Министерство науки и высшего образования РФ
ФГБОУ ВО «Ульяновский государственный университет»
Факультет математики, информационных и авиационных технологий

Волков М.А.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ ПО ДИСЦИПЛИНЕ

«Открытые технологии разработки программного обеспечения»

для студентов магистратуры по направлению 02.04.03 – «Математическое обеспечение и администрирование информационных систем»

Ульяновск, 2019

Методические указания для самостоятельной работы студентов по дисциплине «Открытые технологии разработки программного обеспечения» для студентов магистратуры по направлению 02.04.03 – «Математическое обеспечение и администрирование информационных систем» / составитель: Волков М.А. – Ульяновск: УлГУ, 2019.

Настоящие методические указания предназначены для студентов магистратуры по направлению 02.04.03 – «Математическое обеспечение и администрирование информационных систем». В работе приведены литература по дисциплине и методические указания для самостоятельной работы студентов, которые будут полезны при подготовке к занятиям и к аттестации по данной дисциплине.

Рекомендованы к введению в образовательный процесс Ученым советом Факультета математики, информационных и авиационных технологий УлГУ (протокол № 2/19 от 19 марта 2019 г.).

1. ЛИТЕРАТУРА ДЛЯ ИЗУЧЕНИЯ ДИСЦИПЛИНЫ

1) Зубкова Т.М. Технология разработки программного обеспечения [Электронный ресурс]: учебное пособие / Т.М. Зубкова. - Электрон. текстовые данные. - Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017. - 469 с. - 978-5-7410-1785-2. - Режим доступа: http://www.iprbookshop.ru/78846.html
2) Влацкая И.В. Проектирование и реализация прикладного программного обеспечения [Электронный ресурс]: учебное пособие/ Влацкая И.В., Заельская Н.А., Надточий Н.С.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2015.— 119 с.— Режим доступа: http://www.iprbookshop.ru/54145.html
3) Носова Л.С. Case-технологии и язык UML : учебно-методическое пособие / Носова Л.С.. — Челябинск, Саратов : Южно-Уральский институт управления и экономики, Ай Пи Эр Медиа, 2019. — 67 c. — ISBN 978-5-4486-0670-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/81479.html
4) Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем: учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 432 с. — (Бакалавр. Академический курс). — ISBN 978-5-534-07604-2. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/436514
5) Жаркова Галина Алексеевна. Программирование на языке С++ : учеб. пособие для вузов / Жаркова Галина Алексеевна. - Ульяновск : УлГУ, 2009. - Загл. с экрана; Имеется печ. аналог. - Электрон. текстовые дан. (1 файл : 729 Кб). - Текст : электронный. http://lib.ulsu.ru/MegaPro/Download/MObject/651
6) Жаркова Г. А. Современные системы автоматизации разработки информационных систем : учеб.-метод. пособие / Г. А. Жаркова; Ульяновск. гос. ун-т, Ин-т математики и информ. технологий, Каф. информ. технологий. - Ульяновск : УлГУ, 2007. - Загл. с экрана; Имеется печ. аналог. - Электрон. текстовые дан. (1 файл : 606 Кб). - Текст : электронный. http://lib.ulsu.ru/MegaPro/Download/MObject/652
7) Жаркова Г.А. Методы программирования и прикладные алгоритмы: учеб.-метод. пособие / Жаркова Г.А., А. В. Жарков; УлГУ, ФМИиАТ. - Ульяновск: УлГУ, 2018. - 96 с.
8) Филаткина Елена Владимировна. Экономико-правовые основы рынка программного обеспечения : учеб. пособие для студентов фак. математики и информ. технологий / Филаткина Елена Владимировна; УлГУ, ФМИТ, Каф. информ. технологий. - Ульяновск : УлГУ, 2012. - Имеется печ. аналог. - Электрон. текстовые дан. (1 файл : 551 КБ). - Текст : электронный. http://lib.ulsu.ru/MegaPro/Download/MObject/736

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Ниже приведены методические указания для самостоятельной подготовки к занятиям по дисциплине.

ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа № 1. *Разработка диаграммы вариантов использования для прототипа оболочки ДЭС.*

Цель работы – овладение навыками построения диаграммы вариантов использования на примере оболочки ДЭС.

Общие сведения.

ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ.

Диаграмма вариантов использования является исходным концептуальным представлением (моделью) системы в процессе проектирования и разработки. Она описывает функциональное назначение системы и завершает анализ предметной области, когда определились требования к функциональному поведению проектируемой системы. Эта диаграмма будет в дальнейшем детализироваться в форме логических и физических моделей. Она также станет основой взаимодействия разработчиков и заказчика и войдёт в состав документации по системе.

При построении диаграммы фиксируется множество сущностей или актёров, взаимодействующих с системой по установленным правилам, которые называют теперь вариантами использования. Актёром может быть человек, техническое устройство, программа или какая-либо другая система. Все они являются источниками взаимодействия. В свою очередь, вариант использования служит для ожидания сервисов, которые система предоставляет актеру, т. е. наборов действий, совершаемых системой при диалоге с актёром. Способ реализации действий не уточняется. Визуально диаграммы вариантов использования представляет собой граф специального вида с указанием актёров, вариантов использования, интерфейсов и отношений между ними. Диаграмма вариантов может дополняться пояснительным текстом, который раскрывает смысл составляющих ее компонентов, и называется примечанием или сценарием.

Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его краткое название или выполняемое действие в форме глагола с необходимыми уточнениями:

Цель варианта использования заключается в том, чтобы определить законченный аспект или фрагмент поведения некоторой сущности без раскрытия ее внутренней структуры. Это означает, что после выполнения этого действия, система должна возвратиться в исходное состояние для выполнения следующих запросов. Каждый выполняемый вариантом использования метод реализуется как неделимая транзакция. При дальнейшей детализации модели возможны изменения в том или ином варианте использования, т. е. может иметь место обратная связь.

Стандартным графическим обозначением актера на диаграммах является следующая фигура с надписью: Пациент.

Актеры взаимодействуют с системой посредством передачи и приема сообщений от вариантов использования. Сообщение представляет собой запрос актера на сервис системы. Это взаимодействие отображается на диаграмме в виде ассоциации актёра и варианта использования. Кроме этого, с актерами могут быть связаны интерфейсы, которые определяют, каким образом другие элементы взаимодействуют с актерами.

Таким образом, интерфейс служит для спецификации параметров модели. Которые видимы извне без указания их внутренней структуры. Он не может содержать ни атрибутов, ни состояний, ни направленных ассоциаций. В нем перечисляются только

операции без указания особенностей их реализации. Формально интерфейс эквивалентен абстрактному классу без атрибутов и методов с наличием только абстрактных операций. Графически представляются малым кругом с надписью: Термометр, Вопрос пациенту, Прибор анализа, Анализ спирта.

Интерфейс связывается с вариантом использования сплошной линией без стрелок или пунктирной линией со стрелкой. В последнем случае вариант использования предназначен для спецификации только того сервиса, который необходим для реализации данного интерфейса. В первом случае интерфейс может иметь и другие функции.

Важность интерфейсов заключается в том, что они определяют стыковочные узлы в проектируемой системе, что необходимо для коллективной работы над проектом, при этом они не зависят от способов реализации системы.

Любой элемент диаграммы может иметь пояснения, которые помещаются в прямоугольник на диаграмме с загнутым верхним правым уголком. Этот прямоугольник соединяют пунктирной линией с соответствующим элементом диаграммы. Для записи ограничений на проектируемую систему используется ключевое слово constract, после которого в фигурных скобках на языке OCL записывают требуемое ограничение.

Между компонентами диаграммы могут существовать различные отношения, которые описывают взаимодействие экземпляров одних актеров и вариантов использования с экземплярами других актеров и вариантов. Один актер может взаимодействовать с несколькими вариантами использования. В свою очередь один вариант использования может взаимодействовать с несколькими актерами, представляя для все них свой сервис.

В языке UML имеются следующие виды отношений между актерами и вариантами использования:

1. Отношение ассоциации, которое определяет специфическую роль актера в отдельном варианте использования. На диаграмме оно обозначается сплошной линией с названием. На концах линии может указываться кратность. Кратность характеризует общее количество конкретных экземпляров данного компонента, которые могут выступать в качестве элементов данной ассоциации: 5; 1…5; 2…*; *.

2. Отношение расширения, которое определяет взаимосвязь экземпляров отдельного элемента использования с более обширным вариантом. На диаграмме оно обозначается пунктирной линией со стрелкой от более общего варианта к менее общему с надписью “расширяет”.

3. Отношение обобщения, которое служит для указаний того факта, что некоторый вариант использования А может быть обобщен до варианта использования В. Например, вариант использования “Оформить заказ на приобретение компьютера” может быть обобщен до варианта “Оформить заказ на приобретение товара”. На диаграмме отношение обозначается сплошной линией со стрелкой, указывающей на родительский, т.е. более общий вариант использования. Здесь возможно множественное наследование свойств и поведения отчуждений предков. Такое отношение обобщения может существовать и между отдельными актерами.

4. Отношение включения, которое определяется между двумя вариантами использования и указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. На диаграмме обозначается пунктирной линией со стрелкой и надписью “включает”. Например, варианты использования “Оформить заказ на приобретение компьютера” включает вариант “Выписать счет на оплату компьютера”. Это отношение не следует путать с отношением расширяет, например, вариант использования “ Запросить каталог всех товаров” расширяет вариант “Оформить заказ на приобретение товара”.

В качестве примера приведем диаграмму использования для системы продажи товаров по каталогу. На диаграмме приведены все актеры, варианты использования и

отношения между компонентами диаграммы. Главное назначение диаграммы использования заключается в формализации функциональных требований к системе и возможности согласования полученной модели с заказчиком на ранней стадии проектирования. Не более 20 актеров и 50 вариантов.

Практическая часть работы.

1. Ознакомиться с примером диаграммы вариантов использования, приводимой в приложении.
2. Внимательно изучить графическую нотацию для построения диаграмм вариантов использования.
3. Используя справочную систему ДЭС и ее работающий прототип, определить набор физических компонент и интерфейсов, а также установить имеющиеся между ними зависимости и связи.
4. Разработать структуру диаграммы вариантов использования оболочки ДЭС.
5. Дополнить модель интерфейсами и используемыми базами данных с необходимыми и информационными связями.
6. Нанести на диаграмму взаимосвязи и отношения реализации.
7. При необходимости использовать дополнительные стереотипы, механизмы расширения и помеченные значения.
8. Произвести проверку правильности построения диаграммы вариантов использования и предоставить преподавателю.

Лабораторная работа № 2. *Разработка диаграммы классов для прототипа оболочки ДЭС.*

Цель работы – овладение навыками построения диаграммы классов на примере оболочки ДЭС.

Общие сведения.

ДИАГРАММА КЛАССОВ.

Диаграмма классов служит для представления статической структуры модели системы в терминах классов объектно-ориентированного программирования. Она представляет собой некоторый граф, вершинами которого являются элементы типа “классификатор”, которые связаны различными типами структурных отношений. На диаграмме могут бы также интерфейсы, пакеты и даже объекты классов, при этом пакеты используются для представления более общей модели системы.

Графически класс изображается в виде прямоугольника с указанием имени класса и, возможно, списка атрибутов, а также операций. Атрибуты и операции образуют секции и отделяются друг от друга и от имени горизонтальными линиями. При проектировании в прямоугольнике сначала записывается имя класса, а атрибуты и операции записываются по мере проработки программной системы. Иногда используется и четвёртая секция для указания исключительных ситуаций или определения семантики класса.

Имя класса должно отражать назначение класса и записываться на английском или русском языке с большой буквы. Имя абстрактного класса записывается курсивом. Каждый атрибут класса имеет определённую область видимости:

+ общедоступный (public);

защищённый (ptotected);

- закрытый (private).

Иногда область видимости совсем не указывается. Квантор видимости ставится перед именем атрибута. После имени атрибута 38 указывается его кратность в классе и тип. Кратность задаётся диапазоном целых чисел, разделённых двумя точками. Верхняя граница диапазона может быть звездочкой, которая означает произвольное целое число. Тип атрибута соответствует типу языка реализации.

Операция класса представляет некоторый сервис, представляемый данным классом. В записи операции указывается квантор видимости, имя операции, список

параметров в круглых скобках, а также тип возвращаемого значения. В фигурных скобках далее можно указать тип операции:

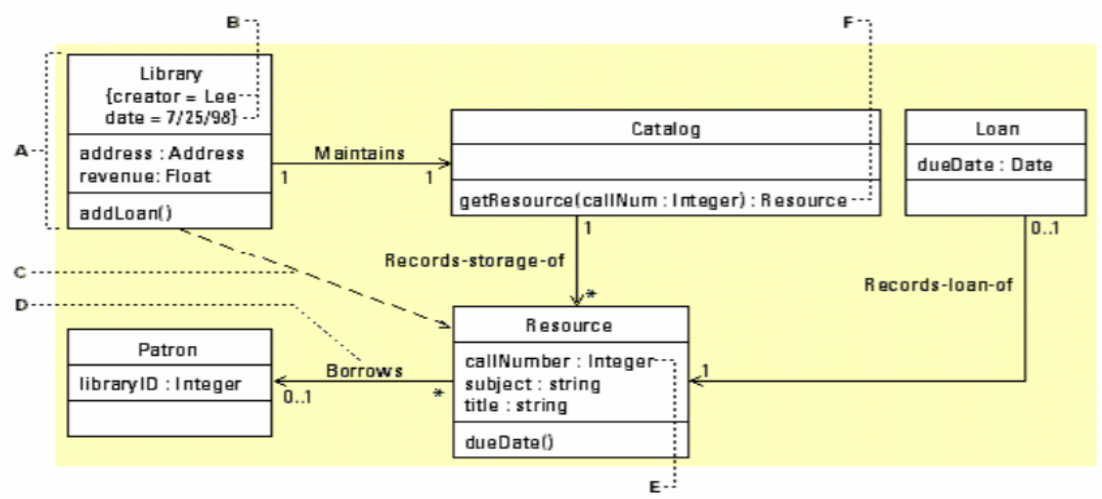
{ concurreney = sequential } - последовательная;

{ concurrency = concurrent } - возможно распараллеливания;

{ concurrency = guarded } - охраняемая.

Между классами могут быть отношения зависимости (---Æ), ассоциации (-), агрегации(<>-), обобщения(<-).

Пример диаграммы классов:



Практическая часть работы.

1. Ознакомиться с примером диаграммы классов, приведённой в Приложении А.
2. Внимательно изучить графическую нотацию для построения диаграмм классов.
3. Используя справочную систему ДЭС и приложение В, определить используемые в оболочке классы, их атрибуты и методы, а также установить имеющиеся между ними и объектами зависимости и отношения.
4. Разработать структуру диаграммы классов оболочки ДЭС.
5. Дополнить каждый класс необходимыми атрибутами и операциями.
6. Нанести на диаграмму взаимосвязи и отношения между классами и объектами.
7. При необходимости использовать дополнительные стереотипы, механизмы расширения, помеченные значения и интерфейсы.
8. Произвести проверку правильности построения диаграммы классов.
9. Сгенерировать реализацию классов прототипа оболочки ДЭС на языке C++.
10. Создать приложение на Visual C++ 6.0 и включить в него сгенерированные исходные коды классов.
11. Отладить приложение в среде Visual C++ 6.0.

3. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Раздел 1. Жизненный цикл программного обеспечения

Тема № 1. Почему программному обеспечению присуща сложность. Почему программному обеспечению присуща сложность. Сложность реальной предметной области, сложность описания поведения больших дискретных систем, сложность управления коллективом разработчиков. Проблемы, возникающие при общении с заказчиками программных систем. Сложность оценки качества программного обеспечения.

Тема № 2. Жизненный цикл программного обеспечения. Жизненный цикл программного обеспечения. Распределение финансовых и временных затрат на реализацию каждого из этапов разработки программного обеспечения.

Раздел 2. Технологии разработки программного обеспечения.

Тема № 3. Выявление требований к программной системе. Работа с заказчиком. Обследование системы, общение с заказчиком, планирование разработки, составление технического задания. Детальный анализ предметной области, принятие окончательного решения о необходимости создания информационной системы, проектирование общей архитектуры системы, выбор метода проектирования.

Тема № 4. Обзор методологий проектирования программных продуктов. Каскадные и итеративные технологии. Критичность и масштабность программных проектов.

Тема № 5. Технологии быстрой разработки программного обеспечения. Технология экстремального программирования. SCRUM технология. Преимущества и недостатки технологий быстрой разработки программного обеспечения. Организация коллективной работы над проектом при использовании технологий быстрой разработки.

Тема № 6. Объектно-ориентированное проектирование программной системы. Построение объектно-ориентированной архитектуры системы. Методы объектно-ориентированного анализа для выявления классов и объектов. CASE-средства объектно-ориентированного проектирования.

Тема № 7. Средства информационной поддержки программных проектов и изделий (CALS) технологий. Средства управления проектами. Применение данных средств при разработке и сопровождении программных продуктов. Использование средств коллективного владения кодом при создании корпоративных информационных систем.

Тема № 8. Тестирование и отладка программных систем. Стратегии и методы тестирования. Прямое и обратное тестирование. Программные средства автоматизации тестирования.

Тема № 9. Оценка качества программного обеспечения. Методики оценки качества ПО. Процессный подход к оценке качества ПО.

Тема № 10. Внедрение и сопровождение программных продуктов. Планирование процесса внедрения программного продукта. Основные задачи решаемые на этапе внедрения. Процесс устранения ошибок на этапе внедрения. Техническая поддержка пользователей на этапе сопровождения.

ПЕРЕЧЕНЬ ВОПРОСОВ К ЭКЗАМЕНУ

1. Что такое промышленный программный продукт? Дать определения пакета прикладных программ, программной системы.
2. Жизненный цикл программного обеспечения. Дать краткую характеристику каждого этапа.
3. Почему программные системы сложны. Привести пять признаков сложной системы.
4. Техническое задание. Перечислить и охарактеризовать разделы, входящие в техническое задание.
5. Унифицированный процесс разработки программного обеспечения. Жизненный цикл унифицированного процесса.
6. Работа с кадрами. Перечислить роли разработчиков и дать характеристику каждой из них.
7. Дать определения проекта, процесса, продукта с точки зрения унифицированного процесса разработки программного обеспечения.
8. Что такое артефакт. В чем преимущества организованного процесса разработки программного обеспечения.
9. Использование языка UML при проектировании сложных программных систем. Какие диаграммы используются в UML для создания моделей программной

системы.

10. Диаграмма вариантов использования, ее назначение. Рассказать о варианте использования и действующем лице. Правила построения диаграммы вариантов использования.

11. Понятие класса и объекта. Что может быть объектом. Что такое атрибут и операция.

12. Пять критериев проверки правильности построения класса.

13. Что такое классификация с точки зрения объектно-ориентированного проектирования программных систем. Теории классификации.

14. Методы классификации.

15. Микропроцесс проектирования. Перечислить этапы и основные виды деятельности, выполняемые на каждом из них.

16. Микропроцесс проектирования – первый этап.

17. Микропроцесс проектирования – второй этап.

18. Микропроцесс проектирования – третий этап.

19. Микропроцесс проектирования – четвертый этап.

20. Диаграммы взаимодействия. Основное назначение.

21. Диаграмма классов. Ее назначение. Что она включает. Рассказать об основных видах связей между классами.

22. Дать определение тестированию и отладке. Особенности и объекты тестирования. Автономное и комплексное тестирование.

23. Дать определение тестированию и отладке. Направления тестирования. Стратегия тестирования. Контрольный лист тестирования модуля.

24. Дать определение тестированию и отладке. Локализация ошибок. Классификация ошибок. Безопасное программирование.

25. Оценки ошибок.

26. Документирование. Состав и содержание документов прилагаемых к программной системе.

27. Внедрение программного комплекса. Планирование испытаний.

28. Внедрение программного комплекса. Подготовка тестовых данных. Анализ результатов испытаний.

29. Что такое качество с точки зрения квалиметрии. Дать определение свойству и показателю качества ПО. Основные задачи решаемые при оценке качества.

30. Оценка качества программного обеспечения. Методы оценки свойств программного обеспечения.