МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра *вычислительной математики и кибернетики*

УТВЕРЖДАЮ

Проректор по учебной работе

_____Н.Г. Зарипов

«02» 09 2015 г.

РАБОЧАЯ ПРОГРАММА

УЧЕБНОЙ ДИСЦИПЛИНЫ

«*Технологии разработки программного обеспечения*»

Уровень подготовки: высшее образование – подготовка кадров высшей квалификации

Направление подготовки научно-педагогических кадров высшей квалификации (аспирантура)
09.06.01 Информатика и вычислительная техника
(код и наименование направления подготовки)

Направленность подготовки
Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей
(наименование программы подготовки)

Квалификация (степень) выпускника
Исследователь. Преподаватель исследователь

Форма обучения
очная

Уфа 2015

Содержание

1. Место дисциплины в структуре образовательной программы

Дисциплина *Технологии разработки программного обеспечения* является дисциплиной*вариативной* части.

Рабочая программа составлена в соответствии с требованиями Федерального государственного образовательного стандарта высшего образования по направлению подготовки научно-педагогических кадров высшей квалификации (аспирантура) *09.06.01 Информатика и вычислительная техника*, утвержденного приказом Министерства образования и науки Российской Федерации от "30" июля 2014 г. № 875 и приказа Министерства образования и науки Российской Федерации от 30.04.2015 N 464 "О внесении изменений в федеральные государственные образовательные стандарты высшего образования (уровень подготовки кадров высшей квалификации)". Является неотъемлемой частью основной образовательной профессиональной программы (ОПОП).

Целью освоения дисциплины является развитие у аспирантов личностных качеств и формирование общепрофессиональных и профессиональных компетенций в соответствии с ФГОС ВОпо направлению подготовки 09.06.01 Информатика и вычислительная техника; изучение основ проектно-ориентированного управления, методов современного наукоемкого менеджмента.

Задачи: углубленное изучение теоретических и методологических основ проектирования, эксплуатации и развития информатики и вычислительной техники; формирование навыков в области применения инструментария моделирования и планирования в процессе управления жизненным циклом приложений.

Входные компетенции:

№	Компетенция	Код	Уровень освоения, определяемый этапом формирования компетенции*	Название дисциплины (модуля), практики, научных исследований, сформировавших данную компетенцию
1.	Владение методологией теоретических и экспериментальных исследований в области профессиональной деятельности;	ОПК1	базовый	Модуль: Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей
1.	Владение культурой научного исследования, в том числе с использованием современных информационно-коммуникационных технологий;	ОПК2	пороговый	На предыдущем уровне высшего образования (специалитет, магистратура)
3.	Способность разрабатывать математическое обеспечение в виде математических моделей объектов, процессов и систем различного типа и современных математических методов, включая методы с применением элементов искусственного интеллекта и его реализация	ПК1	базовый	Модуль: Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей
4.	Способность создавать,	ПК2	базовый	Модуль: Математическое и

	унифицировать и оптимизировать программный код с целью повышения эффективности процессов обработки данных и знаний			программное обеспечение вычислительных машин, комплексов и компьютерных сетей
5.	Способность анализировать качество, надежность программного обеспечения и его соответствия требованиям, спецификациям и стандартам	ПК3	базовый	Модуль: Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей
6.	Способность проектировать и анализировать архитектуру программных систем	ПК4	базовый	Модуль: Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

*- ***пороговый уровень*** *дает общее представление о виде деятельности, основных закономерностях функционирования объектов профессиональной деятельности, методов и алгоритмов решения практических задач;*

*-****базовый уровень*** *позволяет решать типовые задачи, принимать профессиональные и управленческие решения по известным алгоритмам, правилам и методикам;*

*-****повышенный уровень*** *предполагает готовность решать практические задачи повышенной сложности, нетиповые задачи, принимать профессиональные и управленческие решения в условиях неполной определенности, при недостаточном документальном, нормативном и методическом обеспечении.*

Исходящие компетенции:

№	Компетенция	Код	Уровень освоения, определяемый этапом формирования компетенции	Название дисциплины (модуля), практики, научных исследований для которых данная компетенция является входной
1	Владение методологией теоретических и экспериментальных исследований в области профессиональной деятельности;	ОПК1	повышенный	Научно-исследовательская практика
2	Владение культурой научного исследования, в том числе с использованием современных информационно-коммуникационных технологий;	ОПК2	повышенный	Блок 4. ГИА
3	Способность создавать, унифицировать и оптимизировать программный код и с целью повышения эффективности процессов обработки данных и знаний	ПК2	повышенный	Научные исследования
4	Способность анализировать качество, надежность программного обеспечения и его соответствия требованиям,	ПК3	повышенный	Научные исследования

	спецификациям и стандартам			
5	Способность проектировать и анализировать архитектуру программных систем	ПК4	повышенный	Научные исследования

2. Перечень результатов обучения

Процесс изучения дисциплины направлен на формирование элементов следующих компетенций.

Планируемые результаты обучения по дисциплине

№	Формируемые компетенции	Код	Знать	Уметь	Владеть
1	владением методологией теоретических и экспериментальных исследований в области профессиональной деятельности;	ОПК1	цели и задачи исследования, основные методологические подходы исследования процессов функционирования объектов профессиональной деятельности; общие принципы и закономерности в построении, функционировании и развитии, управлении и моделировании процессов объектов исследования	использовать методологии и методы научного исследования на уровнях теоретического познания и эмпирического исследования, использования общелогических методов и приемов исследования;	системными правилами выявления причин нарушения системных принципов функционирования объектов исследования
2	владением культурой научного исследования, в том числе с использованием современных информационно-коммуникационных технологий;	ОПК2	социально-культурное содержание деятельности исследователя; основные этапы решения научных и прикладных задач на ЭВМ;	решать задачи обработки информации с помощью современных инструментальных средств и информационно-коммуникационных технологий;	современными информационно-коммуникационными технологиями для организации своего труда.
3	способность создавать, унифицировать и оптимизировать программный код и с целью повышения эффективности процессов обработки данных и знаний	ПК2	методов и средств эффективной разработки программного кода; особенностей программирования обмена с окружающей средой; приемы и средства унификации программ с сохранением опыта разработки путем документирования	проектировать программное обеспечение с использования специализированных программных пакетов;	владеть методами и инструментами анализа и проектирования программного обеспечения;
4	способность анализировать качество, надежность программного обеспечения и его соответствия требованиям,	ПК3	методов и технологий тестирования кода; основных принципов управления качеством программного обеспечения; средств верификации программного	использовать методы и технологии тестирования кода; оценивать качество программного кода; оценивать программный код на соответствие стандартам	владеть методами и инструментами анализа и проектирования; использовать методы и технологии разработки формализованных

	спецификациям и стандартам		обеспечения; стандартов качества программного обеспечения; этапов и принципов управления качеством процессов разработки в течение жизненного цикла программного обеспечения;		требований и спецификаций для контроля функциональности и качества программного обеспечения; средствами верификации программного обеспечения
5	способность проектировать и анализировать архитектуру программных систем	ПК4	методов проектирования и анализа архитектуры систем; моделей, методов, алгоритмов и программной инфраструктуры для организации глобально распределенной обработки данных	определять состав и объем сведений, необходимых и достаточных для построения адекватной, полной и непротиворечивой архитектуры программного обеспечения; применять специализированные методологии для построения архитектуры программных систем;	средствами проектирования программных систем; методами анализа архитектуры программного обеспечения;

3. Содержание и структура дисциплины (модуля)

Общая трудоемкость дисциплины составляет 7 зачетных единиц (252 часа).
Трудоемкость дисциплины по видам работ

Вид работы	Трудоемкость, час.	
	3 семестр	4 семестр
Лекции (Л)	6	4
Практические занятия (ПЗ)	8	6
Лабораторные работы (ЛР)		
КСР		
Курсовая проект работа (КР)		
Расчетно - графическая работа (РГР)		
Самостоятельная работа (проработка и повторение лекционного материала и материала учебников и учебных пособий, подготовка к лабораторным и практическим занятиям, коллоквиумам, рубежному контролю и т.д.)	85	98
Подготовка и сдача экзамена		36
Подготовка и сдача зачета	9	
Вид итогового контроля (зачет, экзамен)	з	э

Содержание разделов и формы текущего контроля

№	Наименование и содержание раздела	Количество часов						Литература, рекомендуемая студентам*	Виды интерактивных образовательных технологий**
		Аудиторная работа				СРС	Всего		
		Л	ПЗ	ЛР	КСР				
1	Процессы командной разработки программного обеспечения MSF.	2	2			15	19	6.1.1, 6.3.4	*проблемное обучение, лекция-визуализация*
2	Управление жизненным циклом приложений.	4	6			108	118	6.1.1, 6.3.5	*Работа в команде, деловая игра, проблемное обучение, лекция-визуализация*
3	Гибкие технологии разработки ПО. Методология гибкой разработки SCRUM.	2	2			25	29	6.1.1, 6.3.4	*Работа в команде, деловая игра, проблемное обучение, лекция-визуализация*
4	Обеспечение качества программных продуктов.	2	4			35	41	6.1.1	*проблемное обучение, лекция-визуализация*

**Указывается номер источника из соответствующего раздела рабочей программы, раздел (например, Р 6.1 №1, гл.3)*

***Указываются образовательные технологии, используемые при реализации различных видов работы.*

Занятия, проводимые в интерактивной форме, составляют 60% от общего количества аудиторных часов по дисциплине _____.

Практические занятия (семинары)

№ занятия	№ раздела	Тема	Кол-во часов
1	1	Процессы командной разработки программного обеспечения MSF.	2
2	2	Управление жизненным циклом приложений.	2
3	2	Управление жизненным циклом приложений.	2
4	2	Управление жизненным циклом приложений.	2
5	3	Гибкие технологии разработки ПО. Методология гибкой разработки SCRUM.	2
6	4	Обеспечение качества программных продуктов.	2
7	4	Обеспечение качества программных продуктов.	2

4. Учебно-методическое обеспечение самостоятельной работы студентов

Перечень вопросов, структурированных по темам для самостоятельного изучения.

Тема 1 Процессы командной разработки программного обеспечения MSF.

Вопросы для самостоятельного изучения:

1. Основныеположения MSF for Agile Software Development
2. Основные принципы построения команды.
3. Ролевые группы и роли.
4. Зоны ответственности ролевых групп.
5. Задачи ролевых групп и взаимодействие с заинтересованными лицами.

Тема 2Управление жизненным циклом приложений.

Вопросы для самостоятельного изучения:

1. Единая интегрированная платформа для управления полным жизненным циклом приложений.
2. Управление жизненным циклом системных приложений (ALM).
3. Средства управления жизненным циклом приложений: тенденции развития.
4. Инструментарий DevOps.

Тема 3Гибкие технологии разработки ПО. Методология гибкой разработки Scrum.

Вопросы для самостоятельного изучения:

1. Интеграция контроля и обеспечения качества в Scrum.
2. Процесс ICONIX.
3. Стратегия актуализации документации.
4. Масштабирование Agile.
5. Бережливое производство.
6. Диаграммы причинно-следственной связи.
7. Роль аналитика в Scrum.
8. Сроки и долгосрочное планирование в Agile.
9. Лучшие практики управления командой в Scrum.

Тема 4Обеспечение качества программных продуктов.

Вопросы для самостоятельного изучения:

1. Стандартизация обеспечения качества программных средств.
2. Базовые стандарты административного управления качеством продуктов.
3. Стандарты, регламентирующие качество программных средств.
4. Типы мер при измерении показателей качества стандарт ISO/IES 9126-2.
5. Классификация моделей надежности.
6. ГОСТ Р ИСО/МЭК 9126.
7. Тестирование Web-приложений.
8. Тестирование баз данных.

5. Фонд оценочных средств

№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или ее части)	Уровень освоения, определяемый этапом формирования компетенции	Наименование оценочного средства*
1	Процессы командной разработки программного обеспечения MSF.	ОПК1, ОПК2	Повышенный, Повышенный.	Круглый стол, Т
2	Управление жизненным циклом приложений.	ОПК2, ПК2, ПК3, ПК4	Повышенный, Повышенный, Повышенный, Повышенный.	Круглый стол, Деловая игра, Т
3	Гибкие технологии разработки ПО. Методология гибкой разработки SCRUM.	ОПК1	Повышенный.	Деловая игра, Т
4	Обеспечение качества программных продуктов.	ПК3	Повышенный.	Круглый стол, Кейс-задача, Т

** Планируемые формы контроля: защита лабораторной работы (ЗЛР), курсовой работы (КР), расчетно-графической работы (РГР), домашнего задания (ДЗ) написание реферата (Р), эссе (Э), тестирование, ответы на вопросы (Т), кейс-анализ (КА) и т.д.*

Вопросы к зачету (экзамену)

1. Модель проектнойгруппы*MSF*.
2. Модель процессов *MSF*.
3. Управление проектами
4. Управление рисками.
5. Управление подготовкой.
6. Архитектурное проектирование.
7. Разработка приложения.
8. Тестирование приложения.
9. Шаблоны командных проектов.
10. Функциональные возможности TeamFoundationServer.

11. Участники проекта (группы ролей участников проекта разработки ПО, роли каждой из групп, возможные и нежелательные совмещения ролей).
12. Распределение ролей в команде.
13. Организация работы на этапе внедрения программного продукта.
14. Экстремальное программирование.
15. Комбинация практик экстремального программирования.
16. Роли в *Scrum*.
17. Фазы проекта в *Scrum*.
18. Проведение формальных сессий в *Scrum*.
19. Методология*Dynamic Systems Development Method*.
20. Типичная структура команды в *DSDM*.
21. Принцип назначения приоритетов *MoSCoW*.
22. Методология *Feature Driven Development*.
23. Тестирование программного обеспечения.
24. Признаки некачественного дизайна.
25. Качество ПО.
26. Функциональность. Надежность.
27. Удобство применения. Эффективность.
28. Сопровождаемость. Переносимость.
29. Метрики качества ПО.
30. Стандартная оценка значений показателей качества.
31. Управление качеством программных средств.
32. Модели оценки надежности.
33. Классификация моделей надежности.
34. Марковские и пуассоновские модели надежности.

Критерии оценки:

- оценка «отлично» выставляется студенту, если задания (практические) выполнены своевременно и была проявлена дискуссионная активность в рамках круглого стола, полные ответы на вопросы (теоретические);
- оценка «хорошо» - задания (практические) выполненысвоевременно и была проявлена дискуссионная активность в рамках круглого стола, менее половины ответовна теоретические вопросы не совсем полные;
- оценка «удовлетворительно» - задания (практические) выполненысвоевременно и была проявлена дискуссионная активность в рамках круглого стола, ответы на более чем половина теоретических вопросов не совсем полные;
- оценка «неудовлетворительно» - все или часть заданий (практические) выполнена не корректно или несвоевременно, задания не выполнены и отсутствовала дискуссионная активность в рамках круглого стола, ответы на теоретические вопросы неправильные;

- оценка «зачтено» выставляется студенту, если корректно и своевременно выполнены все задания (практические) по темам курса и была проявлена дискуссионная активность в рамках круглого стола;
- оценка «не зачтено» - все или часть заданий (практические) выполнена не корректно или задания не выполнены и отсутствовала дискуссионная активность в рамках круглого стола.

Типовые оценочные материалы

Раздел (тема) дисциплиныПроцессы командной разработки программного обеспечения MSF.

1. Перечень дискуссионных тем для круглого стола (дискуссии, полемики, диспута, дебатов)

Раздел (тема) дисциплины Процессы командной разработки программного обеспечения MSF.

1. МоделиMSF.
2. Распределение точек принятия решения.
3. Проекты Microsoft Consulting Services.
4. Книги библиотеки ITIL.

Критерии оценки:

- оценка «зачтено» выставляется студенту, если была проявлена дискуссионная активность в рамках круглого стола;
- оценка «не зачтено» - отсутствовала дискуссионная активность в рамках круглого стола.

Раздел (тема) дисциплиныУправление жизненным циклом приложений.

1. Перечень дискуссионных тем для круглого стола (дискуссии, полемики, диспута, дебатов)

Раздел (тема) дисциплины Управление жизненным циклом приложений.

1. Решения компании Microsoft по управлению жизненным циклом приложений.
2. Инструменты визуального проектирования в Visual Studio.
3. назначение программы Intelli Trace.
4. Возможности лаборатории тестирования –Lab Management.
5. Анализ подходов архитектурного проектирования ПО.
6. Анализ подходов в формировании требований к ПО.
7. Анализ средств разработки ПО.

Критерии оценки:

- оценка «зачтено» выставляется студенту, если была проявлена дискуссионная активность в рамках круглого стола;
- оценка «не зачтено» - отсутствовала дискуссионная активность в рамках круглого стола.

Раздел (тема) дисциплиныУправление жизненным циклом приложений.Гибкие технологии разработки ПО. Методология гибкой разработки SCRUM.

Деловая игра

Раздел (тема) дисциплины Управление жизненным циклом приложений. Гибкие технологии разработки ПО. Методология гибкой разработки SCRUM.

1 Тема (проблема). Разработка модуля ИС с учетом жизненного цикла приложений и в рамках методологии гибкой разработки SCRUM.

2 Концепция игры. Продуктом проекта является разработанное ПО (модуль ИС). Реализация осуществляется с учетом жизненного цикла приложений и в рамках методологии гибкой разработки SCRUMс учетом имеющихся ресурсов с распределением ролей.

3 Роли:

- SCRUM мастер;
- команда.

4 Ожидаемый (е) результат (ы)Разработанное ПО в рамках в рамках методологии гибкой разработки SCRUM, сопроводительная документация (по проекту), зафиксированный полученный опыт.

Критерии оценки:

- оценка «зачтено» выставляется студенту, если роль «сыграна» корректно;
- оценка «не зачтено» - роль «сыграна» некорректно.

Кейс-задача

Раздел (тема) дисциплины Управление жизненным циклом приложений. Гибкие технологии разработки ПО. Методология гибкой разработки SCRUM.

Задание (я):

- Расчет трудоемкости и сроков разработки ПО;
- Разработка иерархической структуры работ;

Кейс-задача

сформулировать показатели эффективности выполненной разработки; Количественно оценить эффективность.

Раздел (тема) дисциплиныОбеспечение качества программных продуктов**.**

Перечень дискуссионных тем для круглого стола (дискуссии, полемики, диспута, дебатов)

Раздел (тема) дисциплины Обеспечение качества программных продуктов.

1. Стандартизация обеспечения качества программных средств.
2. Базовые стандарты административного управления качеством продуктов.
3. Стандарты, регламентирующие качество программных средств.
4. Метрики качества ПО.
5. Типы мер при измерении показателей качества стандарт ISO/IES 9126-2.
6. Классификация моделей надежности.

Критерии оценки:

- оценка «зачтено» выставляется студенту, если была проявлена дискуссионная активность в рамках круглого стола;
- оценка «не зачтено» - отсутствовала дискуссионная активность в рамках круглого стола.

Кейс-задача

Раздел (тема) дисциплиныОбеспечение качества программных продуктов.

Задание (я):

- Оценить надежность разработанных программных средств с использованием Марковской модели надежности.
- Оценить надежность разработанных программных средств с использованием пуассоновских моделей надежности.

Кейс-задача

Задание (я):

Рассчитать метрики качества программного продукта

Критерии оценки:

- оценка «зачтено» выставляется студенту, если задание выполнено корректно и своевременно;
- оценка «не зачтено» - задание выполнено некорректно.

6. Учебно-методическое и информационное обеспечение дисциплины (модуля)

6.1 Основная литература

1. Орлов, С. А. Технологии разработки программного обеспечения. Современный курс по программной инженерии : [учебник для студ. вузов, обуч. по спец. "Программное обеспечение вычислит. техники и автоматизир. систем" напр. подготовки дипломирован. спец. "Информатика и вычислительная техника"] / С. А. Орлов, Б. Я. Цилькер. - 4-е изд. - СПб.: Питер, 2012.http://www.library.ugatu.ac.ru/pdf/teach/Orlov_Tehnolog_razrab_progr_obespech_Sovr_4izd_2012.pdf

6.2 Дополнительная литература

1. Троцкий, М. Управление проектами / Троцкий М. — Москва: Финансы и статистика, 2011.— 302 с.http://www.razym.ru/biz/biznes/296577-trockiy-m-i-dr-upravlenie-proektami.html

6.3 Интернет-ресурсы (электронные учебно-методические издания, лицензионное программное обеспечение)

На сайте библиотеки http://library.ugatu.ac.ru/ в разделе «Информационные ресурсы», подраздел «Доступ к БД» размещены ссылки на интернет-ресурсы.

1. Липаев В.В. Программная инженерия. Методологические основы. http://www.booksgid.com/programmer/4464-programmnaja-inzhenerija.html
2. Липаев В.В. Проектирование и производство сложных заказных программных продуктов. – М.: СИНТЕГ, 2011. – 408 с.http://www.computer-museum.ru/books/lipaev/lip_proektirovanie_slognoe.pdf
3. PMBOK - Project Management Book of Knowledge, USA, 2008.http://ui.ranepa.ru/media/uploads/attachment/source/2012/12/PMbok4.pdf
4. С. Якимчук. MSF – философия создания IT-решений или голые амбиции лидера, 2004:[http://www.citforum.ru/SE/project/msf/].
5. http://www.intuit.ru/studies/courses/4806/1054/lecture/16115

6.4 Методические указания к практическим занятиям

1. О.Н. Сметанина Методические указанияк практическим занятиям по курсу «Технологии разработки программного обеспечения» (электр. вариант).

7. Образовательные технологии

№	Наименование	Доступ, количество одновременных пользователей	Реквизиты договоров с правообладателями
Программного комплекса			
1	Операционная система MicrosoftWindows	1800 компьютеров	договор ЭФ-193/0503-14
2	MicrosoftOffice	1800 компьютеров	договор ЭФ-193/0503-14

8. Методические указания по освоению дисциплины

Раздел Процессы командной разработки программного обеспечения MSF.

*Знания:*Основные понятия. Модели жизненного цикла программного обеспечения. Зрелость процессов разработки ПО. ИТ-решения по управлению жизненным циклом ПО.IT-решение, принципы MSF.Модель жизненного цикла решения MSF. Модель команд, взаимодействием, рисками, поставками проекта.

Лекция: №1, ***Объем лекций:*** 2 часа

На изучение материала данной темы отводится 2 часа лекционных занятий, 2 часа практических занятий и 15 часов самостоятельной работы.

При самостоятельной проработке материалов темы 1 необходимо использовать:

- учебное пособие 6.1.1, 6.2.1
- презентации № 1 лекционного курса.

При изучении материалов темы 1 необходимо акцентировать внимание на следующих понятиях:

- Технология
- Технология разработки программного обеспечения
- Жизненный цикл программного обеспечения
- Каскадная (водопадная) модель ЖЦПО

- Итерационная спиральная модель ЖЦПО
- Инкрементная итерационная модель ЖЦПО
- Управление жизненным циклом приложений.
- ИТ-решения по управлению жизненным циклом ПО
- Управление продуктом
- Управление программой
- Рекомендации MSF относительно совмещения ролей в команде проекта
- Управление компромиссами
- IT решение
- Итеративный подход
- Интегративный подход
- Фазы проекта
- Вехи проекта
- Ролевой кластер Управление продуктом
- Ролевой кластер Управление программой
- Ролевой кластер Разработка
- Ролевой кластер Тестирование
- Ролевой кластер Удовлетворение потребителя
- Ролевой кластер Управление выпуском
- Ролевой кластер Архитектура

Необходимо самостоятельно освоить следующие вопросы:

1. Основные положения MSF for Agile Software Development
2. Основные принципы построения команды.
3. Ролевые группы и роли.
4. Зоны ответственности ролевых групп.
5. Задачи ролевых групп и взаимодействие с заинтересованными лицами.

При выполнении практического задания по теме 1 необходимо подготовиться для проведения дискуссии в рамках круглого стола по следующим темам:

1. Модели MSF.
2. Распределение точек принятия решения.
3. Проекты Microsoft Consulting Services.
4. ITIL.

Раздел Управление жизненным циклом приложений

Знания:Принципы управления жизненным циклом приложений. Интегрированная среда разработки приложений. Функция анализа покрытия кода. Виды схем архитектурного проектирования.

Лекция: №2, ***Объем лекций:***4 часа

На изучение материала данной темы отводится 4 часа лекционных занятий, 6 часов практических занятий и 108 часов самостоятельной работы.

При самостоятельной проработке материалов темы 2 необходимо использовать:

- учебное пособие 6.1.1, 6.2.2;

- презентации № 2 лекционного курса.

При изучении материалов темы 2 необходимо акцентировать внимание на следующих понятиях:

- Принципы управления жизненным циклом приложений
- Диаграммы UML
- Инструменты визуального проектирования в Visual Studio
- Интегрированная среда разработки приложения
- Платформа модульного тестирования Visual Studio
- Функция анализа покрытия кода
- Виды схем архитектурного проектирования
- назначение профилировщика в Visual Studio.
- Назначение программы Intelli Trace.
- Метрики кода в Visual Studio
- Назначение нагрузочных тестов в Visual Studio
- Возможности лаборатории тестирования –Lab Management

Необходимо самостоятельно освоить следующие вопросы:

1. Единая интегрированная платформа для управления полным жизненным циклом приложений.
2. Управление жизненным циклом системных приложений (ALM).
3. Средства управления жизненным циклом приложений: тенденции развития.
4. Инструментарий DevOps.

При выполнении практического задания по теме 2 необходимо подготовиться для проведения дискуссии в рамках круглого стола по следующим темам:

1. Проектно-ориентированное управление.
2. Управление интегративными процессами в проекте.
3. Управление предметной областью в проекте.

Раздел Гибкие технологии разработки ПО. Методология гибкой разработки SCRUM.
Знания: Принципы и значение гибкой разработки. Управление невыполненной работой.

Лекция: №3, ***Объем лекций:***2 часа

На изучение материала данной темы отводится 2 часа лекционных занятий, 2 часа практических занятий и 25 часов самостоятельной работы.

При самостоятельной проработке материалов темы 3 необходимо использовать:

- учебное пособие 6.1.1, 6.2.1;
- презентации № 3 лекционного курса.

При изучении материалов темы 3 необходимо акцентировать внимание на вопросах:

- Agile Modeling
- Agile Unified Process(AUP)

- Open UP
- Agile Data Method
- DSDM
- Extremeprogramming (XP)
- Adaptive software development (ADD)
- Feature driven development (FDD)
- Getting Real
- MSF fog Agile Software Development
- Scrum
- Организация команды
- Методология Scrum
- Рабочий элемент
- Элементы задела работы продукта
- Элемент работы
- Спринт
- Владелец продукта
- Невыполненная работа по продукту
- Незаконченная работа

Необходимо самостоятельно освоить следующие вопросы:

1. Интеграция контроля и обеспечения качества в Scrum.
2. Процесс ICONIX.
3. Стратегия актуализации документации.
4. Масштабирование Agile.
5. Бережливое производство.
6. Диаграммы причинно-следственной связи.
7. Роль аналитика в Scrum.
8. Сроки и долгосрочное планирование в Agile.
9. Лучшие практики управления командой в Scrum.

При выполнении практического задания по теме 3в виде **Деловой игры** необходимо разработать модуль ИС с учетом жизненного цикла приложений и в рамках методологии гибкой разработки SCRUM. **Концепция игры.** Продуктом проекта является разработанное ПО (модуль ИС). Реализация осуществляется с учетом жизненного цикла приложений и в рамках методологии гибкой разработки SCRUMс учетом имеющихся ресурсов с распределением ролей. **3 Роли:**SCRUM мастер; команда. **Ожидаемый результат.** Разработанное ПО в рамках в рамках методологии гибкой разработки SCRUM, сопроводительная документация (по проекту), зафиксированный полученный опыт.

Кейс-задача

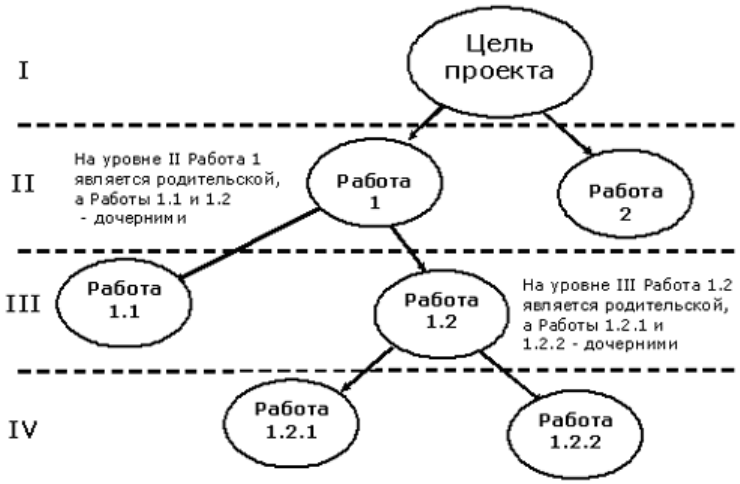
Раздел (тема) дисциплины Управление жизненным циклом приложений. Гибкие технологии разработки ПО. Методология гибкой разработки SCRUM.

Задание (я):

- Расчет трудоемкости и сроков разработки ПО;
- Разработка иерархической структуры работ;

Пример разработки иерархической структуры работ в рамках проекта

Иерархическая структура работ может быть представлена в следующем виде



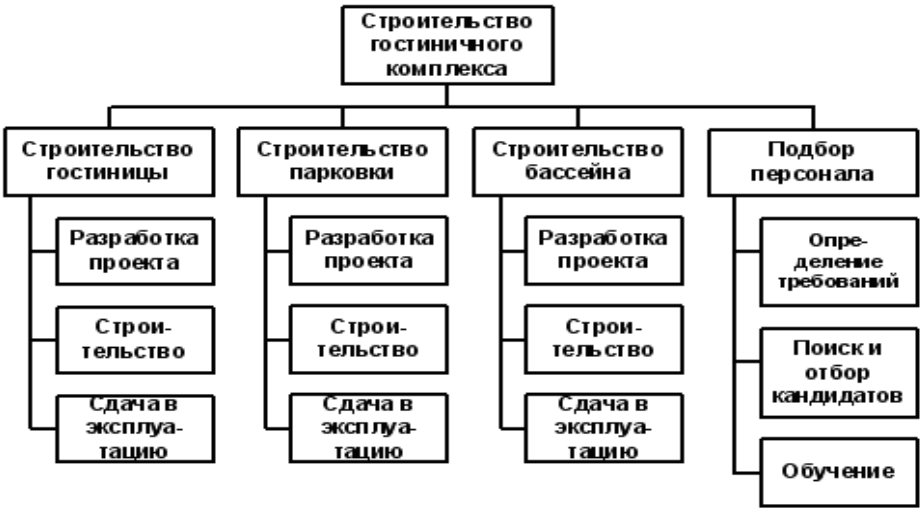
Варианты подходов к детализации работ:

а) Продуктовый подход

б) Подход по жизненному циклу

в) Функциональный подход

г) Организационный подход



Смешанный подход

Чтобы обеспечить рациональный размер пакетов работ, необходимо придерживаться следующих правил:
1. Правило 8/80 - означает, что ни одна из задач не должна иметь объем меньший, чем 8 чел/ч и больший, чем 80 чел/ч (это составляет соответственно от 1 до 10 дней при 8-часовой продолжительности рабочего дня).
2. Правило отчетного периода - продолжительность каждой задачи не должна быть больше периода, через который проводятся совещания, посвященные рассмотрению хода проекта. То есть, если такие совещания проводятся еженедельно, выполнение каждой задачи не должно превышать одной недели.
3. Правило «полезности» - при дроблении задачи на более мелкие следует учитывать, что существуют три причины, обусловливающие целесообразность такого разделения:
- задачу, полученную в результате такого дробления, легче оценить (в силу ее меньшей продолжительности во времени и, следовательно, меньшей неопределенности);
- более мелкие и конкретные задачи легче распределять между отдельными исполнителями;
- более мелкие задачи легче поддаются контролю.
Если же дробление не отвечает этим требованиям, от него следует отказаться.

Пример: «АС продажи документации»

Краткая легенда проекта. Заказчик ОАО «Х» является одним из ведущих производителей сложных технических изделий. Отдел «1», входящий в ОАО «Х», отвечает за продажу дополнительной сопроводительной документации для клиентов ОАО.
Дополнительная документация не входит в стандартную поставку, поскольку владелец этого технического изделия не всегда сам его эксплуатирует, а передает в эксплуатацию другой компании, которая становится клиентом «Х», и закупает у нее эксплуатационную документацию.
Ремонт и техобслуживание конкретного изделия может выполнять третья компания, которой уже потребуется детальная техническая документация по ремонту и обслуживанию. Она также становится клиентом «Х» и закупает у нее требуемую продукцию.
Основная функция отдела «1» — получение и обработка заказов на дополнительную документацию, согласно ежегодно рассылаемому каталогу. В связи с переездом отдела «1» в новое здание, была поставлена задача на разработку и поставку системы, автоматизирующей основную деятельность отдела «1».

ИСР проекта-примера разработки «АС продажи документации»:
1. Проект разработки «АС продажи документации»
1.1. Подготовка технического задания на автоматизацию
1.1.1.1. Проведение аналитического обследования
1.1.1.2. Разработка функциональных требований
1.1.1.3. Разработка требований базовому ПО
1.1.1.4. Разработка требований к оборудованию и к операционно-системному ПО
1.1.1.5. Согласование и утверждение ТЗ
1.1.1.6. ТЗ утверждено

1.2. Поставка и монтаж оборудования
1.2.1.Разработка спецификации на оборудование
1.2.2.Закупка и поставка оборудования
1.2.3. Монтаж оборудования
1.2.4. Установка и настройка операционно-системного ПО
1.2.5. Монтаж оборудования завершен
1.3. Поставка и установка базового ПО
1.3.1 .Разработка спецификаций на базовое ПО
1.3.2.Закупка базового ПО
1.3.3. Развертывание и настройка базового ПО
1.3.4. Базовое ПО установлено у заказчика
1.4. Разработка и тестирование прикладного ПО
1.4.1. Разработка спецификаций на прикладное ПО
1.4.2. Установка и конфигурирование рабочей среды
1.4.3. Проектирование и разработка ПО
1.4.3.1. Авторизация и аутентификация пользователей.
1.4.3.2. Разработка подсистемы заказа документации
1.4.3.2.1. Просмотр каталога продуктов.
1.4.3.2.2. Поиск продуктов по каталогу.
1.4.3.2.3. Заказ выбранных продуктов.
1.4.3.2.4. Просмотр информации о статусе заказа.
1.4.3.2.5. Информирование клиента об изменении статуса заказа.
1.4.3.2.6. Подсистема заказа документации передана в тестовую эксплуатацию (на серверах разработчика).
1.4.3.3. Разработка подсистемы обработки заказов
1.4.3.3.1. Просмотр и обработка заказов исполнителями из службы продаж.
1.4.3.3.2. Просмотр статистики поступления и обработки заказов за период.
1.4.3.3.3. Подсистема обработки заказов передана в тестовую эксплуатацию на оборудовании Заказчика
1.4.3.4. Разработка подсистемы сопровождения каталога
1.4.3.4.1. Подготовка и сопровождение каталога продукции.
1.4.3.5. Исправление ошибок
1.4.4. Тестирование ПО
1.4.4.1. Раунд 1
1.4.4.2. Раунд 2
1.4.4.3. Раунд 3
1.4.4.4. Выходное тестирование
1.4.5. Документирование прикладного ПО
1.5. Обучение пользователей
1.5.1 .Подготовка учебных курсов
1.5.2.Обучение сотрудников Отдела «1»
1.5.3.Обучение руководства ОАО «Х»
1.5.4.Обучение администраторов системы
1.6. Ввод в опытную эксплуатацию
1.6.1. Развертывание и настройка прикладного ПО
1.6.2. Проведение приемо-сдаточных испытаний

1.6.3. Акт передачи системы в опытную эксплуатацию утвержден
1.7. Сопровождение системы в период опытной эксплуатации
1.8. Система передана в промышленную эксплуатацию

Пример расчета трудоемкости и сроков разработки ПО
Прагматичный подход. Метод PERT

Метод оценки трудоемкости проекта PERT (Program / ProjectEvaluationandReviewTechnique, 1958 г.).

Диапазон неопределенности достаточно охарактеризовать 3 оценками:

Mi - наиболее вероятная оценка трудозатрат.

Oi - минимально возможные трудозатраты на реализацию пакета работ (ни один риск не реализовался; быстрее точно не сделаем; вероятность такого, что мы уложимся в эти затраты, равна 0).

Pi — пессимистическая оценка трудозатрат (все риски реализовались).

Оценка средней трудоемкости *Ei* по каждому элементарному пакету:

Ei = (Pi + 4Mi + Oi)/6.

Среднеквадратичное отклонение (СКО):

CKOi = (Pi - Oi)/6.

Согласно центральной предельной теореме теории вероятностей суммарная трудоемкость проекта

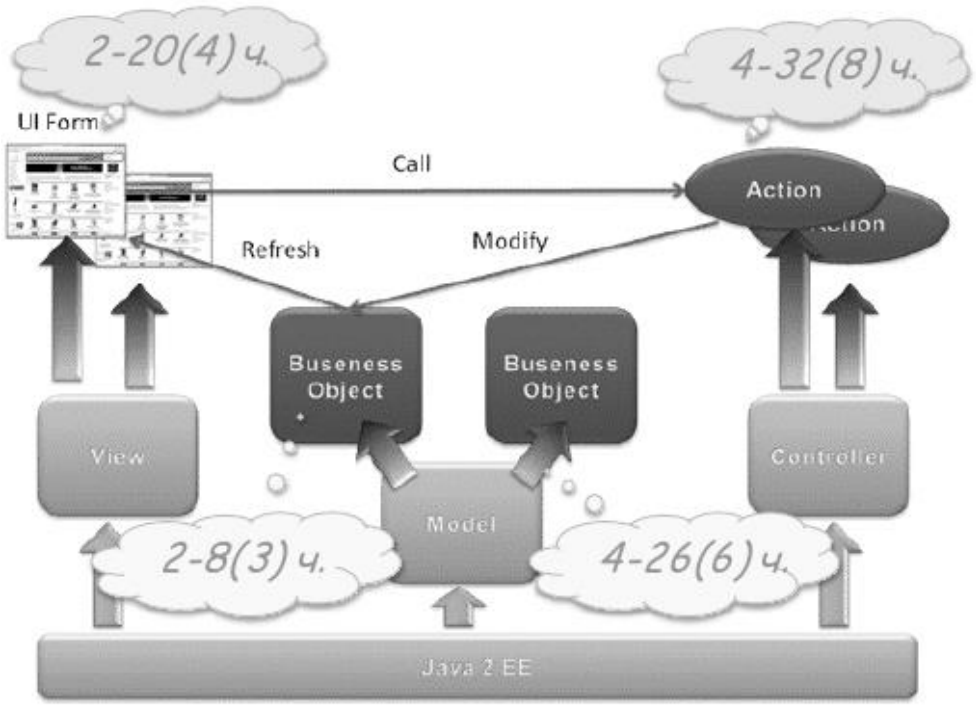
$E = \sum Ei$

СКО для оценки суммарной трудоемкости составляет: $CKO = \sqrt{\sum CKO_i^2}$.

Для оценки суммарной трудоемкости проекта, которую мы не превысим с вероятностью 95%, можно применить формулу:

*E95% = E + 2 * CKO, т.е.* вероятность того, что проект превысит данную оценку трудоемкости, составляет всего 5%.

Список элементарных пакетов работ, который используется при оценке трудоемкости, берется из нижнего уровня ИСР проекта.



Точки расширения:

Пользовательский экран (UI Form) - собирался из готовых визуальных компонентов.

Обработчики (Action): обрабатывают на сервере приложений события от активных визуальных компонентов, входящих в состав экрана.

Объекты (Business Obj) моделируют прикладную область, к ним обращались обработчики событий.

Может быть использован опыт аналогичных разработок.

Все разрабатываемые рабочие места различались по функциональности и сложности, статистика фактических трудозатрат на разработку прикладных систем позволяла оценивать проекты разработки нового приложения достаточно быстро и с высокой достоверностью.

По статистике, разработка и отладка требовала у программиста:

- для одного экрана — от 2 до 20 часов (наиболее вероятно — 4 часа);
- для одного обработчика событий — от 4 до 32 ч. (наиболее вероятно - 8ч.);
- для нового бизнес-объекта — от 2 до 8 ч. (наиболее вероятно — 3 часа);
- для добавления нового бизнес-метода - 2-26 ч. (наиболее вероятно - 6 ч.).

Весь проект прикладной разработки измерялся:

KUI — количество пользовательских экранов.

KAct — количество обработчиков событий.

KBO — количество новых бизнес-объектов.

KBM — количество новых или модифицируемых бизнес-методов.

Если новое разрабатываемое приложение содержит 20 пользовательских экранов, 60 обработчиков событий, 16 новых бизнес-объекта и 40 новых бизнес-методов, которые необходимо добавить, как в новые, так и в уже существующие бизнес-объекты, тогда, согласно статистике,

*EUI = (2 + 4*4 + 20) / 6 = 6.7 чел.*час.,EAct = (4 + 4*8 + 32) / 6 = 11.3 чел.*час.,*

*EBO = (2 +4*3 + 8) / 6 = 3.7 чел.*час.,EBM = (2 + 4*6 + 26) / 6 = 8.7 чел.*час.*

*СКОUI = (20 - 2) / 6 = 3 чел.*час;СКОAct = (32 - 4) / 6 = 4.7 чел.*час*

*СКОBO = (8 - 2) / 6 = 1 чел.*час;СКОBM = (26 - 2) / 6 = 4 чел.*час*

Для средней трудоемкости работ по кодированию в проекте может быть получена следующая оценка*E*: E=20*6,7+60*11,3+16*3,4+40*8,7≈1220 чел.*час.

$$СКО = \sqrt{20*3^2 + 60*4{,}7^2 + 16*1^2 + 40*4^2} = \sqrt{180 + 1325 + 16 + 640} \approx 46 \text{ чел.*час.}$$

Тогда для оценки суммарной трудоемкости проекта, которую не превысим с вероятностью 95%, получим

*E95% = 1220 + 2 *46 ≈ 1300 чел.*час.*

Относительная погрешность в оценке трудоемкости каждой такой элементарной работы составляла десятки процентов, для проекта, относительная погрешность оценки суммарной трудоемкости, сделанной по методу PERT, составила, приблизительно, лишь 4%.

При очень размытых оценках трудоемкости каждой из элементарных работ, ошибки можно сделать как в меньшую, так и большую стороны.

При фактической реализации проекта эти ошибки будут компенсироваться, что позволяет оценить общие трудозатраты по проекту существенно точнее, чем трудозатраты на каждую элементарную работу.

Это утверждение справедливо, если ИСР содержит *все* необходимые работы, которые должны быть выполнены для получения *всех* продуктов проекта.
Поскольку кодирование составляет только 25% общих трудозатрат проекта, то полученная оценка трудоемкости кодирования *4.
Суммарная трудоемкость проекта составит ≈ 5200 чел.*час.
При 100%-м назначении сотрудника на проект не означает, что он все 40 часов в неделю будет тратить на проектные работы.
Считается, что тратить он будет 60–80% своего рабочего времени.
Поэтому, в месяц сотрудник будет работать по проекту, примерно, 165 * 0.8 = 132 чел.*час/мес.
→ трудоемкость проекта составит ≈ 5200/132 ≈ 40 в чел.-месяц.
Оптимальная продолжительность проекта по формуле Б.Боэма:
*T = 2.5 * (40)^1/3 = 8.5 месяцев,*
средняя численность команды - 5 человек.
Потребление ресурсов в проекте неравномерно, поэтому начинать проект должны 1–3 человека, а на стадии реализации начальная численность команды может быть увеличена в несколько раз.

или

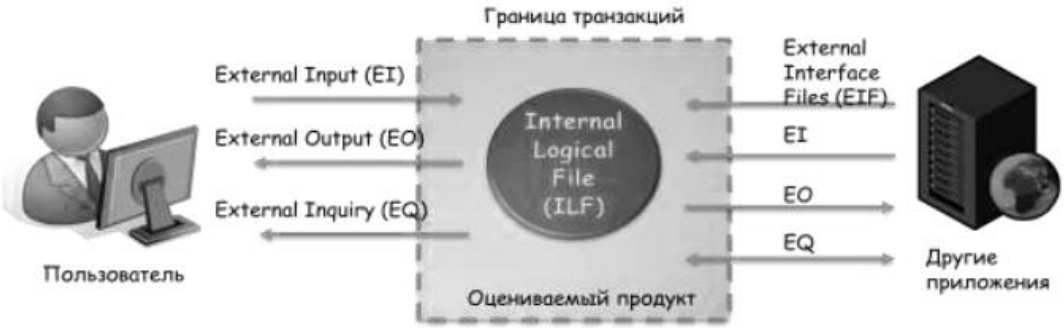
Расчет с использованием метода функциональных точек
Последовательность шагов:
- Определение типа оценки.
- Определение области оценки и границ продукта.
- Подсчет функциональных точек, связанных с данными.
- Подсчет функциональных точек, связанных с транзакциями.
- Определение суммарного количества не выровненных функциональных точек (UFP).
- Определение значения фактора выравнивания (FAV).
- Расчет количества выровненных функциональных точек (AFP).

Типы оценок:
- *Проект разработки.* Оценивается количество функциональности поставляемой пользователям в первом релизе продукта.
- *Проект развития.* Оценивается в функциональных точках проект доработки: добавление, изменение и удаление функционала.
- *Продукт.* Оценивается объем уже существующего и установленногопродукта.

Область оценки может включать:
- Все разрабатываемые функции (для проекта разработки)
- Все добавляемые, изменяемые и удаляемые функции (для проектов поддержки)
- Только функции, реально используемые, или все функции (при оценке продукта и/или продуктов).

Границы продукта определяют:
- Что является «внешним» по отношению к оцениваемому продукту.
- Где располагается «граница системы», через которую проходят транзакции передаваемые или принимаемые продуктом, с точки зрения пользователя.

- Какие данные поддерживаются приложением, а какие — внешние.



Логические данные системы:

Внутренние логические файлы (ILFs) — выделяемые пользователем логически связанные группы данных или блоки управляющей информации, которые поддерживаются внутри продукта.

Внешние интерфейсные файлы (EIFs) — выделяемые пользователем логически связанные группы данных или блоки управляющей информации, на которые ссылается продукт, но которые поддерживаются вне продукта.

Примеры логических данных(информационных объектов):

- клиент,
- счет,
- тарифный план,
- услуга.

Показатели сложности данных:

- *DET* (dataelementtype) — *неповторяемое уникальное поле данных*, например,

Имя Клиента — 1 DET;

Адрес Клиента (индекс, страна, область, район, город, улица, дом, корпус, квартира) — 9 DET's

- *RET* (recordelementtype) — *логическая группа данных*,

например, адрес, паспорт, телефонный номер.

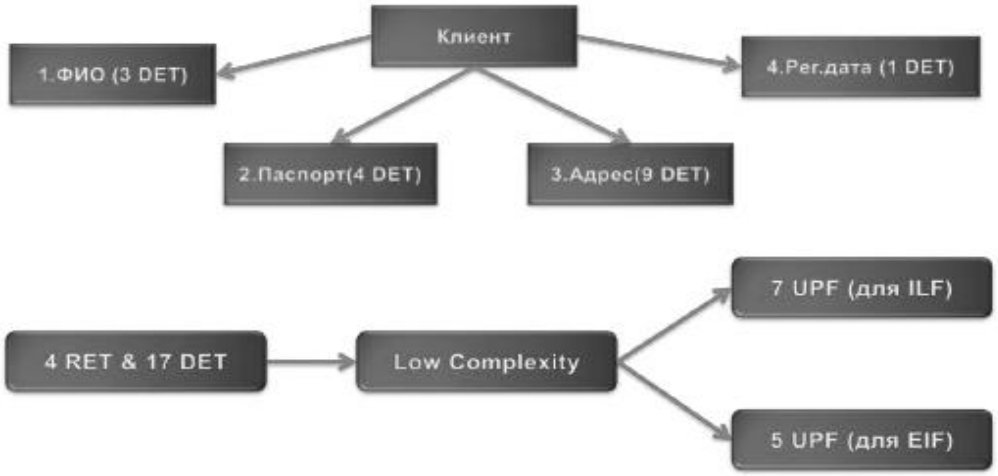
Оценка количества не выровненных ФТ, зависит от сложности данных, которая определяется на основании матрицы сложности.

Оценка данных в невыровненных ФТ – UFP;

- внутренние логические файлы - ILFs;
- внешние интерфейсные файлы- EIFs.

*	1-19 DET	20-50 DET	50+ DET
1 RET	Low	Low	Average
2-5 RET	Low	Average	High
6+ RET	Average	High	High

Сложность данных**	Количество UFP (ILF)	Количество UFP (EIF)
Low	7	5
Average	10	7
High	15	10



Объект «Клиент»: 4 логических группы данных;
17 неповторяемых уникальное полей данных.
По матрице* оцениваются сложность объекта данных - «Low».
Если объект - внутренний логический файл, то по матрице ** его сложность = 7 UPF.
Если объект - внешний интерфейсный файл, то его сложность = 5 UPF.
Типы транзакций:

- *EI* (externalinputs) — *внешние входные транзакции*, элементарные операции по обработке данных или управляющей информации, поступающих в систему из вне.
- *EO* (externaloutputs) — *внешние выходные транзакции*, элементарные операции по генерации данных или управляющей информации, которые выходят за пределы системы. Предполагает определенную логику обработки или вычислений информации из одного или более ILF.
- *EQ* (externalinquiries) — *внешние запросы*, элементарные операции, которые в ответ на внешний запрос извлекают данные или управляющую информацию из ILF или EIF.

Функция	Тип транзакции		
	EI	EO	EQ
Изменяет поведение системы	О	Д	NA
Поддержка одного или более ILF	О	Д	NA
Представление информации пользователю	Д	О	О

Характеристики для оценки сложности транзакции:
FTR (filetypereferenced) — *количество различных файлов* (информационных объектов) типа ILF и/или EIF модифицируемых или считываемых в транзакции.
DET (dataelementtype) — неповторяемое *уникальное поле данных*.
Примеры.
EI: поле ввода, кнопка.
EO: поле данных отчета, сообщение об ошибке.
EQ: поле ввода для поиска, поле вывода результата поиска.

*Матрица сложности внешних входных транзакций (EI)

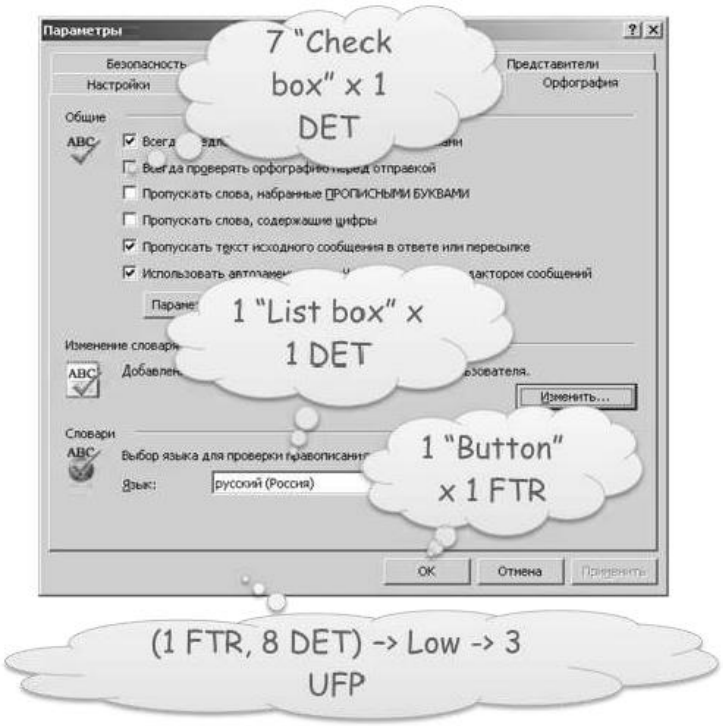
EI	1-4 DET	5-15 DET	16+ DET
0-1 FTR	Low	Low	Average
2 FTR	Low	Average	High
3+ FTR	Average	High	High

Матрица сложности внешних выходных транзакций и внешних запросов (EO & EQ)

EO & EQ	1-5 DET	6-19 DET	20+ DET
0-1 FTR	Low	Low	Average
2-3 FTR	Low	Average	High
4+ FTR	Average	High	High

** Сложность транзакций в не выровненных ФТ (UFP)

Сложность транзакций	Количество UFP (EI & EQ)	Количество UFP (EO)
Low	3	4
Average	4	5
High	6	7



Оценка управляющей транзакции (EI) для диалогового окна (задает параметры проверки орфографии в MS OfficeOutlook).
«Checkbox» - 1 DET * 7;
выпадающий список - 1 DET *1;
управляющая кнопка – транзакция - 1 FTR.

Например, при оценке управляющей транзакции по «OK»: - 1 FTR и 8 DET.
По матрице (*) - сложность транзакции - Low.
По матрице (**) - транзакция – 3 UFP.
Общий объем продукта в не выровненных функциональных точках (UFP) определяется путем суммирования по всем информационным объектам (ILF, EIF) и элементарным операциям (транзакциям EI, EO, EQ).

$$UFP = \sum_{ILF} UFP_i + \sum_{EIF} UFP_i + \sum_{EI} UFP_i + \sum_{EO} UFP_i + \sum_{EQ} UFP_i .$$

Помимо функциональных требований на продукт накладываются общесистемные требования, которые ограничивают разработчиков в выборе решения и увеличивают сложность разработки.
Для учета этой сложности применяется фактор выравнивания (VAF). Параметры VAF определяют системные характеристики продукта:
Обмен данными (0 — продукт представляет собой автономное приложение; 5 — продукт обменивается данными по более, чем одному телекоммуникационному протоколу).
Распределенная обработка данных (*0* — продукт не перемещает данные; 5 — распределенная обработка данных выполняется несколькими компонентами системы).
Производительность (*0* — пользовательские требования по производительности не установлены; 5 — время отклика сильно ограничено критично для всех бизнес-операций, для удовлетворения требованиям необходимы специальные проектные решения и инструменты анализа).
Ограничения по аппаратным ресурсам (*0* — нет ограничений; 5 — продукт целиком) должен функционировать на определенном процессоре и не может быть распределен).
Транзакционная нагрузка (*0* — транзакций не много, без пиков; 5 — число транзакций велико и неравномерно, требуются специальные решения и инструменты).
Интенсивность взаимодействия с пользователем (*0* — все транзакции обрабатываются в пакетном режиме; 5 — более 30% транзакций — интерактивные).
Эргономика (эффективность работы конечных пользователей) (0 — нет специальных требований; 5 — требования по эффективности очень жесткие).
Интенсивность изменения данных (ILF) пользователями (0 — не требуются; 5 — изменения интенсивные, жесткие требования по восстановлению).
Сложность обработки (*0* — обработка минимальна; 5 — требования безопасности, логическая и математическая сложность, многопоточность).
Повторное использование
(0 — не требуется; 5 — продукт разрабатывается как стандартный многоразовый компонент).
Удобство инсталляции (*0* — нет требований; 5 — установка и обновление ПО производится автоматически).
Удобство администрирования (0 — не требуется; 5 — система автоматически самовосстанавливается).

Портируемость (0 — продукт имеет только 1 инсталляцию на единственном процессоре; 5 — система является распределенной и предполагает установку на различные «железо» и ОС).
Гибкость (*0* — не требуется; 5 — гибкая система запросов и построение произвольных отчетов, модель данных изменяется пользователем в интерактивном режиме).
Системные параметры (degree of influence, *DI*) оцениваются по шкале от 0 до 5.
Расчет суммарного эффекта системных характеристик (total degree of influence, *TDI*) осуществляется простым суммированием:
$TDI = \sum DI$
Расчет значения фактора выравнивания производится по формуле
*VAF = (TDI *0.01) + 0.65*
Например, если, каждый из 14 системных параметров получил оценку 3, то их суммарный эффект составит *TDI* = 3 * 14 = 42.
В этом случае значение фактора выравнивания будет: VAF = (42 * 0.01) + 0.65 = 1.07
Начальная оценка количества выровненных ФТ для программного приложения:
*AFP = UFP * VAF.*
При этом учитывается только функциональность, которая реализуется в продукте.
Проект разработки продукта оценивается в *DFP* (developmentfunctionalpoint):
*DFP = (UFP + CFP) * VAF,*
где *CFP* (conversionfunctionalpoint) — ФТ, подсчитанные для дополнительной функциональности, которая потребуется при установке продукта, например, миграции данных.
Проект доработки и совершенствования продукта оценивается в *EFP* (enhancementfunctionalpoint):
*EFP = (ADD + CHGA + CFP) * VAFA + (DEL* VAFB),*
где
ADD — ФТ для добавленной функциональности;
CHGA — ФТ для измененных функций, рассчитанные после модификации;
VAFA — величина фактора выравнивания рассчитанного после завершения проекта;
DEL — объем удаленной функциональности;
VAFB — величина фактора выравнивания рассчитанного до начала проекта.
Суммарное влияние процедуры выравнивания лежит в пределах ±35% относительно объема рассчитанного в UFP.

или

Расчет с использованием методики COCOMO II
Различаются две стадии оценки проекта:
- *предварительная* оценка на начальной фазе и
- *детальная* оценка после проработки архитектуры.

Формула оценки трудоемкости проекта (чел.*мес.):

$PM = A * SITE^{E} * \prod_{i=1}^{n} EM_i$, A=2,94,

$E = B + 0{,}001 * \sum_{j=1}^{5} SF_j$, B=0,91,

где

SIZE — размер продукта в KSLOC

EMi— множители трудоемкости

SFj— факторы масштаба

n=7 — для предварительной оценки

n=17 — для детальной оценки

Для оценки трудоемкости необходимо знать размер программного продукта в тысячах строках исходного кода (KSLOC, Kilo Source Lines Of Code).

Размер программного продукта может быть, например, оценен экспертами с применением метода PERT.

Если анализ продукта определять методом ФТ, то его размер может быть рассчитан с использованием собственных статистических данных или с использованием статистики по отрасли.

Оценка количества строк, необходимых на реализацию одной не выровненной ФТ для некоторых распространенных языков программирования

Язык программирования	Оценка количества строк		
	Наиболее вероятная	Оптимистичная	Пессимистичная
Assembler	172	86	320
C	148	9	704
C++	60	29	178
C#	59	51	66
J2EE	61	50	100
JavaScript	56	44	65
PL/SQL	46	14	110
VisualBasic	50	14	276

Факторы масштаба SF – определяются характеристиками проекта:

- *PREC* — прецедентность, наличие опыт аналогичных разработок (VeryLow — опыт в продукте и платформе отсутствует; ExtraHigh — продукт и платформа полностью знакомы)
- *FLEX* — гибкость процесса разработки (VeryLow — процесс строго детерминирован; ExtraHigh — определены только общие цели).
- *RESL* — архитектура и разрешение рисков (VeryLow — риски неизвестны/не проанализированы; ExtraHigh — риски разрешены на 100%)
- *TEAM* — сработанность команды (VeryLow — формальные взаимодействия; ExtraHigh — полное доверие, взаимозаменяемость и взаимопомощь).
- *PMAT* — зрелостьпроцессов (VeryLow — CMM Level 1; ExtraHigh — CMM Level 5)

Значение фактора масштаба, в зависимости от оценки его уровня

Фактор масштаба	Оценка уровня фактора					
	VeryLow	Low	Nominal	High	VeryHigh	ExtraHigh
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

Для предварительной оценки необходимо оценить уровень семи множителей трудоемкости M:

- *PERS* — квалификация персонала (ExtraLow — аналитики и программисты имеют низшую квалификацию, текучесть больше 45%; ExtraHigh — аналитики и программисты имеют высшую квалификацию, текучесть меньше 4%);
- *RCPX* — сложность и надежность продукта (ExtraLow — продукт простой, специальных требований по надежности нет, БД маленькая, документация не требуется; ExtraHigh — продукт очень сложный, требования по надежности жесткие, БД сверхбольшая, документация требуется в полном объеме);
- *RUSE* — разработка для повторного использования (Low — не требуется; ExtraHigh — требуется переиспользование в других продуктах);
- *PDIF* — сложность платформы разработки (ExtraLow — специальные ограничения по памяти и быстродействию отсутствуют, платформа стабильна; ExtraHigh — жесткие ограничения по памяти и быстродействию, платформа нестабильна);
- *PREX* — опыт персонала (ExtraLow — новое приложение, инструменты и платформа; ExtraHigh — приложение, инструменты и платформа хорошо известны);
- *FCIL* — оборудование (ExtraLow — инструменты простейшие, коммуникации затруднены; ExtraHigh — интегрированные средства поддержки жизненного цикла, интерактивные мультимедиа коммуникации);
- *SCED* — сжатие расписания (VeryLow — 75% от номинальной длительности; VeryHigh — 160% от номинальной длительности).

Влияние множителей трудоемкости в зависимости от их уровня определяется их числовыми значениями

	Оценка уровня множителя трудоемкости

	ExtraLow	VeryLow	Low	Nominal	High	Very High	ExtraHigh
PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.5
RCPX	0.49	0.60	0.83	1.00	1.33	1.91	2.72
RUSE	n/a	n/a	0.95	1.00	1.07	1.15	1.24
PDIF	n/a	n/a	0.87	1.00	1.29	1.81	2.61
PREX	1.59	1.33	1.22	1.00	0.87	0.74	0.62
FCIL	1.43	1.30	1.10	1.0	0.87	0.73	0.62
SCED	n/a	1.43	1.14	1.00	1.00	1.00	n/a

Суммарная трудоемкость не равна простой сумме трудоемкостей разработки каждого из *N* компонентов:

$$PM \neq \sum_{k=1}^{N} PM_k .$$

Простая сумма не учитывает взаимосвязи компонентов и трудозатраты на их интеграцию.

Методика COCOMO II определяет следующую последовательность вычисления трудоемкости при многокомпонентной разработке.

Суммарный размер продукта рассчитывается, как сумма размеров его компонентов:

$$SIZE^A = \sum_{k=1}^{N} SIZE_k .$$

Базовая трудоемкость проекта:

$$PM^B = A*(SIZE^A)^E * SCED .$$

Базовая трудоемкость каждого компонента:

$$PM_k^B = PM^B * \frac{SIZE_k}{SIZE^A} .$$

Оценка трудоемкости компонентов с учетом всех множителей трудоемкости, кроме множителя *SCED*

$$PM_k^{'} = PM_k^B * \prod_{i=1}^{6} EM_i .$$

Итоговая трудоемкость определятся по формуле:

$$PM = \sum_{k=1}^{n} PM_k^{'} .$$

Длительность проекта:

$$TDEV = C*(PM_{NS})^{D+0,2*0,01*\sum_{j=1}^{5} SF_j} * \frac{SCED}{100},$$

где,

C = 3,67*; D* = 0,28;

PM_{NS} — трудоемкость без учета множителя *SCED,* определяющего сжатие расписания.

Кейс-задача
сформулировать показатели эффективности выполненной разработки; Количественно оценить эффективность.

Пример: показатели эффективности проекта

Размер эффекта от реализации проекта непосредственно определяется его ожидаемой эффективностью, проявляющейся:
- в продуктовом аспекте (улучшение качества и расширение ассортимента товаров);
- в технологическом аспекте (рост производительности труда и улучшение его условий);
- в функциональном аспекте (повышение эффективности управления);
- в социальном аспекте (улучшение качества жизни) и т.д.

Эффект (конечный результат) от реализации проекта может иметь вид нового технологического процесса, нового продукта для заказчика, применения новой информационной системы, элемента или подсистемы, встроенной в другую систему, анализа осуществимости проекта или программы обучения. Т.е., конечный результат зависит от целей проекта.

1. *Макроэкономическая эффективность* характеризует влияние проекта на национальную и региональную экономику. Может выражаться такими показателями, как рост экспорта, увеличение валового регионального продукта и т.п. Многие результаты проекта (например, социальные, экологические, демографические, научные) могут проявиться через достаточно отдаленное время и не иметь прямого количественного выражения. Это затрудняет оценку макроэкономической эффективности проекта и требует особой тщательности в прогнозировании его результатов.
2. *Бюджетная эффективность* в общем случае может быть охарактеризована как превышение доходов бюджета, возникающих в результате реализации проекта (в виде налогов, поступлений от экспорта и т.п.) над расходами бюджета (прямое финансирование, налоговые льготы, инвестиционный налоговый кредит и т.п.), связанными с данным проектом.
3. *Коммерческая эффективность* определяется как разница между доходами и расходами участников проекта, возникающими вследствие его реализации (чистые денежные потоки по проекту).

Основным требованием при оценке эффективности проекта является учет разновременности затрат и приведение их к единому периоду времени — дисконтирование.

Текущая (современная, приведенная) стоимость денежных средств (present value — *PV*) означает сегодняшнюю стоимость сумм, которые будут получены в будущем (через определенный период времени). Расчет текущей стоимости денежных сумм осуществляется на основе коэффициента дисконтирования. *Дисконтирование* — это вычисление текущей стоимости некой денежной суммы.

Коэффициент дисконтирования (k_d) - приведения сумм, получаемых в будущем, к настоящему моменту - показывает сегодняшнюю стоимость 1 денежной единицы, которая будет получена через t периодов времени при процентной ставке r.

Чистая дисконтированная (текущая) стоимость(*NPV*) - разница между рыночной стоимостью проекта и затратами на его реализацию.

Расчет коэффициента дисконтирования:

Поток платежей и поступлений рассматривается как дискретный (прерывный): итоги подводятся на конец каждого года (квартала, месяца), полученные значения величины денежных потоков приводятся к текущему моменту времени исходя из формулы сложного процента $k_d = (1+r)^{-t}$ (1).

Если поток платежей и поступлений рассматривается как непрерывный, дисконтирование проводится на основе непрерывного годового коэффициента дисконтирования, рассчитываемого по формуле: $k_d = e^{-rt}$ (2).

Для перехода от номинальной процентной ставки (r_N) при непрерывном начислении процента к эффективной процентной ставке (r) и обратно используются следующие формулы:

$$r = e^{r_N} - 1;\ r_N = \ln(1+r).$$

Пример

!!!Допустим, что заключен договор на выполнение определенной работы, по окончании которой (через два года) обещано заплатить один миллион рублей. Если процентная ставка по депозитам составляет 10%, то текущая стоимость дохода составит: $1000000*(1+0{,}10)^{-2}$=826446 руб.

Таким образом, стоимость вознаграждения составляет 826,4 тыс. руб., что тоже немало, но меньше обещанного миллиона!!!

Учет эффекта дисконтирования позволяет сделать 2 основных вывода о современной стоимости сумм, получаемых по прошествии определенного времени.

1. Текущая стоимость некоторой суммы будет тем ниже, чем более отдален во времени момент ее получения.
2. Текущая стоимость данной суммы при фиксированном сроке ее получения будет тем ниже, чем будет выше ставка учетного %.

Важное значение для точности инвестиционных расчетов имеет учет инфляции.

Исходя из предположения о том, что можно относительно точно спрогнозировать темпы инфляции за период (I) и определить желаемый уровень реальной доходности инвестиций (R) рассчитывать учетную ставку процента по формуле: $R=r(1+I)+I$ (4),

где r — расчетная (реальная) процентная ставка.

В основе применения инвестиционных расчетов лежат определенные исходные условия и предпосылки, выполнение которых обеспечивает как возможность осуществления самих расчетов, так и возможность получения заслуживающих доверия результатов.

Среди этих условий выделим следующие.

1. Рассматривается долгосрочный проект, имеющий срок реализации или полезного использования несколько лет или периодов иной длительности, если в качестве единицы периода выбран срок менее одного года (полугодие, квартал, месяц и т.п.) В случаях, когда это специально не оговаривается, считается, что единицей периода является один год.
2. Каждый проект описывается платежным рядом, элементы которого представляют собой сальдо доходов и расходов инвестора (денежные потоки) за каждую единицу периода реализации. Отрицательное значение компонента платежного ряда

означает, что в данном году расходы инвестора превысили его доходы, а положительное значение свидетельствует о превышении доходов над расходами.

3. Существование развитого рынка капитала, обеспечивающего возможность получения внешнего финансирования инвестиционных проектов, и дополнительного эффективного использования временно свободного капитала инвестора.

4. Предполагается, что будущие доходы и расходы инвестора, связанные с реализацией проекта, точно известны, т.е. речь идет о гарантированном вложении капитала и отсутствии неопределенности исходной информации. В условиях использования долгосрочных проектов такое условие является достаточно жестким. Инвестор, выбирая методы обоснования проектов, должен отдавать себе отчет в том, что ни один из них не может вполне соответствовать реальным хозяйственным процессам, а будущие фактические результаты могут значительно отличаться от тех, которые были получены в процессе инвестиционных расчетов.

5. В процессе осуществления динамических инвестиционных расчетов учитываются только экономические факторы, определяющие будущие результаты проекта.

При этом определенная группа факторов и условий, которая оказывает практическое влияние на эти результаты, не учитывается. В их число входят политические, социальные, правовые и прочие подобные факторы. Если необходимо учесть влияние подобных факторов на реализацию проектов, то наряду с методами инвестиционных расчетов следует провести специальные исследования и обоснования, связанные с использованием качественных методов анализа и прогноза.

Оценка денежных потоков является самым важным этапом в финансово-инвестиционном анализе проекта.

С точки зрения предприятия денежные потоки проекта определяются как разность между денежными потоками фирмы за каждый период в случае реализации проекта и денежными потоками в случае отказа от проекта:

Денежный поток проекта (*CFt*) = *CFt* фирмы при принятии проекта - *CFt* фирмы без принятия проекта.

При оценке эффективности проекта учитываются только релевантные денежные потоки - разница между перспективным движением наличности, связанным с реализацией проекта, и оным при отсутствии проекта.

Шаблон движения денежных средств по проекту

Показатели	Год		
	Мес. 1	Мес. 2	Мес. n
1.Средства на начало периода			
Поступления			
2. Выручка от реализации проекта			
Текущие затраты			
3. Сырье и материалы			
4. Арендная плата			
5. Заработная плата			
6. Начисления на заработную плату			

7. Расходы на содержание и эксплуатацию оборудования			
8. Амортизация ОС			
9. Услуги сторонних организаций			
10. Коммунальные услуги - всего:			
- отопление, водоснабжение и канализация			
- телефон (арендная плата, оплата переговоров)			
- электроэнергия			
11. Накладные расходы			
12. Проценты по кредитам, оплата услуг банка			
13. НДС к уплате (стр.13.1 - стр.13.2)			
13.1 НДС полученный			
13.2 НДС уплаченный			
14. Налоги, учитываемые в составе затрат			
- транспортный налог (по ставкам)			
- на имущество (2,2% от стоимости имущества предприятия)			
15. Итого текущих расходов (сумма строк 5-17)			
16. Прибыль (строка 2 - строка 18)			
17. Налог на прибыль (стр. 19 *29%)			
18.Операционный денежный поток (стр.16 - стр.18+стр.8)			
19. Инвестиционный денежный поток			
19.3. Капитальные вложения			
20.Совокупный инвестиционный и операционный денежные потоки (стр.18 +стр.19)			
21. Финансовый денежный поток			
21.1 кредиты			
21.2 погашение кредитов			
21.3 расчеты с учредителями (выплата дивидендов)			
22. Остаток средств на конец периода (стр.1 + стр.20 + стр.21)			

Среди основных методов инвестиционных расчетов можно выделить следующие:
- метод чистой дисконтированной стоимости;
- метод внутренней нормы доходности;
- метод дисконтированного периода окупаемости.

Чистая дисконтированная (текущая) стоимость (*NPV*) - разница между рыночной стоимостью проекта и затратами на его реализацию. Представляет собой сумму дисконтированных по годам денежных потоков за все периоды реализации проекта:

$NPV = \sum NCF_t (1+r)^{-t}$ (5), где r - процентная ставка, использующаяся для данного инвестиционного проекта (норма дисконтирования); T - период реализации проекта.

Чистая дисконтированная стоимость показывает настоящую стоимость разновременных результатов от реализации конкретного проекта. Другими словами,

чистая дисконтированная стоимость - мера той добавочной или вновь создаваемой стоимости, которую мы получим, финансируя сегодня первоначальные затраты проекта.

Инвестиционное предложение следует рассматривать, если чистая дисконтированная стоимость проекта положительная. В случае если чистая текущая стоимость проекта меньше 0, проект должен быть отклонен. Из нескольких альтернативных проектов следует выбирать тот, у которого при прочих равных условиях больше чистая текущая стоимость.

Положительная величина чистой текущей стоимости свидетельствует не только о полном возмещении затрат на инвестиционный проект при прогнозируемом уровне доходности капитала, но и о получении дополнительного дохода, т.е. об увеличении активов предприятия вследствие принятия проекта.

При расчете дисконтированной стоимости принято делать некоторые допущения, которые значительно упрощают инвестиционные расчеты.

1. Капитал можно привлечь и разместить под один и тот же процент.
2. Денежные притоки и оттоки происходят в начале или конце каждого периода, а не возникают в течение всего периода.
3. Денежные потоки точно определены и нет необходимости делать поправку на риск.
4. В качестве стратегической цели принимается максимизация благосостояния фирмы или владельца.

Одна из важнейших проблем при использовании критерия чистой дисконтированной стоимости - *выбор ставки дисконтирования*. С теоретической точки зрения она представляет собой стоимость капитала предприятия, т.е. те альтернативные издержки, которые связаны с инвестированием в данный проект.

Пример

!!!Предприятие рассматривает инвестиционный проект, объем инвестируемого капитала по которому составляет 700 млн руб., при этом за последующие пять лет ожидаются следующие чистые денежные потоки от реализации проекта: в первый год - 200 млн руб., во второй - 300 млн руб., в третий - 300 млн руб., в четвертый - 200 млн руб., в пятый - 100 млн руб. Требуется найти чистую дисконтированную стоимость инвестиционного проекта при условии, что норма дисконтирования составляет 14%.

Решение приводится в таблице!!!

Год	Инвестиции, млн. руб.	Чистый денежный поток, NCF_t, млн. руб.	Коэффициент дисконтирования, k_{dt}(14%), млн. руб.	Дисконтированный чистый денежный поток, NCF_t*k_{dt}, млн. руб.
0	700	-	-	-700
1	-	200	0,8772	175,4
2	-	300	0,7695	230,9
3	-	300	0,6750	202,5
4	-	200	0,5921	118,4
5	-	100	0,5194	51,9

Чистая дисконти-рованная стоимость, NPV	-	-	-	79,2

Принятие решения при сравнении проектов*A*и *B* на основе значения показателя чистой дисконтированной стоимости может осуществляться в следующих условиях:
1. *NPVA*> 0, а *NPVB* < 0. Тогда выбирается проект *A*.
2. *NPVA*> 0; *NPVB* > 0; *NPVA* >*NPVB*. Выбирается проект *A*.
3. *NPVA*> 0; *NPVB* > 0; *NPVA* = *NPVB*. Для третьей ситуации необходимо использовать дополнительные методы расчета на основе *NPV*. К таким дополнительным методам относятся *дисконтированный период окупаемости проекта* и *доля дисконтированной стоимости*.
На методе чистой дисконтированной стоимости основано правило окупаемости, в соответствии с которым предприятия выбирают такие сроки окупаемости инвестиционных проектов, при которых чистая дисконтированная стоимость будет максимальной. Если инвестиции по проекту осуществляются равномерно, то оптимальный дисконтированный период окупаемости(*DPP*) может быть определен по формуле

$DPP = \frac{1}{r} - \frac{1}{r(1+r)^T}$, где$T$ - срок жизни проекта.

Инвестиции считаются приемлемыми, если дисконтированный период окупаемости меньше некоторого заранее определенного числа лет.
Если инвестиции и денежные потоки проекта неравномерны, дисконтированный период окупаемости рассчитывается путем вычитания из первоначальных инвестиций суммы дисконтированных чистых денежных потоков до того момента, пока сальдо не будет равным нулю.
!!!Пример. Допустим, что требуемая норма доходности инвестиций 12,5%.
Инвестиционные затраты составляют 300 млн руб., а денежные потоки в течение 5 лет запланированы в размере 100 млн руб. в год. Тогда рассчитаем дисконтированный период окупаемости.

Год	Денежный поток, млн. руб.		Кумулятивный денежный поток, млн. руб.	
	Недисконти-рованный	Дисконти-рованный	Недисконтированный	Дисконти-рованный
1	100	89	100	89
2	100	79	200	168
3	100	70	300	238
4	100	62	400	300
5	100	55	500	355

Таким образом, потребуется четыре года. Отметим, что проект, который окупается на основе дисконтированного периода окупаемости, всегда имеет положительную чистую текущую стоимость. В примере она равна 55 млн руб.!!!

К достоинствам этого критерия можно отнести легкость понимания, учет фактора времени, положительное влияние на ликвидность проекта, соответствие критерию чистой текущей стоимости. Однако, у него есть существенные недостатки: субъективность в определении времени возврата инвестиций, игнорирование денежных потоков за пределами срока окупаемости и склонность к отказу от прибыльных долгосрочных проектов.

Доля дисконтированной стоимости рассчитывается как отношение чистой дисконтированной стоимости к величине первоначально инвестированного капитала:

$$D_{NPV} = \frac{NPV}{K_0},$$

где K_0 - величина первоначальных капитальных вложений. Величина чистой дисконтированной стоимости обратно пропорциональна процентной ставке (норме доходности капитала). Графически эта зависимость проиллюстрирована на рис. Этот график называется *диаграммой чистой текущей стоимости*.

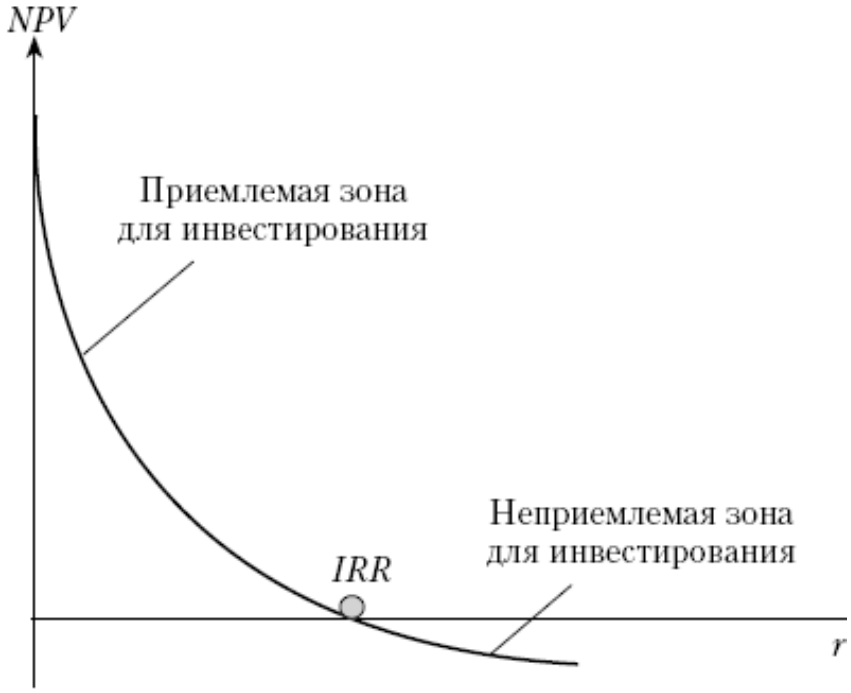


Рис. Диаграмма чистой текущей стоимости

Точка, в которой график чистой дисконтированной стоимости пересекает ось абсцисс, называется внутренней нормой доходности проекта (*IRR*). Уравнение для расчета внутренней нормы доходности выглядит следующим образом: $\sum_{t=0}^{T} NCF_t (1+IRR)^{-t} = 0$.

Проект является приемлемым, если его внутренняя норма доходности превышает уровень доходности, требуемый инвестором.

Раздел Обеспечение качества программных продуктов.

Знания:Модульное тестирование. Исследовательское тестирование.Интеграционное тестирование. Функциональное тестирование. Нагрузочное тестирование. Регрессионное тестирование.Комплексное тестирование. Приемочное тестирование.

Лекция: №4, ***Объем лекций:*** 2 часа

На изучение материала данной темы отводится 2 часа лекционных занятий, 4 часа практических занятий и 35 часов самостоятельной работы.

При самостоятельной проработке материалов темы 4 необходимо использовать:

- учебное пособие 6.1.1;
- презентации № 4 лекционного курса.

При изучении материалов темы 4 необходимо акцентировать внимание на следующих понятиях:

- Качество программного продукта,
- Программное обеспечение,
- Программный продукт,
- Надежность,
- Производительность,
- Удобство сопровождения,
- Мониторинг,
- Диспетчер,
- Команда,
- Шлюз,
- Модульное тестирование,
- Интеграционное тестирование,
- Функциональное тестирование,
- Нагрузочное тестирование,
- Регрессионное тестирование,
- Комплексное тестирование,
- Приемочное тестирование,
- Microsoft Test Manager,
- Lab Management,
- Рефакторинг,
- Признаки некачественного дизайна кода.

Необходимо самостоятельно освоить следующие вопросы:

1. Стандартизация обеспечения качества программных средств.
2. Базовые стандарты административного управления качеством продуктов.
3. Стандарты, регламентирующие качество программных средств.
4. Типы мер при измерении показателей качества стандарт ISO/IES 9126-2.
5. Классификация моделей надежности.
6. ГОСТ Р ИСО/МЭК 9126.
7. Тестирование Web-приложений.
8. Тестирование баз данных.

При выполнении практического задания по теме 4 в виде **Кейс-задачи**оценить надежность разработанных программных средств с использованием Марковской модели надежности; оценить надежность разработанных программных средств с использованием пуассоновских моделей надежности.

Пример. Марковские и пуассоновские модели надежности

Марковский процесс характеризуется дискретным временем и конечным множеством состояний. Временной параметр пробегает неотрицательные числовые значения, а процесс (цепочка) определяется набором вероятностей перехода $p_{ij}(n)$, т.е. вероятностью перейти на n-шаге из состояния i в состояние j. Процесс называется однородным, если он не зависит от n. В моделях, базирующихся на процессе Маркова, предполагается, что количество дефектов, обнаруженных в ПС, в любой момент времени зависит от поведения системы и представляется в виде стационарной цепи Маркова. При этом количество дефектов конечное, но является неизвестной величиной, которая задается для модели в виде константы. Интенсивность отказов в ПС или скорость прохода по цепи зависит лишь от количества дефектов, которые остались в ПС. К этой группе моделей относятся: Джелински-Моранды, Шика-Вулвертона, Шантикумера и др.

Ниже рассматриваются некоторые модели надежности, которые обеспечивают рост надежности ПО (модели роста надежности), находят широкое применение на этапе тестирования и описывают процесс обнаружения отказов при следующих предположениях:

- все ошибки в ПС не зависят друг от друга с точки зрения локализации отказов;
- интенсивность отказов пропорциональна текущему числу ошибок в ПС (убывает при тестировании программного обеспечения);
- вероятность локализации отказов остается постоянной;
- локализованные ошибки устраняются до того, как тестирование будет продолжено;
- при устранении ошибок новые ошибки не вносятся.

Приведем основные обозначения величин при описании моделей роста надежности:

m- число обнаруженных отказов ПО за время тестирования;

X_i- интервалы времени между отказамиi-1 иi, приi=1,…,m;

S_i- моменты времени отказов (длительность тестирования доi-отказа),S_i=X_kприi=1,…,m;

T- продолжительность тестирования ПО (время, для которого определяется надежность);

N- оценка числа ошибок в ПО в начале тестирования;

M- оценка числа прогнозированных ошибок;

MT- оценка среднего времени до следующего отказа;

$E(T_p)$ - оценка среднего времени до завершения тестирования;

$Var(T_p)$ - оценка дисперсии;

$R(t)$ - функция надежности ПО;

$Z_i(t)$ - функция риска в момент времениtмеждуi-1 иi-отказами;

c- коэффициент пропорциональности;

b- частота обнаружения ошибок.

Далее рассматриваются несколько моделей роста надежности, основанные на этих предположениях и использовании результатов тестирования программ в части отказов, времени между ними и др.

Модель Джелинского-Моранды. В этой модели используются исходные данные, приведенные выше, а также:
m- число обнаруженных отказов за время тестирования;
X_i- интервалы времени между отказами;
T- продолжительность тестирования.
Функция риска$Z_i(t)$ в момент времени*t*расположена между *i*-1 и *I*имеет вид:
$Z_i(t)=c(N-n_{i-1})$,
где$i=1,\ldots, m$, $T_{i-1}<t<T_i$.
Эта функция считается ступенчатой кусочнопостоянной функцией с постоянным коэффициентом пропорциональности и величиной ступени -*c*. Оценка параметровсиNпроизводится с помощью системы уравнений:

$$\sum_{i=1}^{m}\frac{1}{N-N_{i-1}}-\sum_{i=1}^{m}cX_i=0,\quad \frac{n}{c}-NT-\sum_{i=1}^{m}X_i n_{i-1}=0.$$

При этом суммарное время тестирования вычисляется так: $T=\sum_{i=1}^{m}X_i$.

Выходные показатели для оценки надежности относительно указанного времени *T* включают:

- число оставшихся ошибок$M_m=N-m$;
- среднее время до текущего отказа$MT_m=1/(N-m)c$;
- среднее время до завершения тестирования и его дисперсию $E(T_p)=\sum_{i=1}^{N-n}\frac{1}{ic}$,

  $Var(T_p)=\sum_{i=1}^{N-n}\frac{1}{(ic)^2}$.

При этом функция надежности вычисляется по формуле: $Rm(t)=\exp(-(N-m)ct)$, при$t>0$ и числе ошибок, найденных и исправленных на каждом интервале тестирования, равным единице.
Модель Шика-Вулвертона. Модель используется тогда, когдаинтенсивность отказовпропорциональна не только текущему числу ошибок, но и времени, прошедшему с момента последнего отказа. Исходные данные для этой модели аналогичны выше рассмотренной модели Джелински-Моранды:
m- число обнаруженных отказов за время тестирования,
X_i- интервалы времени между отказами,
T- продолжительность тестирования.
Функции риска $Z_i(t)$ в момент времени между*i*-1 и*i*=*m*отказами определяются следующим образом: $Z_i(t)=c(N-n_{i-1})$, где $i=1,\ldots, m$; $T_{i-1}<t<T_i$, $T=\sum_{i=1}^{m}X_i$.

Эта функция является линейной внутри каждого интервала времени между отказами, возрастает с меньшим углом наклона. Оценка *c* и *N* вычисляется из системы уравнений:
К выходнымпоказателям надежностиотносительно продолжительности*T*относятся:

- число оставшихся ошибок*Mm*=*N*-*m*;
- среднее время до следующего отказа *MTt*=(p/(2(*N*-*m*)*c*))1/2;
- среднее время до завершения тестирования и его дисперсия

– $E(T_p) = \sum_{i=1}^{N-m} \sqrt{\frac{\pi}{2ic}}$, $Var(T_p) = \sum_{i=1}^{N-m} \frac{2-\pi/2}{ic}$.

Функция надежности вычисляется по формуле: $R_T(t) = \exp(-\frac{(N-m)ct^2}{2})$, $t \geq 0$.

Модели пуассоновского типабазируются на выявлении отказов и моделируются неоднородным процессом, который задает $\{M(t), t \geq 0\}$- неоднородный пуассоновский процесс с функцией интенсивности $\lambda(t)$, что соответствует общему количеству отказов ПС за время его использования *t*.

Модель Гоело-Окумото. В основе этой модели лежит описание процесса обнаружения ошибок с помощью неоднородного пуассоновского процесса, ее можно рассматривать как модель экспоненциального роста. В этой моделиинтенсивность отказовтакже зависит от времени. Кроме того, в ней количество выявленных ошибок трактуется как случайная величина, значение которой зависит от теста и других условных факторов.

Исходные данные этой модели:

m- число обнаруженных отказов за время тестирования;

X_i- интервалы времени между отказами;

T- продолжительность тестирования.

Функция среднего числа отказов, обнаруженных к моменту*t*, имеет вид $m(t)=N(1-e^{-bt})$, где*b*- интенсивность обнаружения отказов и показатель роста надежности$q(t)=b$.

Функция интенсивности $\lambda(t)$ в зависимости от времени работы до отказа равна $\lambda(t) = Nb^{-b}$, $t \geq 0$.

Оценка*b*и*N*получаются из решения уравнений: $\frac{m}{N} - 1 + \exp\{(-bT)\} = 0$,

$\frac{m}{b} - \sum_{i=1}^{m}\{t_i\} - N_m \exp\{(-bT)\} = 0$.

Выходныепоказатели надежностиотносительно времени*T*определяют:

– среднее число ошибок, которые были обнаружены в интервале [0, *T*], по формуле$E(N_t)=N_{exp}(-bT)$,
– функцию надежности $R_T(t)=exp(N(e^{-bt}-e^{-bt(t+T)}))$, $t \geq 0$.

В этой модели обнаружение ошибки трактуется как случайная величина, значение которой зависит от теста и операционной среды.

В других моделях количество обнаруженных ошибок рассматривается как константа.

В моделях роста надежности исходной информацией для расчета надежности являются интервалы времени между отказами тестируемой программы, число отказов и время, для которого определяется надежность программы при отказе. На основании этой информации по моделям определяютсяпоказатели надежностивида:

– вероятностьбезотказнойработы;
– среднее время до следующего отказа;
– число необнаруженных отказов (ошибок);
– среднее время дополнительного тестирования программы.

Модель анализарезультатовпрогона тестовиспользует в своих расчетах общее число экспериментов тестирования и число отказов. Эта модель определяет только

вероятностьбезотказнойработы программы и выбрана для случаев, когда предыдущие модели нельзя использовать (мало данных, некорректность вычислений). Формула определения вероятностибезотказнойработы по числу проведенных экспериментов имеет видP=1-Nex/N, гдеNex- число ошибочных экспериментов, N- число проведенных экспериментов для проверки работы ПС.
Таким образом, можно сделать вывод о том, что модели надежности ПС основаны на времени функционирования и/или количестве отказов (ошибок), полученных в программах в процессе их тестирования или эксплуатации. Модели надежности учитывают случайный марковский и пуассоновский характер соответственно процессов обнаружения ошибок в программах, а также характер иинтенсивность отказов.

Кейс-задача
Рассчитать метрики качества программного продукта
Оценка размера программ есть оценка по номинальной шкале, на основе которой определяются только категории программ без уточнения оценки для каждой категории. К данной группе оценок можно отнести метрику **Холстеда**. Основа метрики - четыре измеряемые характеристики программы:

- NUOprtr (Number of Unique Operators) - число уникальных операторов программы, включая символы - разделители, имена процедур и знаки операций (словарь операторов);
  - NUOprnd (Number of Unique Operands) - число уникальных операндов программы (словарь операндов);
  - Noprtr (Number of Operators) - общее число операторов в программе;
  - Noprnd (Number of Operands) - общее число операндов в программе.

Операнд – это специальная величина, которая обрабатывается в программе. Оператор – это некоторое действие, которое выполняется с операндом.
Например: NUOprtr=27; NUOprnd=9; Noprtr=364; Noprnd=36.
Опираясь на эти характеристики, получаемые непосредственно при анализе исходных текстов программ, М. Холстед вводит следующие **оценки**:

- **Словарь программы** (HalsteadProgramVocabulary) HPVoc = NUOprtr + NUOprnd, HPVoc = 27 + 9 =36;
- **Длина программы** (Halstead Program Length) HPLen = Noprtr + Noprnd, HPLen = 364 + 36 = 400;
- **Объем программы** (Halstead Program Volume) HPVol = HPLen log2 HPVoc, HPVol = 400 log2 36 = 400*6 = 2400.

ДалееХолстед вводит **оценку сложность программы** (Halstead Difficulty), которая вычисляется как HDiff = NUOprtr/2* (NOprnd / NUOprnd), HDiff = 27/2* (36 / 9)= 13,5*4=54.
Используя HDiff Холстед вводит оценку HEff (HalsteadEffort): HEff = HDiff* HPVol, с помощью, которой описывается усилия программиста при разработке, HEff = 54* 2400=129600.
Метрики сложности потока управления программ. Как правило, с помощью этих оценок оперируют либо плотностью управляющих переходов внутри программ, либо взаимосвязями этих переходов. И в том и в другом случае программа представляется в виде управляющего графа. Впервые графическое представление

программ было предложено **Маккейбом.** Основной метрикой сложности он предполагает считать цикломатическую сложность графа программы, или, как ее еще называют, **цикломатическое число Маккейба**, характеризующее трудоемкость тестирования программы.

Для вычисления цикломатического числа Маккейба CC (Cyclomatic Complexity) применяется формула:CC =L – N + 2P,где L – число дуг ориентированного графа;
N – число вершин;P – число компонентов связности.

К метрикам сложности также относится:

- NORM (Number of Remote Methods) – **количество вызываемых удаленных методов**. При формировании значения этой метрики просматриваются все конструкторы и методы класса, и подсчитывается количество вызываемых удаленных методов. Удаленным методом является метод, который не определен в классе и его родителях.
- RFC (Response for Class) – отклик на класс — **количество методов, которые могут вызываться экземплярами класса**. Эта метрика вычисляется как сумма количества локальных методов и количества удаленных методов.
- WMPC1 (Weighted Methods per Class 1) – **взвешенная насыщенность класса** – дает относительную меру его сложности; если считать что все методы имеют одинаковую сложность, то это будет просто число методов в классе. Эта метрика определяется суммой сложности всех методов класса, где каждый метод взвешивается подсчетом его цикломатического числа. Количество методов и их сложность связаны для определения времени и усилий, которые потребуются для разработки и поддержки класса.
- WMPC2 (Weighted Methods per Class 2). Эта метрика является **мерой сложности класса, полагающая, что класс с большим количеством методов, чем другой, является более сложным**, и что метод с большим количеством параметров также является более сложным. Для расчета данной метрики используются только методы определенные в конкретном классе, все методы, наследуемые от родительского класса, не включаются.
- LOCOM1 (Lack of Cohesion of Methods 1) – **недостатоксвязностиметодов.** Связность – это степень взаимодействия между элементами отдельного модуля, характеристика его насыщенности.

Наименее желательной является связность по случайному принципу, когда в одном модуле собираются совершенно независимые абстракции. Связанность объектов — мера их взаимозависимости. Разработчики стремятся спроектировать не зацепленные (то есть слабо связанные) объекты, поскольку они имеют больше шансов на повторное использование.

Для каждой пары методов класса определяется набор полей, к которым они имеют доступ. Если множество полей имеет доступ к непересекающемуся множеству полей класса, то число P увеличивается на 1. Если оба метода используют хотя бы одно общее поле, то параметр Q увеличивается на 1. После рассмотрения каждой пары методов результат вычисляется как RESULT = (P > Q) ? (P — Q) : 0. Высокое значение этой метрики говорит о высокой связности методов — это означает, что потребуются большие усилия при тестировании этих методов, так как методы могут воздействовать на одни и те же атрибуты класса. Это также говорит о низкой

готовности к повторному использованию. Метрика была определена Чидамбером и Кемерером (Chidamber & Kemerer) в 1993.

- LOCOM2 (LackofCohesionofMethods 2) – связанность методов — **мера насыщенности абстракции**. Класс, который может вызывать существенно больше методов, чем равные ему по уровню классы, является более сложным. У класса с низкой связанностью можно подозревать случайную или неподходящую абстракцию: такой класс должен быть переабстрагирован на несколько классов или его обязанности должны быть переданы другим существующим классам. Метрика подсчитывает процентное отношение методов, не имеющих доступа к специфичным атрибутам класса ко всем атрибутам данного класса.
- LOCOM3 (LackofCohesionofMethods 3) – измеряет **степень различия методов в классе по атрибутам**.

Пример:

Рассмотрим множество методов M_1, M_2, ... , M_m. Эти методы имеют доступ к набору атрибутов данных класса A_1, A_2, ... , A_a. Положим $a(M_k)$ = число атрибутов с которым работает метод M_k , а $m(A_k)$ = число методов которые имеют доступ к атрибуту A_k. Тогда LOCOM3 = (1/a * SUM1a(m(Ai)-m))/(1-m)*100. Низкое значение этой меры говорит о хорошей декомпозиции в классе выражаемой в его простоте и понятности и готовности к повторному использованию. Высокое значение отсутствия связности увеличивает сложность, повышает вероятность ошибок в процессе разработки.

В эту группу метрик также попадают базовые метрики:

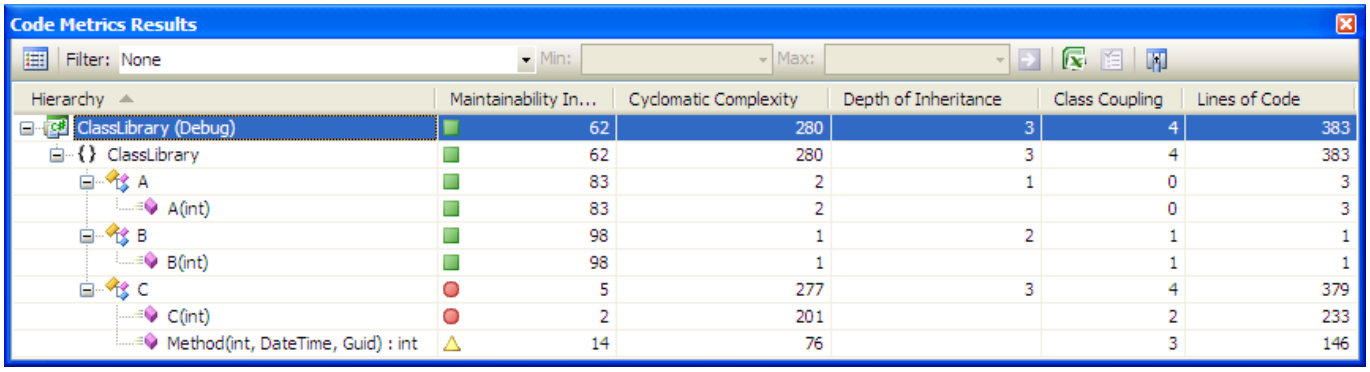
- LOC (Lines Of Code) – количество строк кода
- NOC (Number Of Classes) – количество классов

Улучшение значений метрических характеристик из этой группы говорит о положительном эффекте от применения оптимизирующего метода/подхода. Например, уменьшение значений метрик Холстеда, цикломатической сложности говорит о том, что полученная реализация в целом лучше по соображениям топологической сложности. Уменьшение уровня связности, взвешенности метода на класс или отклика класса говорит об улучшенной функциональной декомпозиции. Уменьшение количества строчек кода также говорит о положительном эффекте, если ту же функциональность можно изложить при помощи нового улучшающего подхода за меньшее количество строк.

или

Например.

Средство оценки качества кода в MS Visual Studio:



Результаты содержат 5 метрик для вашего кода.
Maintainability Index – комплексный показатель качества кода. Этот показатель разработан специалистами из Carnegie Mellon Software Engineering Institute. Рассчитывается метрика по следующей формуле: MI = MAX(0, (171 — 5.2 * ln(HV) — 0.23 * CC — 16.2 * ln(LoC)) * 100 / 171)
HV – Halstead Volume, вычислительная сложность. Чем больше операторов, тем больше значение этой метрики;
CC – Cyclomatic Complexity. Эта метрика описана ниже;
LoC – количество строк кода.
Эта метрика может принимать значения от 0 до 100 и показывает относительную сложность поддержки кода. Чем больше значение этой метрики, тем легче поддерживать код.
Visual Studio помечает методы/классы зеленым цветом, если значение метрики находится в пределах от 20 до 100, желтым цветом, если значение находится в пределах от 10 до 20, и красным цветом, когда значение меньше 10.
Cyclomatic Complexity – показывает структурную сложность кода, т.е. количество различных ветвей в коде. Чем больше этот показатель, тем больше тестов должно быть написано, для полного покрытия кода.
Depth of Inheritance – глубина наследования. Эта метрика показывает для каждого класса, какой он по счету в цепочке наследования. Например, есть 3 класса A, B, C, B унаследован от A, а C унаследован от B, то значение этой метрики для классов A,B и C будет равно соответственно 1, 2 и 3.
Class Coupling – показывает степень зависимости классов друг с другом. В расчет берутся уникальные классы из параметров, локальных переменных, возвращаемого типа, базового класса, атрибутов. Хороший дизайн программного обеспечения предполагает небольшое количество связанных классов. Чем их больше, тем сложнее в дальнейшем переиспользовать этот класс, а также поддерживать, т.к. существует очень много зависимостей.
Lines of Code – показывает количество строк кода. Этот показатель показывает не точное количество строк в вашем файле, т.к. подсчет основан на IL-коде. В расчет не берутся пустые строчки, комментарии, строчки со скобками, объявление типов и пространств имен. Большое количество строк в методе/классе может показывать на ошибки в проектировании и на то, что этот код можно разделить на несколько частей.

9. Материально-техническое обеспечение дисциплины

- лекционные аудитории с современными средствами демонстрации 6-415, 6-416, 6-213.

- кафедральные лаборатории, обеспечивающих реализацию ОПОП ВО: 6-218 Учебно-научная лаборатория «Технологии искусственного интеллекта в социально-экономических исследованиях, 6-417 Лаборатория информатики и программирования, 6-417а Учебно-научная лаборатория «Интеллектуальных технологий проектирования сложных систем».

Оборудование

Компьютеры IntelPentiumDual-CoreE5300 BOX 2.6 Гц/ASUSTekP5KRL-AM (RTL) Socket775/2x1GbSATA – 5 к.;

Компьютеры LGA 1155/5000/IntelH61/DDR3/4/SATA+4/ОЗУ DDR-IIIDIMM 2 Gb/HDD 500Gb/DVDRAM&DVD+R/RW/ATX 450W – 5 к.;

Компьютер Pentium 4 631 3,0ГГц BOX/ASUSTEKP5G-MXSocket775/2xDDR 1/160 SATA-IIWD/FDD 3.5 HD – 5 к.;

Состав комплекта (3):

Блок системный IntelCore 2 DuoE7400 BOX/2XDDR-IIDIMM

Монитор 17” Acer V173 Ab/BB/DB

Клавиатура PS/2

Мышь PS/2.

10. Адаптация рабочей программы для лиц с ОВЗ

При инклюзивном обучении лиц с ОЗВ предоставляется возможность использовать следующие материально-технические средства:

- для студентов с ОВЗ по зрению предусматривается применение средств преобразования визуальной информации в аудио и тактильные сигналы, таких как, брайлевская компьютерная техника, электронные лупы, видеоувеличители, программы невизуального доступа к информации, программы-синтезаторов речи;
- для студентов с ОВЗ по слуху предусматривается применение сурдотехнических средств, таких как, системы беспроводной передачи звука, техники для усиления звука, видеотехника, мультимедийная техника и другие средства передачи информации в доступных формах;
- для студентов с нарушениями опорно-двигательной функции предусматривается применение специальной компьютерной техники с соответствующим программным обеспечением, в том числе, специальные возможности операционных систем, таких, как экранная клавиатура и альтернативные устройства ввода информации.

ЛИСТ

согласования рабочей программы

Направление подготовки: **09.06.01 Информатика и вычислительная техника**
код и наименование

Направленность подготовки (программа): **Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей**
наименование

Дисциплина: Технологии разработки программного обеспечения

Учебный год 2014/2015

РЕКОМЕНДОВАНА заседанием кафедры Вычислительной математики и кибернетики
наименование кафедры

протокол № 15 от "25" июня 2015 г.
Заведующий кафедрой _____ Н. И. Юсупова
подпись расшифровка подписи

Исполнители: проф. _____ О.Н. Сметанина
должность подпись расшифровка подписи

СОГЛАСОВАНО:

Заведующий кафедрой[1]
Вычислительной математики
и кибернетики _____ Н. И. Юсупова _____
наименование кафедры *личная подпись* *расшифровка подписи* *дата*

Председатель НМС по УГСН _____ Ишмуратов А.Ч.
протокол № 3 от "28" августа 2015 г.
09.00.00 Информатика и вычислительная техника
личная подпись *расшифровка подписи*

Библиотека директор _____ С.Ф. Шагафиева 28.08.15
личная подпись *расшифровка подписи* *дата*

Начальник отдела аспирантуры _____ Долгаков Р.К. 29.08.15
личная подпись *расшифровка подписи* *дата*

Рабочая программа зарегистрирована в ООПМА и внесена в электронную базу данных
Начальник _____ Лаптиан И.А. 31.08.15
личная подпись *расшифровка подписи* *дата*

[1] Согласование осуществляется с выпускающими кафедрами (для рабочих программ, подготовленных на кафедрах, обеспечивающих подготовку для других направлений и специальностей)

43