3. Батоврин, В.К. Системная и программная инженерия. Словарь-справочник [Электронный ресурс] : учебное пособие / В.К. Батоврин. — Электрон. дан. — Москва : ДМК Пресс, 2010. — 280 с. — Режим доступа: https://e.lanbook.com/book/1097. — Загл. с экрана.

4. Методические указания по дисциплине Программная инженерия [Электронный ресурс] / . — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2013. — 24 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/61752.html

5. Кленин А.С. Методические указания по подготовке проектных работ для студентов направлений «Прикладная математика и информатика» (профиль системное программирование) и «Прикладная информатика» (Версия 2.0/А.С. Кленин, Н.В. Кленина – Владивосток. Дальневост. федерал. ун-т, 2016. – 72 с..

Дополнительная литература

(печатные и электронные издания)

1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс] / Б. Мейер. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 285 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/39552.html

2. Липаев В.В. Программная инженерия сложных заказных программных продуктов [Электронный ресурс] : учебное пособие / В.В. Липаев. — Электрон. текстовые данные. — М. : МАКС Пресс, 2014. — 309 c. — 978-5-317-04750-4. — Режим доступа: http://www.iprbookshop.ru/27297.html

3. Батоврин В.К. Системная и программная инженерия. Словарь-справочник [Электронный ресурс] : учебное пособие для вузов / В.К. Батоврин. — Электрон. текстовые данные. — Саратов: Профобразование, 2017. — 280 c. — 978-5-4488-0129-7. — Режим доступа: http://www.iprbookshop.ru/63956.html

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. http://www.moeobrazovanie.ru/specialities_vuz/programmnaya_inzheneriya.html
2. http://progbook.net/thnologi_programming/3709-texnologiya-razrabotki-programmnogo-obespecheniya.html
3. https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%8F

Перечень информационных технологий и программного обеспечения

- RussianLanguagePackfor .NET v4.0 (русификация сообщений об ошибках времени выполнения)
- Автоматическая тестирующая система ДВФУ https://imcs.dvfu.ru/cats/

VI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Дисциплина «Программная инженерия»» является базисом для программиста любого профиля и поэтому изучается студентами по направлению подготовки 09.03.03 «Прикладная информатика».

Процесс изучения дисциплины осуществляется в следующих организационных формах:

- выполнение аудиторных практическх работ;
- самостоятельное изучение материала;
- выполнение контрольных работ;
- подготовка и сдача зачета.

В дисциплине можно выделить две области:

- базовые знания, относительно стабильные, составляющие ядро дисциплины;
- технологические знания, связанные с освоением конкретных программных сред и языков программирования.

Базовые знания основных принципов алгоритмизации, понимание процесса работы программы, обработки компьютером данныхобразуют понятийное ядро дисциплины и служат основой для изучения многих дисциплин специальности.Эта область включает в себя системный подход к решению прикладных информационных задач, алгоритмическое мышление, знание терминологии и современных средств разработки программного обеспечения.

Технологическая часть дисциплины связана с практическим освоением умений и навыков построения алгоритмов и программирования в наиболее распространенных программных средах. Отдельное внимание на занятиях уделяется различным способам организации коллективного взаимодействия при разработке программ, решению стандартных технологических задач.

Практические работы проводится в компьютерных классах и подкреплены методическими указаниями, рекомендациями и требованиями к представлению и оформлению результатов работы.

Самостоятельная работа включает изучение теоретического материала дисциплины и выполнение индивидуальных работ.

Для изучения дисциплины приводится перечень рекомендуемой литературы, методические указания и вопросы к контрольным заданиям и экзамену.

В качестве основы для изучения дисциплины можно взять учебники, учебные пособия, электронные материалы и методические указания, приведенные в списке литературы.

При изучении теоретического материала следует по методическим указаниям ознакомиться с планом темы. Освоив теоретический материал, необходимо самостоятельно, без помощи литературы, сделать попытку ответить на вопросы по теме. С каждой темой связан перечень ключевых понятий. После изучения темы необходимо уметь самостоятельно давать определение понятий.

VII. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Компьютерный класс: 15 Моноблоков/HPP-B0G08ES#ACB| HP 8200E AiO i52400S 500G 4/0G 28PC

Проектор 3-chip DLP, 10 600 ANSI-лм, WUXGA 1 920x1 200 (16:10) PT-DZ110XE Panasonic; экран 316x500 см, 16:10 с эл. приводом; крепление настенно-потолочное ElproLargeElectrolProjecta; профессиональная ЖК-панель 47", 500 Кд/м2, Full HD M4716CCBA LG; подсистема видеоисточников документ-камера CP355AF Avervision; подсистема видеокоммутации; подсистема аудиокоммутации и звукоусиления; подсистема интерактивного управления; беспроводные ЛВС обеспечены системой на базе точек доступа

Корпус 20, ауд. D 734, 734а

Приложение 1

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

по дисциплине «Программная инженерия»

Направление подготовки 09.03.03 Прикладная информатика

Прикладная информатика в компьютерном дизайне

Форма подготовки очная

Владивосток
2017

План-график выполнения самостоятельной работы по дисциплине

№ п/п	Контролируемые разделы / темы дисциплины	Коды и этапы формирования компетенций		Оценочные средства	
				текущий контроль	промежуточная аттестация
1	Формализация постановки задачи	ОК-4, ПК-8, ПК-13 ПК-16	знает	Практическая работа (ПР-1)	Зачет
			умеет	Практическая работа (ПР-1)	
			владеет	Практическая работа (ПР-2)	
2	Проектирование компонент программного продукта	ОК-4, ПК-8, ПК-13 ПК-16	знает	Практическая работа (ПР-3)	Зачет
			умеет	Практическая работа (ПР-3)	
			владеет	Практическая работа (ПР-3)	
3	Создание компонент программного обеспечения	ОК-4, ПК-8, ПК-13 ПК-16	знает	Практическая работа (ПР-4)	Зачет
			умеет	Практическая работа (ПР-4)	
			владеет	Практическая работа (ПР-4)	
4	создании технической документации по результатам работы	ОК-4, ПК-8, ПК-13 ПК-16	знает	Практическая работа (ПР-5, ПР-6)	Зачет
			умеет	Практическая работа (ПР-5, ПР-6)	
			владеет	Практическая работа (ПР-5, ПР-6)	

Рекомендации по самостоятельной работе студентов

Содержание самостоятельной работы студентов по дисциплине:

- работа с лекционным материалом, поиск и обзор литературы и электронных источников информации по заданной теме;
- изучение тем, вынесенных на самостоятельную проработку;
- подготовка к лабораторным работам;
- подготовка к зачёту.

Творческая проблемно-ориентированная самостоятельная работа

направлена на развитие интеллектуальных умений, комплекса универсальных (общекультурных) и профессиональных компетенций, повышение творческого потенциала бакалавров и заключается в:

- поиске, анализе, структурировании и презентации информации;
- разработке учебного программного продукта;
- исследовательской работе и участии в научных студенческих семинарах и олимпиадах;
- анализе научных публикаций по заранее определенной преподавателем теме;
- широко применяется метод коллективной разработки и взаимной оценки программных систем, распределения ролевых функций между студентами.

Приложение 2

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

НАЗВАНИЕ ШКОЛЫ (ФИЛИАЛА)

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине «Программная инженерия»

Направление подготовки 09.03.03 Прикладная информатика

Прикладная информатика в компьютерном дизайне

Форма подготовки очная

Владивосток
2017

Программа зачета

1. Проведение научных исследований (экспериментов, наблюдений и количественных измерений) программных продуктов, проектов, процессов, методов и инструментов программной инженерии
2. Построение моделей программных проектов и программных продуктов с использованием инструментальных средств компьютерного моделирования
3. Описание проводимых исследований, подготовка данных для составления обзоров и отчетов
4. Сбор и анализ требований заказчика к программному продукту
5. Оценка выбора вариантов программного обеспечения
6. Подготовка коммерческого предложения заказчику, презентации
7. Проектирование компонент программного продукта в объеме, необходимом для их конструирования в рамках поставленного задания
8. Создание компонент программного обеспечения (кодирование, отладка, модульное и интеграционное тестирование)
9. Измерение кода в соответствии с планом
10. Разработка тестового окружения и создание тестовых сценариев
11. Разработка и оформление эскизной, технической и рабочей проектной документации
12. Средства автоматизированного проектирования, разработки, тестирования и сопровождения программного обеспечения
13. Методы и инструментальные средства управления инженерной деятельностью и процессами жизненного цикла программного обеспечения
14. Контроль, оценка и обеспечение качества программной продукции
15. Процесс разработки программного обеспечения

16. Участие в создании технической документации по результатам выполнения работ
17. Обучение и аттестация пользователей программных систем
18. Разработка методик обучения технического персонала и пособий по применению программных систем
19. Техническая документация (графики работ, инструкции, планы, оборудование, программное обеспечение)
20. Планирование и координирование работы по настройке и сопровождению программного продукта
21. Организация работы малых коллективов исполнителей программного проекта
22. Ввод в эксплуатацию программного обеспечения (инсталляция, настройка параметров, адаптация, администрирование)
23. Профилактическое и корректирующее сопровождение программного продукта в процессе эксплуатаци

ВВЕДЕНИЕ

Программная инженерия (англ. software engineering) — приложение систематического измеримого подхода к развитию, оперированию и обслуживанию программного обеспечения, а также исследованию этих подходов; то есть, приложение дисциплины инженерии к программному обеспечению.

Это интегрирование принципов математики, информатики и компьютерных наук с инженерными подходами, разработанными для производства осязаемых материальных артефактов.

Также программная инженерия определяется как системный подход к анализу, проектированию, оценке, реализации, тестированию, обслуживанию и модернизации программного обеспечения, то есть применение инженерии к разработке программного обеспечения.

Процесс поставки задачи охватывает действия и задачи, выполняемые поставщиком, который снабжает заказчика программным продуктом или услугой. Данный процесс включает действия:

Инициирование поставки — заключается в рассмотрении поставщиком заявочных предложений и принятии решения согласиться с выставленными требованиями и условиями или предложить свои.

Планирование включает задачи:

- принятие решения поставщиком относительно выполнения работ своими силами или с привлечением субподрядчика;
- разработку поставщиком плана управления проектом, содержащего организационную структуру проекта, разграничение ответственности,
- технические требования к среде разработки и ресурсам, управление субподрядчиком.

Процесс разработки предусматривает действия и задачи, выполняемые разработчиком, и включает следующие действия:

Подготовительная работа начинается с выбора модели жизненного цикла ПО, соответствующей масштабу, значимости и сложности проекта.

Действия и задачи процесса должны соответствовать выбранной модели. Разработчик должен выбрать, адаптировать к условиям проекта и использовать согласованные с заказчиком стандарты, методы и средства разработки, а также составить план выполнения работ.

Анализ требований к системе подразумевает определение ее функциональных возможностей, пользовательских требований, требований к

надежности и безопасности, требований к внешним интерфейсам и т. д.

Требования с системе оцениваются исходя из критериев реализуемости и возможности проверки при тестировании.

Проектирование архитектуры системы на высоком уровне заключается в определении компонентов ее оборудования, ПО и операций, выполняемых эксплуатирующим систему персоналом.

Архитектура системы должна соответствовать требованиям, предъявляемым к системе, а также принятым проектным стандартам и методам.

Анализ требований к ПО предполагает определение следующих

характеристик для каждого компонента ПО:

- функциональных возможностей, включая характеристики производительности и среды функционирования компонента;
- внешних интерфейсов;
- спецификаций надежности и безопасности;
- эргономических требований;
- требований к используемым данным;

- требований к установке и приемке;
- требований к пользовательской документации;
- требований к эксплуатации и сопровождению.

Требования к ПО оцениваются исходя из критериев соответствия требованиям к системе, реализуемости и возможности проверки при тестировании.

Проектирование архитектуры ПО включает задачи (для каждого компонента ПО):

- создание по требованям к ПО в архитектуру, определяющую на высоком уровне структуру ПО и состав ее компонентов;
- разработку и документирование программных интерфейсов ПО и баз данных;
- разработку предварительной версии пользовательской документации;
- разработку и документирование предварительных требований к тестам и планам интеграции ПО.

Архитектура компонентов ПО должна соответствовать требованиям, предъявляемым к ним, а также принятым проектным стандартам и методам.

Детальное проектирование ПО включает следующие задачи:

- описание компонентов и интерфейсов между ними на более низком уровне, достаточном для их последующего самостоятельного кодирования и тестирования;
- разработку и документирование детального проекта базы данных;
- обновление (при необходимости) пользовательской документации;
- разработку и документирование требований к тестам и плана
- тестирования компонентов ПО;
- обновление плана интеграции ПО.

Кодирование и тестирование ПО охватывает задачи:

- разработку и документирование каждого компонента ПО и базы данных а также совокупности тестовых процедур и данных для их тестирования;
- тестирование каждого компонента ПО и базы данных на соответствие предъявляемых к ним требованиям (результаты тестирования компонентов должны быть документированы);
- обновление (при необходимости) пользовательской документации;
- обновление плана интеграции ПО.

Интеграция ПО предусматривает сборку разработанных компонентов ПО в соответствии с планом интеграции и тестирование агрегированных компонентов.

Для каждого из агрегированных компонентов разрабатываются наборы тестов и тестовые процедуры, предназначенные для проверки каждого из квалификационных требований при последующем квалификационном тестировании.

Квалификационное тестирование — это набор критериев и условий, которые необходимо выполнить, чтобы квалифицировать программный продукт как соответствующий своим спецификациям и готовый к использованию в условиях эксплуатации.

Квалификационное тестирование ПО проводится разработчиком в присутствии заказчика (по возможности) для демонстрации того, что ПО удовлетворяет своим спецификациям и готово к использованию в условиях эксплуатации. Квалификационное тестирование выполняется для каждого компонента ПО по всем разделам требований при широком варьировании тестов.

При этом также проверяются полнота технической и пользовательской документации и ее адекватность самим компонентам ПО.

Интеграция системы заключается в сборке всех ее компонентов, включая ПО и оборудование. После интеграции система, в свою очередь, подвергается квалификационному тестированию на соответствие совокупности требований к ней. При этом также производится оформление и проверка полного комплекта документации на систему.

Установка ПО осуществляется разработчиком в соответствии с планом в той среде и на том оборудовании, которые предусмотрены договором. В процессе установки проверяется работоспособность ПО и баз данных. Если устанавливаемое программное обеспечение заменяет существующую систему, разработчик должен обеспечить их параллельное функционирование в соответствии с договором.

Приемка ПО предусматривает оценку результатов квалификационного тестирования ПО и системы и документирование результатов оценки, которые проводятся заказчиком с помощью разработчика. Разработчик выполняет окончательную передачу ПО заказчику в соответствии с договором, обеспечивая при этом необходимое обучение и поддержку.

Процесс эксплуатации охватывает действия и задачи оператора – организации, эксплуатирующей систему и включает действия:

Подготовительная работа — проведение оператором следующих задач:

- планирование действий и работ, выполняемых в процессе
- эксплуатации, и установку эксплуатационных стандартов;
- определение процедур локализации и разрешения проблем, возникающих в процессе эксплуатации.

Эксплуатационное тестирование осуществляется для каждой очередной редакции программного продукта, после чего она передается в эксплуатацию. Эксплуатация системы выполняется в предназначенной для этого среде в соответствии с пользовательской документацией.

Поддержка пользователей заключается в оказании помощи и консультаций

при обнаружении ошибок в процессе эксплуатации ПО.

Процесс сопровождения предусматривает действия и задачи, выполняемые

службой сопровождения. В соответствии со стандартом IEEE-90 под сопровождением понимается внесение изменений в ПО в целях исправления

ошибок, повышения производительности или адаптации к изменившимся условиям работы или требованиям.

ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

Прежде всего, хорошая программа должна делать то, что ожидает от нее заказчик – т.е. удовлетворять требованиям заказчика. Такие требования называют функциональными.

Но кроме функциональных требований, существует еще ряд общих характеристик, которым в той или иной степени должна обладать каждая программа. Эти характеристики принято называть нефункциональными требованиями.

К нефункциональным требованиям относят:

Сопровождаемость (maintainability). Сопровождаемость означает, что программа должна быть написана с расчетом на дальнейшее развитие. Это критическое свойство системы, т.к. изменения ПО неизбежны вследствие изменения бизнеса.

Сопровождение программы выполняют, как правило, не те люди, которые ее разрабатывали. Сопровождаемость включает такие элементы как

- наличие и понятность проектной документации,
- соответствие проектной документации исходному коду,
- понятность исходного кода,
- простота изменений исходного кода,
- простота добавления новых функций.

*Надежность (*dependability). Надежность ПО включает такие элементы как:

1. Отказоустойчивость – возможность восстановления программы и данных в случае сбоев в работе.
2. Безопасность – сбои в работе программы не должны приводить к опасным последствиям (авариям).

3. Защищенность от случайных или преднамеренных внешних воздействий (защита от дурака, вирусов, спама).

Эффективность (efficiency). ПО не должно впустую тратить системные ресурсы, такие как память, процессорное время, каналы связи. Поэтому эффективность ПО оценивается следующими показателями:

1. время выполнения кода,
2. загруженность процессора,
3. объем требуемой памяти,
4. время отклика и т.п.

Удобство использования (usability). ПО должно быть легким в использовании, причем именно тем типом пользователей, на которых рассчитано приложение. Это включает в себя интерфейс пользователя и адекватную документацию.

Причем, пользовательский интерфейс должен быть не интуитивно, а

профессионально понятным пользователю.

Следует отметить, что реализация нефункциональных требований часто требует больших затрат, чем функциональных.

Так, сопровождаемость требует значительных усилий по поддержанию соответствия проекта исходному коду и применения специальных методов создания модифицируемых программ.

Надежность – дополнительных средств восстановления системы при сбоях. Эффективность – поиска специальных архитектурных решений и оптимизации кода.

А удобство – проектирования не «интуитивно» понятного, а профессионально понятного интерфейса пользователя.

ПРОФЕССИОНАЛЬНЫЕ И ЭТИЧЕСКИЕ ТРЕБОВАНИЯ

Развитие средств вычислительной техники, коммуникаций и программных систем (Internet, телекоммуникации, распределенные системы, IP телефония, компьютерные игры и обучающие программы) оказывает все большее воздействие на общество.

Роль специалистов по программному обеспечению при этом все время возрастает. Они работают в определенном правовом и социальном окружении, находятся под действием, международных, национальных и местных законодательств.

Ясно, что программисты, как и специалисты других профессий, должны быть честными и порядочными людьми. Но вместе с тем, программисты не могут руководствоваться только моральными нормами или юридическими ограничениями, т.к. они обычно бывают связаны более тонкими профессиональными обязательствами:

Конфиденциальность – программные специалисты должны уважать конфиденциальность в отношении своих работодателей или заказчиков независимо от того, подписывалось ли ими соответствующее соглашение.

Компетентность – программный специалист не должен завышать свой истинный уровень компетентности и не должен сознательно браться за работу, которая этому уровню не соответствует.

Защита интеллектуальной собственности – специалист должен соблюдать законодательство и принципы защиты интеллектуальной собственности при использовании чужой интеллектуальной собственности. Кроме того, он должен защищать интеллектуальную собственность работодателя и клиента.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

<ШКОЛА ЕСТЕСТВЕННЫХ НАУК>

МАТЕРИАЛЫ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

по дисциплине «Программная инженерия»

Направление— 230700.62 «Прикладная информатика»

г. Владивосток

Практические задания

Общие требования

Разработать и реализовать групповой программный проект, соответствующий условию задания. Сформировать пакет документации к проекту. Разработать презентацию и доклад, произвести защиту проекта.

Задание 1.

Разработать и реализовать программный проект, производящий расчеты по коммерческому кредиту отдельного пользователя.

Программа должна позволять пользователю:

- Рассчитывать ежемесячный платёж по кредиту.
- Показывать общую сумму долга.
- Показывать переплату за кредит.
- Сохранять данные о платежах.

Задание 2.

Разработать и реализовать программный проект, поддерживающий работу с числами, записанными в различных системах счисления. с основаниями от 2 до 36;

Программа должна позволять пользователю:

- Поддерживать работу с числами, записанными в различных системах счисления с основаниями от 2 до 36;
- Обеспечивать перевод целых и дробных чисел из одной системах счисления в другую;
- Совершать вычислительные операции (сложение, вычитание, умножение, деление)в различных системах счисления;

- Обучать пользователя основным правилам перевода в системах счисления

  из десятичной системы счисления в любую другую и обратно;

  из двоичной системы счисления в восьмеричную и шестнадцатеричную.
- Проверять знания пользователя о переводе чисел в системы счисления.

Задание 3.

Задачей проекта является создание приложения, помогающее ученикам самостоятельно подготовиться к решению одного из заданий ЕГЭ по информатике и ИКТ – задания А13. Необходимо предоставить ученику возможности:

- формирования задания,
- ввода собственного ответа,
- проверки правильности ответа
- получения наглядного объяснения решения.

Задание 4.

Разработать и реализовать программный проект, выполняющий следующие задачи:

- Обеспечение ввода пользователем данных о его результатах ЕГЭ и желаемых ВУЗах/направлениях.
- Оценка шанса на поступление пользователя на каждое доступное направление, исходя из введенных пользователем начальных данных (наименования сданных по ЕГЭ предметов и баллы по ним, наименования интересуемых ВУЗов и интересуемых разделов направлений).

- Отображение результата оценки: таблица с заголовком «Направление — ВУЗ — Проходной балл — требуемые предметы — шанс на поступление», отсортированная по шансу на поступление.

Задание 5.

Разработать и реализовать программный проект, прогнозирующий с некоторой вероятностью проявление тех или иных наследственных признаков.

Предоставить пользователю возможности:

- заполнения генеалогического древа семьи.
- Изменения генетических признаков для каждого его члена семьи.
- Сохранения в БД информации о дереве и признаках, списке доступных признаков и их классификации.

Лабораторные работы

Лабораторная работа №1. Постановка задачи

• Формирование состава проектных групп. Распределение ролей в группе.

• Выбор темы проекта из предложенного списка, либо предложение инициативной темы.

• Анализ предметной области: определение границ предметной области, освоение основных принципов организации деятельности и проблем в данной предметной области.

• Обзор и анализ существующих решений: формирование набора критериев, поиск и классификация существующих решений, вывод о необходимости разработки программной системы.

• Формулировка неформальной постановки задачи.

• Построение диаграммы вариантов использования.

• Документирование результатов работы.

Лабораторная работа №2. Проектные решения.

• Обзор и разработка необходимых математических методов решения и алгоритмов.

• Обоснование выбора средств реализации.

• Формирование общих требований к программному продукту.

• Формирование функциональных требований программной системы.

• Документирование результатов работы.

Лабораторная работа №3. Разработка проекта программной системы

• Проектирование интерфейса программной системы. Соблюдение требований удобства для пользователей.

• Построение диаграммы состояний.

• Разработка архитектуры программной системы. Компоненты системы.

• Документирование результатов работы.

Лабораторная работа №4. Структуры данных программной системы

• Описание внутренних структур данных.

• Построение диаграммы сущностей.

• Описание формата файлов.

• Описание вешнего формата данных.

Лабораторная работа №5. Реализация программной системы

• Разработка плана реализации программной системы.

• Автономная реализация компонент системы.

• Разработка стратегия тестирования программной системы.

• Автономная и комплексная отладка программной системы.

• Экспериментальное тестирование программной системы.

• Технические характеристики системы.

Лабораторная работа №6. Презентация программного продукта.

• Оформление отчета о разработке программной системы.

• Разработка и оформление пользовательской инструкции.

• Разработка устного доклада о разработке программной системы.

• Разработка и создание презентации к устному докладу.

• Защита проекта. Оценка проектов других проектных групп.

функций (параметры функций вводятся пользователем для каждого элемента вектора результата отдельно, при формировании вектора функций).

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Дальневосточный федеральный университет»

(ДВФУ)

<ШКОЛА ЕСТЕСТВЕННЫХ НАУК>

Материалы для организации самостоятельной работы студентов

по дисциплине «Программная инженерия»

Направление— 230700.62 «Прикладная информатика»

г. Владивосток

Лекция 1 Программная инженерия: назначение, основные принципы и понятия [1]

Предпосылки и история

В конце 60-х – начале 70-х годов прошлого века произошло событие, которое вошло в историю как первый кризис программирования. Событие состояло в том, что стоимость программного обеспечения стала приближаться к стоимости аппаратуры («железа»), а динамика роста этих стоимостей позволяла прогнозировать, что к середине 90-годов все человечество будет заниматься разработкой программ для компьютеров. Тогда и заговорили о программной инженерии (или технологии программирования, как это называлось в России) как о некоторой дисциплине, целью которой является сокращение стоимости программ.

С тех пор программная инженерия прошла достаточно бурное развитие. Этапы развития программной инженерии можно выделять по-разному. Каждый этап связан с появлением (или осознанием) очередной проблемы и нахождением путей и способов решения этой проблемы. На слайде представлены ряд фундаментальных проблем разработки программ и найденных фундаментальных методов их решения. Эти методы и по сей день составляют основу подходов к проектированию программных продуктов.

1.1.1. Повторное использование кода (модульное программирование)

Проблема. На первых этапах становления программной инженерии (даже когда она так еще не называлась) было отмечено, что высокая стоимость программ связана с разработкой одинаковых (или похожих) фрагментов кода в различных программах. Вызвано это было тем, что в различных программах как части этих программ решались одинаковые (или похожие) задачи: решение нелинейных уравнений, расчет заработной платы, … Использование при создании новых программ ранее написанных фрагментов сулило существенное снижение сроков и стоимости разработки.

Модульное программирование. Главный принцип модульного программирования состоял в выделении таких фрагментов и оформлении их в виде модулей. Каждый модуль снабжался описанием, в котором устанавливались правила его использования – интерфейс модуля. Интерфейс задавал связи модуля с основной программой – связи по данным и связи по управлению. При этом возможность повторного использования модулей определялась количеством и сложностью этих связей, или насколько эти связи удалось согласовывать с организацией данных и управления основной программы. Наиболее простыми в этом отношении оказались модули решения математических задач: решения уравнений, систем уравнений, задач оптимизации. К настоящему времени накоплены и успешно используются большие библиотеки таких модулей.

Для многих других типов модулей возможность их повторного использования оказалась проблематичной в виду сложности их связей с основной программой. Например, модуль расчета зарплаты, написанный для одной фирмы, может не подойти для другой, т.к. зарплата в этих фирмах рассчитывается не во всем одинаково. Повторное использование модулей со сложными интерфейсами является достаточно актуальной и по сей день. Для ее решения разрабатываются специальные формы (структуры) представления модулей и организации их интерфейсов.

1.1.2. Рост сложности программ (структурное программирование)

Проблема. Следующий этап возрастания стоимости ПО был связан с переходом от разработки относительно простых программ к разработке сложных программных комплексов. К числу таких сложных программ относятся: системы управления космическими объектами, управления оборонным

комплексом, автоматизации крупного финансового учреждения и т.д. Сложность таких комплексов оценивалась следующими показателями:

1. Большой объем кода (миллионы строк)
2. Большое количество связей между элементами кода
3. Большое количество разработчиков (сотни человек)
4. Большое количество пользователей (сотни и тысячи)
5. Длительное время использования

Для таких сложных программ оказалось, что основная часть их стоимости приходится не на создание программ, а на их внедрение и эксплуатацию. По аналогии с промышленной технологией стали говорить о жизненном цикле программного продукта, как о последовательности определенных этапов: этапа проектирования, разработки, тестирования, внедрения и сопровождения.

Структурное программирование. Этап сопровождения программного комплекса включал действия по исправлению ошибок в работе программы и внесению изменений в соответствии с изменившимися требованиями пользователей. Основная причина высокой стоимости (а порой и невозможности выполнения) этапа сопровождения состояла в том, что программы были плохо спроектированы – документация была не понятна и не соответствовала программному коду, а сам программный код был очень сложен и запутан. Нужна технология, которая обеспечит «правильное» проектирование и кодирование. Основные принципы технологии структурного проектирования и кодирования:

2. Нисходящее функциональное проектирование, при котором в системе выделяются основные функциональные подсистемы, которые потом разбиваются на подсистемы и т.д. (принцип «разделяй и властвую»)
3. Применение специальных языков проектирования и средств автоматизации использования этих языков
4. Дисциплина проектирования и разработки: планирование и документирование проекта, поддержка соответствие кода проектной документации
5. Структурное кодирование без goto

1.1.3. Модификация программ (ООП)

Проблема. Следующая проблема роста стоимости программ была вызвана тем, что изменение требований к программе стали возникать не только на стадии сопровождения, но и на стадии проектирования – проблема заказчика, который не знает, что он хочет. Создание программного продукта превратилось в его перманентное перепроектирование. Возник вопрос как проектировать и писать программы, чтобы обеспечить возможность внесений изменений в программу, не меняя ранее написанного кода.

Объектно-ориентированное программирование. Решением этой проблемы стало использование подхода или метода, который стали называть объектно-ориентированным проектированием и программированием. Суть подхода состоит в том, что вводится понятие класса как развитие понятия модуля с определенными свойствами и поведением, характеризующими обязанностями класса. Каждый класс может порождать объекты – экземпляры данного класса. При этом работают основные принципы (парадигмы) ООП:

1. Инкапсуляция – объединение в классе данных (свойств) и методов (процедур обработки).
2. Наследование – возможность вывода нового класса из старого с частичным изменением свойств и методов
3. Полиморфизм – определение свойств и методов объекта по контексту

Проиллюстрировать возможности принципов ООП можно на следующем примере. В организации, состоящей из трех отделов надо начислять заработную плату. В программе каждый отдел представлен своим модулем – объектом, а начисление зарплаты – объектом «Зарплата». При необходимости расчета зарплаты объекту «Отдел» передается экземпляр объекта «Зарплата». Объект

«Отдел» передает объекту «Зарплата» необходимые данные и затем с помощью методов объекта «Зарплата» выполняет необходимые расчеты.

В отделе 3 частично изменились правила начисления зарплаты. В этой ситуации при объектно-ориентированном подходе из класса «Зарплата» выводится класс «Зарплата 1», который наследует неизменившиеся правила начисления зарплаты и переопределяет изменившиеся. Здесь при расчете зарплаты объектам «Отдел 1» и «Отдел 2» будет передаваться объект «Зарплата», а объекту «Отдел 3» - объект «Зарплата 1». При таких изменениях:

1. Срабатывает принцип наследования: код «Зарплата», «Отдел 1» и «Отдел 2» остаются без изменения, а код «Зарплата 1» изменяется ровно настолько, насколько это необходимо.
2. Срабатывает принцип полиморфизма: код «Отдел 3» также не изменяется – он продолжает считать, что работает с объектом «Зарплата»

1.1.4. Некоторые итоги

Программная инженерия (или технология программирования) как некоторое направление возникло и формировалось под давлением роста стоимости создаваемого программного обеспечения. Главная цель этой области знаний - сокращение стоимости и сроков разработки программ.

Программная инженерия прошла несколько этапов развития, в процессе которых были сформулированы фундаментальные принципы и методы разработки программных продуктов. Основной принцип программной инженерии состоит в том, что программы создаются в результате выполнения нескольких взаимосвязанных этапов (анализ требований, проектирование, разработка, внедрение, сопровождение), составляющих жизненный цикл программного продукта. Фундаментальными методами проектирования и разработки являются модульное, структурное и объектно-ориентированное проектирование и программирование.

1.1.5. Продолжение кризиса программирования

Несмотря на то, что программная инженерия достигла определенных успехов, перманентный кризис программирования продолжается. Связано это с тем, рубеж 80–90-х годов отмечается как начало информационно-технологической революции, вызванной взрывным ростом использования информационных средств: персональный компьютер, локальные и глобальные вычислительные сети, мобильная связь, электронная почту, Internet и т.д.

Цена успеха – кризис программирования принимает хронические формы:

- США тратит ежегодно более $200 млрд. на более чем 170 тыс. проектов разработки ПО в сфере IT;
- 31,1% из них закрываются, так и не завершившись; 52,7% проектов завершаются с превышением первоначальных оценок бюджета/сроков и ограниченной функциональностью;
- потери от недополученного эффекта внедрения ПО измеряются триллионами.

Статистика по 30,000 проектам по разработке ПО в американских компаниях показывает следующее распределение между:

- *У*спешными – вовремя и в рамках бюджета был выполнен весь намеченный фронт работ
- Проблемными – нарушение сроков, перерасход бюджета и/или сделали не все, что требовалось
- Проваленными – не были доведены до конца из-за перерасхода средств, бюджета, качества.

Источник: The Standish Group International *The Standish Group International, Inc., Extreme Chaos, 2000* - http://www1.standishgroup.com//sample_research/PDFpages/extreme_chaos.pdfПрограммная инженерия – что это такое?

1.2.1. Начнем с определений

На сегодняшний день нет единого определения понятия «программная инженерия». На слайде приведено несколько таких определений, данных крупными специалистами в этой области, или зафиксированные в документах ведущих организаций.

Сам термин – software engineering (программная инженерия) - впервые был озвучен в октябре 1968 года на конференции подкомитета НАТО по науке и технике (г.Гармиш, Германия). Присутствовало 50 профессиональных разработчиков ПО из 11 стран. Рассматривались проблемы проектирования, разработки, распространения и поддержки программ. Там впервые и прозвучал термин «программная инженерия» как некоторая дисциплина, которую надо создавать и которой надо руководствоваться в решении перечисленных проблем.

Вскоре после этого в Лондоне состоялась встреча 22-х руководителей проектов по разработке ПО. На встрече анализировались проблемы и перспективы развития ПО. Отмечалась возрастающее воздействие ПО на жизнь людей. Впервые серьезно заговорили о надвигающемся кризисе ПО. Применяющиеся принципы и методы разработки ПО требовали постоянного усовершенствования. Именно на этой встрече была предложена концепция жизненного цикла ПО (SLC – Software Lifetime Cycle) как последовательности шагов-стадий, которые необходимо выполнить в процессе создания и эксплуатации ПО. Вокруг этой концепции было много споров. В 1970 г. У.У. Ройс (W.W. Royce) произвел идентификацию нескольких стадий в типичном цикле и было высказано предположение, что контроль выполнения стадий приведет к повышению качества ПО и сокращению стоимости разработки.

1.2.2. Разберемся в вопросах

Для того, чтобы получить представление о том, что такое программная инженерия, попробуем разобраться в следующих вопросах:

- Что такое программное обеспечение (software)?
- Что такое программная инженерия?
- В чем разница между программной инженерией (software engineering) и информатикой (computer science)?
- В чем разница между программной инженерией и системной инженерией (systems engineering)?
- В чем отличие программной инженерии от других инженерий?

1.2.3. Что такое программное обеспечение (software)?

Программное обеспечение это набор компьютерных программ, процедур и связанной с ними документации и данных (ISO/IEC 12207). Взгляд на ПО как только на программу, сидящую в компьютере слишком узок. Дело в том, что продается (поставляется) не только программа, но еще и документация, в которой можно прочитать как установить программу и как ей пользоваться и данные для установки программы в различных условиях (конфигурационные файлы). Поэтому ПО иногда называют программным продуктом. Т.е. программный продукт (программное обеспечение) – это не только программы, а также вся связанная с ними документация и конфигурационные данные, необходимые для корректной работы программы. А специалисты по программному обеспечению разрабатывают программные продукты, т.е. такое ПО, которое может быть продано потребителю.

В зависимости от того, для кого разрабатываются программные продукты (конкретного заказчика или рынка, программные продукты бывают двух типов:

- коробочные продукты (generic products – общие продукты или shrink-wrapped software – упакованное ПО)

— заказные продукты (bespoke – сделанный на заказ или customized products – настроенный продукт). Важная разница между ними заключается в том, кто ставит задачу (определяет, или специфицирует требования). В первом случае это делают сами разработчики на основе анализа рынка (маркетинга) – и при этом рискуют сами. Во втором – заказчик и при этом рискует, что разработчик не сможет реально выполнить все требования в срок и при выделенном бюджете.

1.2.4. Что такое программная инженерия?

Программная инженерия — это инженерная дисциплина, которая связана со всеми аспектами производства ПО от начальных стадий создания спецификации до поддержки системы после сдачи в эксплуатацию. В этом определении есть две ключевые фразы:

- Инженерная дисциплина
- Все аспекты производства ПО

Инженерная дисциплина. Инженеры – это те специалисты, которые выполняют практическую работу и добиваются практических результатов. Ученый может сказать: проблема неразрешима в рамках существующих теорий и это будет научный результат, достойный опубликования и защиты диссертации.

Для решения задачи инженеры применяют теории, методы и средства, пригодные для решения данной задачи, но они применяют их выборочно и всегда пытаются найти решения, даже в тех случаях, когда теорий или методов, соответствующих данной задаче, еще не существует. В этом случае инженер ищет метод или средство для решения задачи, применяет его и несет ответственность за результат – ведь метод или средство еще не проверены. Набор таких инженерных методов или способов, теоретически возможно не обоснованных, но получивших неоднократное подтверждение на практике, играет большую практическую роль. В программной инженерии они получили название лучших практик (best practices).

Инженеры работают в условиях ограниченных ресурсов: временных, финансовых и организационных (оборудование, техника, люди). Иными словами, продукт должен быть создан в установленные сроки, в рамках выделенных средств, оборудования и людей. Хотя это в первую очередь относится к созданию заказных продуктов (оговаривается в условиях контракта), но при создании коробочных продуктов эти ограничения имеют не меньшее значение, т.к. здесь они диктуются условиями рыночной конкуренции.

Все аспекты производства ПО. Программная инженерия занимается не только техническими вопросами производства ПО (специфицирование требований, проектирование, кодирование,…), но и управлением программными проектами, включая вопросы планирования, финансирования, управления коллективом и т.д. Кроме того, задачей программной инженерии является разработка средств, методов и теорий для поддержки процесса производства ПО.

Программные инженеры применяют систематичные и организованные подходы к работе для достижения максимальной эффективности и качества ПО. Их задача состоит в адаптации существующих методов и подходов к решению свой конкретной проблемы.

1.2.5. В чем отличия от информатики?

Информатика (computer science) занимается теорией и методами вычислительных и программных систем, в то время как программная инженерия занимается практическими проблемами создания ПО. Информатика составляет теоретические основы программной инженерии и инженер по программному обеспечению должен знать информатику. Так же, как инженер по электронике должен знать физику. В идеале, программная инженерия должна быть поддержана какими-то теориями информатики, но самом деле это не всегда так. Программные инженеры зачастую

используют приемы, которые применимы только в конкретных условиях и не могут быть обобщены на другие случаи, а элегантные теории информатики не всегда могут быть применены к реальным большим системам.

И наконец, информатика – это не единственный теоретический фундамент программной инженерии, т.к. круг проблем, стоящих перед программным инженером значительно шире просто написания программ. Это еще управление финансами, организация работ в коллективе, взаимодействие с заказчиком и т.д. Решение этих проблем требуют фундаментальных знаний, выходящих за рамки информатики.

1.2.6. В чем отличие от других инженерий?

Отличие программной инженерии от других инженерий интересно прежде всего с точки зрения двух вопросов:

- Почему доля провальных проектов в программной инженерии так велика по сравнению с другими инженериями?
- Можно ли в программной инженерии применять опыт других инженерий?

Эти вопросы является фундаментальными для программной инженерии. По этому поводу высказывается много мнений (и часто противоположных). Остановимся на некоторых более или менее очевидных отличиях программной инженерии от других инженерий.

Прежде всего, отметим, что жизненный цикл продукта любой инженерии в упрощенном виде включает фазы: проектирование, создание образца, испытание, производство, эксплуатация.

Компьютерная программа – это (в отличие от объектов других инженерий) не материальный объект (просьба не путать с носителем программы – устройством памяти любого типа). Отсюда следуют следующие отличия. Фаза производства состоит в копировании образца на другие носители. Стоимость фазы исчезающее мала. Если кодирование считать элементом проектирования (что очень близко к истине), то отсутствует также и фаза создания образца (строится компилятором и линковщиком)

Отсюда следуют следующие выводы:

- Стоимость программы – это стоимость только ее проектирования
- Стоимость проектирования коробочных продуктов «размазывается» по копиям
- Стоимость заказных продуктов (массово не копируемых) остается высокой

1.2.7. В чем еще отличие от других инженерий?

Второе существенное отличие состоит в том, что программа – искусственный объект. Т.е. для программы нет объективных законов, которым бы подчинялось ее поведение. Например, у инженера – строителя есть объективные законы строительной механики: равновесия моментов и сил, устойчивости механических систем и т.д. Инженер – строитель может проверить свои архитектурные решения на соответствие этим законам и тем самым обеспечить удачу проекта. Эти законы объективны, они будут действовать всегда. У программного инженера на первый взгляд также есть типовые, проверенные временем архитектурные решения (например, клиент-серверная архитектура). Но эти решения определяются уровнем развития вычислительной техники (и адекватным им уровнем требований). С появлением техники с принципиально новыми возможностями программному инженеру придется искать новые решения.

Прямым следствием отсутствия возможности «теоретического» контроля проекта является то, что тестирование продукта – это единственный способ убедиться в его качестве. Именно поэтому стоимость тестирования составляет существенную стоимость ПО. Кстати, строительный инженер,

как правило, лишен возможности такого «тестирования» своего продукта перед сдачей его в эксплуатацию.

Ну и наконец, программная инженерия – молодая дисциплина, опыт которой насчитывает всего несколько десятков лет. По сравнению с опытом строительной инженерии (тысячелетия) это конечно очень мало. Программную инженерию иногда сравнивают с ранней строительной, когда законы строительной механики еще не были известны и строительные инженеры действовали методом проб и ошибок, накапливая бесценный опыт. Несмотря на молодой возраст, программная инженерия также накопила определенный опыт, который позволяет (при разумном его применении) делать удачные проекты. Этот опыт выражен в основных принципах программной инженерии, которые мы с вами сейчас рассмотрим.

Подробнее о проблемах проектирования ПО можно посмотреть в неоднозначной статье Кони Бюрера «От ремесла к науке: поиск основных принципов разработки ПО»
http://interface.ru/fset.asp?Url=/rational/science.htm

1.2.8. Из чего складывается стоимость ПО?

Структура стоимости ПО существенно зависит от типа ПО, применяемых методов его разработки и … метода оценки. Так, многие авторы отмечают высокую долю стоимости этапа сопровождения. Для некоторых типов ПО она может составлять 60 и более процентов от общей стоимости. Между тем, этап сопровождения включает выполнение двух видов работ: исправление ошибок в программе (несоответствий первоначальным требованиям) и внесение изменений в программу (добавление новых требований). При другом подходе к оценке можно считать, что этап сопровождения на стоит оценивать отдельно, т.к. исправление ошибок можно отнести к продолжению тестирования, а внесение изменений – к новому проекту.

Типовое распределение стоимости между основными этапами (без сопровождения) выглядит следующим образом:

- 15% - спецификация – формулировка требований и условий разработки
- 25% - проектирование – разработка и верификация проекта
- 20% - разработка – кодирование и тестирование компонент
- 40% - интеграция и тестирование – объединение и сборочное тестирование продукта

Отклонения от этой схемы в зависимости от типа ПО выглядят следующим образом:

Для коробочного ПО характерна более высокая доля тестирования за счет сокращения прежде всего доли спецификации (до 5%)

Распределение стоимости заказного ПО зависит от его сложности. При сложном ПО также возрастает доля интеграции и тестирования, но за счет сокращения доли проектирования и разработки Доля спецификаций может возрастать. Сокращение доли проектирования и разработки достигается за счет применения опробованных проектных решений и повторного использования готовых компонент.

Применение опробованных решений и готовых компонент при создании коробочных продуктов позволяет повысить качество и сократить сроки разработки.

1.2.9. Еще вопросы

Для того, чтобы получить представление о том, в чем состоит накопленный программной инженерией опыт, попробуем разобраться в следующих вопросах:

- Что такое программный процесс?
- Что такое модель программного процесса?

- Что такое методы программной инженерии?
- Что такое CASE (Computer-Aided Software Engineering)?
- Какими свойствами обладает хорошая программа?
- Какие основные трудности стоят перед программной инженерией?

1.2.10. Программный процесс?

Одним из основных понятий программной инженерии является понятие жизненного цикла программного продукта и программного процесса.

Жизненный цикл – непрерывный процесс, начинающийся с момента принятия решения о создании ПО и заканчивающийся снятием его с эксплуатации. Жизненный цикл разбивается на отдельные процессы.

Процесс – совокупность действий и задач, имеющих целью достижение значимого результата. Основными процессами (иногда называют этапами или фазами) жизненного цикла являются:

- Разработка спецификации требований (результат – описания требований к программе, которые обязательны для выполнения – описание того, что программа должна делать)
- Разработка проекта программы (результат – описание того, как программа будет работать)
- Кодирование (результат – исходный код и файлы конфигурации)
- Тестирование программы (результат - контроль соответствия программы требованиям)
- Документирование (результат – документация к программе)

Кроме основных, существует много дополнительных и вспомогательных процессов, связанных не созданием продукта, а с организацией работ (нефункциональные процессы): создание инфраструктуры, управление конфигурацией, управление качеством, обучение, разрешение противоречий, …

Процесс должен быть установлен. Полное установление процесса предполагает:

- Описание процесса – детальное описание действий и операций процесса
- Обучение процессу – проведение занятий с персоналом по освоению процесса
- Введение метрик – установление количественных показателей хода выполнения
- Контроль выполнения – измерение метрических показателей и оценка хода выполнения
- Усовершенствование – изменение процесса при меняющихся условиях применения

Применение полных (тяжелых) процессов требует дополнительных ресурсов (зачастую существенных) и далеко не всегда окупается полученным результатом. Поэтому, выбор состава процессов, степени их установленности (полноты установленности) в каждом конкретном случае может быть сделан по-разному в соответствии с выбранной моделью процесса.

1.2.11. Модель программного процесса?

Модель программного процесса — это упрощенное описание программного процесса, представленное с некоторой точки зрения. Модель задается в виде практических этапов, необходимых для создания ПО. В модели мы говорим, что и как мы будем делать. Т.е. какие процессы, с какой степенью конкретизации и в какой последовательности мы будем выполнять. Выбор модели процесса является первым шагом в создании ПО. Правильны выбор модели очень важен, т.к. во многом определяет успех проекта. Выбор тяжелых процессов может утопить проект, а слишком легкое отношение к процессам – к потере контроля над ходом выполнения.

В соответствии с двумя типами процессов – основных и дополнительных - можно говорить о двух типах моделей процесса: модели процесса разработки (модели жизненного цикла) и модели организации работ по выполнению разработки.

К первым типам моделей (модели жизненного цикла) относятся модели, в которых описывается порядок выполнения действий по созданию продукта. К наиболее известным моделям относятся:

- Водопадная (каскадная) модель – процесс разбивается но последовательное выполнение стадий; каждая стадия начинается после полного завершения предыдущей, продукт создается завершением последней стадии и должен полностью соответствовать изначально установленным требованиям.
- Спиральная (циклическая) модель – процесс также разбивается на стадии, но стадии выполняются циклическим повторением. На первом цикле создается прототип продукта, выполняющий часть требований. Дальнейшие циклы связаны с наращиванием прототипа до полного удовлетворения требований.
- Компонентная модель предполагает сборку продукта из заранее написанных частей – компонент. Основной упор делается на интеграцию и совместное тестирование компонент.
- Формальная модель основана на формальном описании требований с последующим преобразованием (трансляцией) требований в исходный код. Применение формальной модели гарантирует соответствие кода описанным требованиям.

Следует отметить, что различия между этими моделями существуют только в теории. На практике спиральная модель может быть дополнена элементами каскадной и компонентной. Задача программного инженера – подобрать правильную их комбинацию, ориентируясь только на конечный результат.

Ко второму типу моделей – моделей организации работ относятся:

- Модель потока работ (workflow model) — показывает последовательность действий, выполняемых людьми на различных этапах разработки ПО. Для каждого действия указываются входы, выходы (результаты) и связи по входам и выходам.
- Модель потоков данных (data flow model) — представляет процесс в виде последовательного преобразования данных. Каждое преобразование может выполняться одним или несколькими действиями.
- Ролевая модель — показывает роли людей, участвующих в программном процессе, а также действия и результаты, за которые они отвечают.

1.2.12. Методы программной инженерии?

Метод программной инженерии — это структурный подход к созданию ПО, который способствует производству высококачественного продукта эффективным в экономическом аспекте способом. В этом определении есть две основные составляющие: (а) создание высококачественного продукта и (б) экономически эффективным способом. Иными словами, метод – это то, что обеспечивает решение основной задачи программной инженерии: создание качественного продукта при заданных ресурсах времени, бюджета, оборудования, людей.

Начиная с 70-х годов создано достаточно много методов разработки ПО. Наиболее известны:

Метод структурного анализа и проектирования Том ДеМарко (1978),
Метод сущность-связь проектирования информационных систем Чен (1976)
Метод объектно-ориентированного анализа Буч (1994), Рамбо (1991).
Метод программной индустрии основан на идее создания моделей ПО с поэтапным преобразованием этих моделей в программу – окончательную модель решаемой задачи. Так, на этапе спецификаций создается модель – описание требований, которая далее преобразуется в модель проекта ПО, проект –

в программный код. При этом важно, чтобы модели метода представлялись графически с помощью некоторого языка представления моделей.

Методы должны включать в себя следующие компоненты:

- Описание моделей системы и нотация, используемая для описания этих моделей (например, объектные модели, конечно-автоматные модели и т.д.)
- Правила и ограничения, которые надо выполнять при разработке моделей (например, каждай объект должен иметь одинаковое имя)
- Рекомендации — эвристики, характеризующие хорошие приемы проектирования в данном методе (скажем, рекомендация о том, что ни у одного объекта не должно быть больше семи подобъектов)
- Руководство по применению метода — описание последовательности работ (действий), которые надо выполнить для построения моделей (все атрибуты должны быть задокументированы до определения операций, связанных с этим объектом)

Нет идеальных методов, все они применимы только для тех или иных случаев. Нет абсолютных методов – применяемые на практике методы могут включать элементы различных подходов. Выбор метода составляет задачу специалиста по программной инженерии.

1.2.12.1. Модель прецедентов (требований)

На слайде представлена модель прецедентов, или вариантов использования (Use case) в нотации языка UML (Unified Modeling Language), поддерживающего объектно-ориентированный анализ и проектирование ПО. Модель описывает (специфицирует) требования к программе регистрации курсов в ВУЗе.

На диаграмме представлены действующие лица (акторы) и действия (прецеденты), которые они выполняют:

- Студент регистрируется на курсе
- Преподаватель: выбирает курс для преподавания и запрашивает расписание курсов

Для каждого действия – прецедента дается его содержательное описание. Далее на основе этой модели строится путем поэтапного ее уточнения и преобразования будут строиться другие модели программы.

1.2.12.2. Модель классов

На слайде представлена одна из таких моделей – диаграмма классов. На диаграмме показаны основные классы программы: ВУЗ, факультет, Студент, Курс, Преподаватель. Приведены основные атрибуты и методы классов.

Кроме того, отмечены отношения (зависимости) между классами:

- Так отмечено, что ВУЗ состоит из Факультетов (агрегация), причем, один ВУЗ может состоять из одного или нескольких факультетов.
- Каждый преподаватель работает на факультете, но при этом, каждый преподаватель может работать на нескольких факультетах и на каждом факультете могут работать несколько преподавателей.

1.2.12.3. Модель сущность-связь

На слайде представлена модель сущность-связь для задачи автоматизации продаж товаров со склада. Представлены сущности, участвующие в процессе: Покупатель, Накладная, Список товаров, Склад, … Для каждой сущности представлены ее атрибуты – для покупателя это код покупателя, имя, адрес, банк. Выделены ключевые атрибуты, однозначно идентифицирующие экземпляры сущностей (у покупателя это код). Установлены связи между сущностями: Покупатель получает Накладную. Отмечены свойства связей: один покупатель может получать несколько накладных.

Далее эта модель преобразуется в реляционную модель базы данных для хранения информации о выделенных сущностях.

Представленная на слайде модель записана в нотации IE (Information Engineering). В других нотациях она будет выглядеть иначе.

1.2.12.4. Нотации модели

На слайде представлен фрагмент модели сущность-связь задачи учета сотрудников, записанный в различных нотациях: Чена (автор метода сущность-связь), Мартина (соавтор), стандарта IDEF1X (Information Modeling Method) и Баркера.

1.2.13. Что такое CASE?

CASE - Computer Aided System Engineering - различного рода инструментальные программы, используемые для поддержки процесса создания программ

CASE средства могут быть классифицированы по нескольким признакам:

- По уровню применения:
  - Upper CASE -средства анализа требований
  - Middle CASE - средства проектирования
  - Low CASE - ссредства разработки приложений
- Специализированные
  - Средства проектирования баз данных
  - Средства реинжиниринга (восстановления) модели (формирование ERD на основе анализа схем БД или формирования диаграмм на основе анализа программных кодов)
- Вспомогательные
  - Планирования и управления проектом
  - Конфигурационного управления
  - Тестирования
- Интегрированные CASE охватывают все этапы и процессы создания ПО от анализа требований до тестирования и выпуска документации. Интегрированные CASE выступают, как правило, в виде набора согласованных по интерфейсу средств, предназначенных для поддержки отдельных этапов процесса.

В настоящее время существует очень много CASE средств и фирм, специализирующихся на их разработке (Rational). При выборе CASE средств следует руководствоваться основным принципом: сначала метод создания ПО, а потом – CASE средства, применимые для этого метода. Выбор наоборот чреват нехорошими последствиями.

Computer Aided System Engineering - использование компьютеров для поддержки процесса создания программ.

Это широкий спектр программ – инструментальных средств, применяемых на разных этапах разработки ПО: спецификации требований, проектирования, кодирования, тестирования, документирования, …

1.2.14. Свойства хорошей программы?

Удовлетворять функцио-нальным требованиям. Прежде всего, хорошая программа должна делать то, что ожидает от нее заказчик – т.е. удовлетворять требованиям заказчика. Такие требования называют функциональными. Но кроме функциональных требований, существует еще ряд общих характеристик, которым в той или иной степени должна обладать каждая программа. Эти характеристики принято называть нефункциональными требованиями. К нефункциональным требованиям относят:

- Сопровождаемость (maintainability). Сопровождаемость означает, что программа должна быть написана с расчетом на дальнейшее развитие. Это критическое свойство системы, т.к. изменения ПО неизбежны вследствие изменения бизнеса. Сопровождение программы выполняют, как правило, не те люди, которые ее разрабатывали. Сопровождаемость включает такие элементы как наличие и понятность проектной документации, соответствие проектной документации исходному коду, понятность исходного кода, простота изменений исходного кода, простота добавления новых функций.
- Надежность (dependability). Надежность ПО включает такие элементы как:
  - Отказоустойчивость – возможность восстановления программы и данных в случае сбоев в работе
  - Безопасность – сбои в работе программы не должны приводить к опасным последствиям (авариям)
  - Защищенность от случайных или преднамеренных внешних воздействий (защита от дурака, вирусов, спама).
- Эффективность (efficiency). ПО не должно впустую тратить системные ресурсы, такие как память, процессорное время, каналы связи. Поэтому эффективность ПО оценивается следующими показателями: время выполнения кода, загруженность процессора, объем требуемой памяти, время отклика и т.п.
- Удобство использования (usability). ПО должно быть легким в использовании, причем именно тем типом пользователей, на которых рассчитано приложение. Это включает в себя интерфейс пользователя и адекватную документацию. Причем, пользовательский интерфейс должен быть не интуитивно, а профессионально понятным пользователю.

Следует отметить, что реализация нефункциональных требований часто требует больших затрат, чем функциональных. Так, сопровождаемость требует значительных усилий по поддержанию соответствия проекта исходному коду и применения специальных методов создания модифицируемых программ. Надежность – дополнительных средств восстановления системы при сбоях. Эффективность – поиска специальных архитектурных решений и оптимизации кода. А удобство – проектирования не «интуитивно» понятного, а профессионально понятного интерфейса пользователя.

1.2.15. Основные трудности

Трудностей достаточно много. Все они в той или иной степенью связаны с главной проблемой программной инженерии: поиском универсального метода и процесса, пригодного для создания программного продукта любого типа в любых условиях. Здесь главная проблема – меняющиеся условия. В этой связи разработчики ПО сталкиваются со следующими трудностями:

- Наследование ранее созданного ПО (legacy systems). Существует достаточно много систем, созданных много лет назад, морально устаревших, но продолжающих работать. Проблема наследования таких систем состоит в их сопровождении – поддержке и развитии.
- Разнородность программных систем. ПО должно работать в распределенных сетях, на разнородном оборудовании, в разных средах, под управлением различных операционных систем.

- Сокращение времени на разработку. Запросы рынка требования к программным системам меняются очень быстро. Суть проблемы состоит в том, чтобы сократить время разработки ПО без снижения его качества.

Эти трудности часто оказываются связанными между собой. Задача разработки сетевого варианта старой локальной базы данных в ограниченные сроки является достаточно типичной.

1.2.16. Профессинальные и этические требования

Развитие средств вычислительной техники, коммуникаций и программных систем (Internet, телекоммуникации, распределенные системы, IP телефония, компьютерные игры и обучающие программы) оказывает все большее воздействие на общество. Роль специалистов по программному обеспечению при этом все время возрастает. Они работают в определенном правовом и социальном окружении, находятся под действием, международных, национальных и местных законодательств.

Ясно, что программисты, как и специалисты других профессий, должны быть честными и порядочными людьми. Но вместе с тем, программисты не могут руководствоваться только моральными нормами или юридическими ограничениями, т.к. они обычно бывают связаны более тонкими профессиональными обязательствами:

- Конфиденциальность – программные специалисты должны уважать конфиденциальность в отношении своих работодателей или заказчиков независимо от того, подписывалось ли ими соответствующее соглашение.
- Компетентность – программный специалист не должен завышать свой истинный уровень компетентности и не должен сознательно браться за работу, которая этому уровню не соответствует.
- Защита интеллектуальной собственности – специалист должен соблюдать законодательство и принципы защиты интеллектуальной собственности при использовании чужой интеллектуальной собственности. Кроме того, он должен защищать интеллектуальную собственность работодателя и клиента. Внимание: создаваемая им интеллектуальная собственность является собственностью работодателя или клиента.
- Злоупотребление компьютером – программный специалист не должны злоупотреблять компьютерными ресурсами работодателя или заказчика; под злоупотреблениями мы здесь понимаем широкий спектр — от игр в компьютерные игрушки на рабочем месте до распространения вирусов и т.п.

1.2.16.1. Кодекс этики IEEE-CS/ACM

В разработке таких этических обязательств ведущую роль играют профессиональные сообщества. Такие общества, как

- ACM – Association for Computing Machinery - Ассоцтация по вычислительной технике,
- IEEE – Institute of Electrical and Electronic Engineers – Институт инженеров по электротехнике и электронике
- CS - British Computer Society – Британское компьютерное общество

совместно разработали и опубликовали IEEE-CS/ACM Software Engineering Code of Ethics and Professional Practices – Кодекс этики и профессиональной практики программной инженерии..

Члены этих организация принимают обязательство следовать этому кодексу в момент вступления в организацию

Кодекс содержит восемь Принципов, связанных с поведением и решениями, принимаемыми профессиональными программистами, включая практиков, преподавателей, менеджеров и руководителей высшего звена

Кодекс распространяется также на студентов и «подмастерьев», изучающих данную профессию

Кодекс имеет краткую и полную версии

1.2.16.2. Кодекс этики - Преамбула

Краткая версия кодекса

- суммирует стремления кодекса на высоком уровне абстракции.
- полная версия показывает как эти стремления отражаются на деятельности профессиональных программистов.
- без высших принципов детали кодекса станут казуистическими и нудными;
- без деталей стремления останутся возвышенными, но пустыми;
- вместе же они образуют целостный кодекс.

Программные инженеры должны добиваться, чтобы анализ, спецификация, проектирование, разработка, тестирование и сопровождение программного обеспечения стали полезной и уважаемой профессией. В соответствии с их приверженностью к процветанию, безопасности и благополучию общества, программные инженеры будут руководствоваться следующими Восемью Принципами

1.2.16.3. Кодекс этики: 8 принципов

1. ОБЩЕСТВО

- Программные инженеры будут действовать соответственно общественным интересам.

2. КЛИЕНТ И РАБОТОДАТЕЛЬ

- Программные инженеры будут действовать в интересах клиентов и работодателя, соответственно общественным интересам.

3. ПРОДУКТ

- Программные инженеры будут добиваться, чтобы произведенные ими продукты и их модификации соответствовал высочайшим профессиональным стандартам.

4. СУЖДЕНИЕ

- Программные инженеры будут добиваться честности и независимости в своих профессиональных суждениях

5. МЕНЕДЖМЕНТ

1. Менеджеры и лидеры программных инженеров будут руководствоваться этическим подходом к руководству разработкой и сопровождением ПО, а также будут продвигать и развивать этот подход

6. ПРОФЕССИЯ

- Программные инженеры будут улучшать целостность и репутацию своей профессии соответственно с интересами общества

4. КОЛЛЕГИ

- Программные инженеры будут честными по отношению к своим коллегам и будут всячески их поддерживать

8. ЛИЧНОСТЬ

- Программные инженеры в течение всей своей жизни будут учиться практике своей профессии и будут продвигать этический подход к практике своей профессии

Полная версия кодекса: IEEE-CS/ACM Software Engineering Ethics and Professional Practices. http://www.computer.org/tab/seprof/code.htm#Public

Литературные источники:

1. Программная инженерия: назначение, основные принципы и понятия

Версия издания 0.1.1, 24 ноября 2004 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Дальневосточный федеральный университет»

(ДВФУ)

<ШКОЛА ЕСТЕСТВЕННЫХ НАУК>

КОНТРОЛЬНО-ИЗМЕРИТЕЛЬНЫЕ МАТЕРИАЛЫ

по дисциплине «Программная инженерия»

Направление— 230700.62 «Прикладная информатика»

г. Владивосток

Самостоятельная работа

№	Тема самостоятельной работы	Часы
1	Описание предметной области. Обзор и анализ существующих решений.	20
2	Формирование функциональных требований.	10
3	Обзор математических методов решения и разработка алгоритмов.	50
4	Проектирование и реализация интерфейса.	30
5	Реализация проекта ПС.	60
6	Динамический контроль поведения программы на множестве тестов. Пользовательское тестирование.	10
	Всего за семестр	180

Коллоквиум

Типы вопросов
1. Правила выбора тестов из бесконечной области. 2. Мероприятия, необходимые для обеспечения экономически эффективной поддержки программного обеспечения. 3. Сохранение целостности и прослеживаемости конфигурации на протяжении всего жизненного цикла системы. 4. Этапы создания программной системы.

Контрольная работа

Тема: «Оценка дружественности интерфейса ПС»

Теоретические вопросы к экзамену:

- Проведение научных исследований программных продуктов, проектов, процессов, методов и инструментов программной инженерии.
- Построение моделей программных проектов программных продуктов.
- Инструментальные средства компьютерного моделирования.

- Подготовка данных для составления обзоров и отчетов.
- Сбор и анализ требований заказчика к программному продукту
- Оценка выбора вариантов программного обеспечения.
- Подготовка коммерческого предложения заказчику, презентация.
- Проектирование компонент программного продукта.
- Создание компонент программного обеспечения (кодирование, отладка, модульное и интеграционное тестирование)
- Измерение кода в соответствии с планом
- Разработка тестового окружения и создание тестовых сценариев
- Разработка и оформление эскизной, технической и рабочей проектной документации
- Средства автоматизированного проектирования, разработки, тестирования и сопровождения программного обеспечения
- Методы управления инженерной деятельностью и процессами жизненного цикла программного обеспечения
- Контроль, оценка и обеспечение качества программной продукции
- Процесс разработки программного обеспечения
- Создание технической документации по результатам выполнения работ
- Обучение пользователей программных систем
- Сопровождение программного продукта
- Организация работы малых коллективов исполнителей программного проекта

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Дальневосточный федеральный университет»

(ДВФУ)

<ШКОЛА ЕСТЕСТВЕННЫХ НАУК>

ГЛОССАРИЙ

по дисциплине «Программная инженерия»

Направление— 230700.62 «Прикладная информатика»

г. Владивосток

Архитектура — организационная структура системы, включающая в себя разделение системы на части, связи между этими частями, механизмы взаимодействия и основные принципы проектирования системы.

Действие (action) — выполнимое атомарное вычисление, которое приводит к изменению состояния системы иои возврату значения.

Конструирование — фаза процесса разработки программного обеспечения, в течение которой проводится подробное проектирование, а система реализуется в коде и тестируется.

Исключение — сигнал, возникающий с помощью механизмов исключений в ответ на дефекты поведения.

Реализация — стадия разработки системы, в которой описывается функционирование системы в некой среде (например, языке программирования, базе данных или цифровом аппаратном обеспечении).

Модуль — программный блок, служащий для хранения и управления.

Интерфейс — именованное множество операций, описывающих поведение элемента.

Канал связи — (*channel, data line*) система технических средств и среда распространения сигналов для односторонней передачи данных от источника к получателю.