Абрамова О. Ф., Лясин Д. Н.

Методика применения принципов ролевого подхода для организации цикла лабораторных занятий

Учебно-методическое пособие

Волжский

2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ВОЛЖСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО
УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Абрамова О.Ф., Лясин Д. Н.

Методика применения принципов ролевого подхода для организации цикла лабораторных занятий

Электронное учебно-методическое пособие

2019

УДК 004.45(07)
ББК 32.81я73
А 161

Рецензенты:
заведующий кафедрой математики, информатики и естественных наук Волжского филиала Волгоградского государственного университета, канд. физ.-мат. наук, доцент
Полковников А. А.
заведующий кафедрой экономической теории, математики и информационных систем МБОУ ВО «Волжский институт экономики, педагогики и права»
Орехова Е. В.

Издается по решению редакционно-издательского совета
Волгоградского государственного технического университета

Абрамова, О.Ф.
Методика применения принципов ролевого подхода для организации цикла лабораторных занятий. [Электронный ресурс] : учебно-методическое пособие/ О.Ф. Абрамова, Д. Н. Лясин ; ВПИ (филиал) ВолгГТУ, – Электрон. текстовые дан. (1 файл: 3,87 МБ). – Волжский, 2019. – Режим доступа: http://lib.volpi.ru. – Загл. с титул. экрана.
ISBN 978-5-9948-3240-0

Пособие представляет собой описание методики организации цикла лабораторных работ по проектированию и разработке программных продуктов с использованием принципов ролевого подхода.

Предназначено для студентов, обучающихся по направлению подготовки бакалавров 09.03.01 «Информатика и вычислительная техника» и рекомендуется для изучения дисциплины «Проектирование и разработка программного обеспечения».

Ил. 3, библиограф.: 15 назв.

ISBN 978-5-9948-3240-0

Оглавление

ВВЕДЕНИЕ

Область IT-технологий является одной из самых быстроразвивающихся отраслей, что в значительной степени усложняет процесс обучения и формирования высококлассных специалистов. Традиционные методы обучения, конечно, никто не отменял, но в данном случае использовать инновационные методы и методики просто необходимо [1, 2]. Особенно это касается обучения специалистов в области программной инженерии [3]. Разработка современных сложных программных систем ведется, как правило, коллективом разработчиков. Поэтому важно привить студенту не только навыки практические: программирования, моделирования систем, но и навыки коммуникации: общения с коллегами, ведения диалога с заказчиком, умения представить в выгодном свете свои разработки и т.д.[4].

В качестве основных методов обучения можно выделить три: пассивный, активный и интерактивный. И если первые два метода достаточно широко распространены в современном образовательном процессе, то последний – интерактивный метод – используется не так активно, так как требует дополнительных усилий, знаний и умений от преподавателя. Интерактивные технологии обучения направлены в большей степени на значительное повешение роли обучающегося в образовательном процессе, осуществление диалога не только между учителем и учеником, как предполагает классический подход к образованию, а в большей мере диалог между самими учащимися. Причем общение между обучающимися направлено не столько на усвоение уже полученных знаний, а на приобретение новых.

Реализацию интерактивных технологий для проведения лабораторных занятий в техническом вузе можно вкратце описать следующим образом. Основными специалистами в группе разработчиков программных систем (в обобщенном случае) можно считать аналитика, тестеровщика и, конечно, руководителя проекта. Поэтому одним из вариантов интерактивного подхода можно считать организацию выполнения лабораторных работ малой группой

студентов (3 человека). Причем каждый студент в такой группе играет определенную роль, соотносимую с ролью в настоящем профессиональном коллективе разработчиков программного обеспечения. Т.е. используются действенные, эффективные и актуальные методы: работа в малых группах, организация проведения занятий в форме деловой игры.

Еще один способ организации интерактивной ролевой игры в рамках лабораторных занятий – это формирование групп для выполнения очередного задания в соответствии с отделами полноценной фирмы по проектированию и разработке программного обеспечения. Например, отдел аналитики, отдел технических писателей (или постановщиков задач), отдел проектировщиков, отдел программистов и т.д. В случае, если за цикл лабораторных работ студенты должны полностью отработать основные этапы жизненного цикла программного продукта, что и требуется, например, в таких дисциплинах, как «Проектирование разработка программного обеспечения» и «Конструирование программного обеспечения», такой подход даже более предпочтителен, так как позволит формировать задачи для участников групп более однотипные и реально выполнимые для разного уровня подготовленности.

Процесс формирования рабочих групп так же важен, как и распределение ролей в этих группах. На основании различных исследований, можно рекомендовать формирование группы именно преподавателем, который должен исключить создание так называемых групп «соседей» [5], т.е. студентов, связанных дружескими отношениями вне учебных занятий. А так же максимально равномерно распределить по группам студентов с разным уровнем знаний, разной степенью обучаемости и не учитывать половой признак. Т.е. при комплектовании состава групп преподавателю не следует идти на поводу у обучающихся, требующих организовать группы «по желанию», что может свести все положительные качества технологии «на нет».

Все участники игры получают исчерпывающую информацию по каждому этапу выполнения учебного проекта как в устном, так в электронном виде. Причем информация им предлагается как общего характера (полезная всем), так и конкретизированная для определенной роли. Каждое требование подтверждается практическим примером. И, в конечном итоге, каждый студент получает описание структуры будущего отчета по лабораторной работе, содержащего как пункты, которые можно реализовать только общими усилиями группы, так и пункты, которые ему придется реализовывать самостоятельно. Но каждый пункт для самостоятельно проработки, во-первых, опирается на знания и информацию, полученные другими участниками группы, а во-вторых, должен быть разъяснен по окончании выполнения коллегам. Для выполнения общих пунктов достаточно будет знаний, полученных на лекционных и лабораторных занятиях от преподавателя, а так же выложенных в УМКД. Но выполнение индивидуальных пунктов потребует от студента дополнительных усилий по поиску и сортировке материала, изучению дополнительных, но важных для формирования образа мышления будущего специалиста в области программной инженерии тем, развитию и формированию способности формулировать краткие описания и делать выводы, а так же ясно и четко объяснять проделанную работу. Для повышения эффективности процесса с точки зрения получения профессиональных навыков важно использовать для практической реализации современные программные средства моделирования программных систем [2,4].

Отчет лабораторной работы производится только устно, всей группой, участвующей в разработке учебного проекта, что позволяет сымитировать реальное общение с заказчиком на производстве. Задача преподавателя: сформировать обстановку, максимально соответствующую реальной, максимально сводя проведение отчета к ситуации «аквариум» [5], когда отчет по разработке проекта проводится группой публично. При этом у обучающихся появляется возможность сравнения работ, анализа как личной,

групповой работы, так и достижений и ошибок других групп. Преподавателю важно не пускать данные моменты на самотек, а обращать внимание всех студентов на успехи их товарищей, предлагая взять на вооружение найденные и использованные методы и подходы.

Для повышения эффективности процесса с точки зрения получения профессиональных навыков важно использовать для практической реализации современные программные средства моделирования программных систем [2,4]. Например, BOUML – CASE-средство, предназначенное для автоматизации этапов анализа и проектирования программного обеспечения, а также для генерации кодов на различных языках и выпуска проектной документации. В основе работы BOUML лежит построение различного рода диаграмм и спецификаций, определяющих логическую и физическую структуры модели, ее статические и динамические аспекты.

Основными достоинствами BOUML являются:

- работает под Linux, MacOS X и Windows, что позволяет программировать одновременно в C + +, Java, PHP, Python и IDL;
- бесплатное распространение в сети Интернет (до версии 5.0);
- является расширяемым, а внешние инструменты (плагины, потому что они выполняются за пределами BOUML) могут быть разработаны в C + + или Java;
- обладает высоким быстродействием и не требует много памяти для управления несколькими тысячами классов.

В результате разработки проекта с помощью CASE-средства BOUML формируются следующие документы:

- диаграммы UML, в совокупности представляющие собой модель разрабатываемой программной системы;
- спецификации классов, объектов, атрибутов и операций;
- заготовки текстов программ.

Соответствовать таким требованиям невозможно без самостоятельного поиска дополнительных знаний и личной заинтересованности в результате. С другой стороны, присутствие некой общей основы в задании к лабораторной работе даст уверенность каждому студенту и освободит ему время для поиска действительно оригинальных идей и оптимизации уже существующих решений. А в качестве катализатора процесса самообразования студента можно предложить набор приятных «бонусов» (баллов) за дополнительные действия в процессе решения и оформления поставленной задачи. Например: разработка новых и оптимизация существующих алгоритмов решения, нахождение нетривиальных, интересных решений, оформление как программного продукта (реализация доступного, красочного и понятного интерфейса), так и документации и т.п.

Такой подход к организации лабораторных занятий можно применить, практически, в любой технической дисциплине, предполагающей в качестве итогового контроля знаний разработать некий программный продукт.

Предлагаемая в данном учебно-методическом пособии методика проведения лабораторных занятий подразумевает организацию занятий по определенному плану, описанному ниже, и перечень требований к участникам группы, которые подробно описаны в методических указаниях к лабораторной работе №1. В остальных работах приводятся только дополнения либо уточнения при необходимости.

Метод «мозговой штурм»

Концепция мозгового штурма, позволяющая быстро генерировать идеи, не оценивая и не преобразуя их, в литературе описана достаточно подробно.

Рассмотрим подробно правила проведения сеансов «мозгового штурма».

Роли во время сеансов

Каждый участник исполняет одну из ролей:

- *Лидер* (арбитр, председатель). В его обязанности входит слушать и направлять процесс в правильное русло. Он обеспечивает объявленный порядок, если нужно, начинает процесс генерирования идей, помогает секретарю в записи идей.
- *Секретарь*. Записывает все идеи таким образом, чтобы записи были видны всем участникам, например, используя плакаты, доски, проекторы. В больших группах может потребоваться несколько секретарей.
- *Участник*. Обычно это несколько организаторов проекта, имеющие различные точки зрения. Главная задача участников – это генерирование требований и стимулирование к высказыванию своих мыслей других участников сеанса. Число участников может быть достаточно большим, но наилучшие результаты достигаются в группах из 6-7 человек.

Правила проведения сеанса

1. *Общие правила.*

Основные правила проведения сеансов «мозгового штурма»:

- Принимаются любые идеи, все идеи потенциально хороши. Важно *количество*, а не качество идей.
- Идея принадлежит всей группе, а не только тому, кто ее высказал.
- Ценной является каждая точка зрения, т.к. уникальный подход к ситуации позволяет сформировать различные решения.

- В группы включаются только те люди, которые могут быть полезны.
- Обсуждение идей происходит после завершения сеанса.

2. *Правила для участников сеанса.*

Правила для лидера:

- Идеи должны высказываться устно или в виде записок.
- Объявите участникам основные правила сеанса.
- Определите цель или тему сеанса.
- Получите как можно больше идей от участников, поощряйте «дикие» или нелепые идеи.
- Помогайте участникам генерировать идеи, изменять и комбинировать их.
- Не допускайте критики, обсуждений и споров.
- Установите ограничения во времени, поддерживайте высокий темп для уменьшения задержек и исключения оценки идей.

Правила для секретарей:

- Кратко записывайте суть идей, если нужно, попросите короткое пояснение, для формулировок используйте слова автора идеи.
- Записывайте идеи таким образом, чтобы все участники могли их видеть.

Правила для участников:

- Полностью погрузитесь в процесс генерирования идей и мыслите свободно.
- Проявляйте творческий подход, будьте открыты для новых оригинальных идей, не бойтесь рисковать, подключайте воображение, дополняйте и расширяйте идей других участников.
- Не оценивайте идеи других, не критикуйте и не осуждайте, не прерывайте других докладчиков.

- Думайте быстро, высказывайте каждую идею кратко, размышлять будете после.

Проведение сеанса

Рекомендуется следующий сценарий проведения «мозгового штурма».

- Перед началом сеанса укажите его тему, обсудите правила и представьте участников.
- Для создания атмосферы уверенности и готовности генерировать идеи проводится 5-10 минутная разминка, тема которой должна быть нейтральной и понятной всем участникам, но не связанной с основной темой сеанса.
- «Мозговой штурм». Предложите высказывать любые (рациональные, радикальные, необычные) идеи, которые приходят на ум. Активно работайте 10 – 15 минут и сделайте перерыв. Останавливайтесь до того, как утихнет энтузиазм. После перерыва продолжайте работать. Если поток идей иссякает, то рассмотрите и уточните записанные идеи, убедитесь, что идеи понятны всем участникам. Объедините подобные идеи, удалите дубликаты. Определите критерии оценки идей.
- Достижение единодушного мнения. Попросите каждого члена группы проголосовать за первые 10-15 идей доработанного списка и анонимно представить его решение. Отведите на это достаточное время.
- Завершение сеанса. Поблагодарите участников сеанса, попросите заполнить форму оценки и сообщите, когда оцененные идеи будут им возвращены.

Обработка результатов сеанса

Редактированием идей после завершения сеанса обычно занимается лидер. Желательно, чтобы ему помогали (не более двух) разработчики приложения.

Из всех идей необходимо отбросить те, которые не завершены или не могут быть использованы. Остальные идеи могут быть разбиты на три категории:

- *Реальные* – представляют собой требования, которые могут быть оттестированы.
- *Возможные* – это те идеи, которые могут быть использованы в качестве требований.
- *Маловероятные* – идеи, скорее всего не являющиеся разумными требованиями.

Результатом редактирования должен быть *документ*, который в дальнейшем используется для спецификации требований к системе.

Методические указания к лабораторной работе №1

Тема: «Обследование объекта и обоснование необходимости создания ПО (ГОСТ 34.601-90»

Цель работы:

Изучение и практическое применение методов анализа осуществимости разработки ПО.

Ход лабораторного занятия

В начале занятия выбираются руководитель группы и секретарь.

В течение занятия, используя методологию «мозгового штурма», необходимо выполнить следующие этапы:

1. Провести исследование предприятия:
   a. Подразделения организации и предполагаемые пользователи системы.
   b. Основные бизнес-процессы организации (все).
   c. Бизнес-процессы, подлежащие автоматизации.
   d. Информационные потоки на предприятии, перечень документов.
   e. Анализ существующего уровня автоматизации в организации (список программного обеспечения, используемого в компании; данные об использовании этих пакетов в каждом из подразделений организации).
2. Провести предварительный анализ предприятия.

   В ходе анализа ответить на вопросы:

   - *Что произойдет с организацией, если система не будет введена в эксплуатацию?*
   - *Какие текущие проблемы существуют в организации, и как новая система поможет их решить?*

- *Каким образом система будет способствовать целям бизнеса?*
- *Требует ли разработка системы технологии, которая до этого не использовалась в организации?*

3. Выделить 3-4 программных аналога и проанализировать их с точки зрения похожих и уникальных функций.

4. Определить ***общие требования к системе*** в текстовом виде, которые согласовываются с Заказчиком.

Рекомендации по выполнению работы

Определение *общих требований к системе* имеет целью

- определить задачи, решаемые системой,
- круг заинтересованных лиц в разработке системы,
- границы использования системы
- основные свойства.

При разработке общих требований выполняются:

1. Определение задач, решаемых системой;

2. Выявление заинтересованных лиц в работе системы (например, пользователей системы, администраторов системы, лиц, пользующихся результатами, работы системы и т.д.);

3. Определение области применения системы;

4. Определение различных ограничений, налагаемых на систему (технические, экономические, системные и т.д.);

5. Определение цели создания системы;

6. Определение особенностей системы.

Результаты обсуждения должны быть записаны секретарем группы в таблице:

Вопрос	Участник 1	…	Участник n

Требования к оформлению отчетов

Каждый студент составляет индивидуальный отчет по лабораторной работе. Оценка деятельности группы производится по отчету руководителя.

В отчетах следует указать следующее.

Руководитель

Руководитель составляет отчет на основании отчетов всех участников группы. Отчет руководителя может содержать предложение участника группы без изменений (с указанием авторства), с изменениями (с указанием авторства и перечнем изменений), либо собственный вариант решения. При необходимости руководитель может либо разделять задачи между участниками группы, либо предлагать весь список задач каждому участнику. Второй вариант предпочтительнее, так как позволит формировать итоговый отчет, руководствуясь большим количеством вариантов решения.

При отсутствии в итоговом отчете руководителя предложений какого-либо из участников без дополнительных уточнений, этот участник получает максимально 1 балл за выполнение лабораторной работы.

Структура отчета руководителя

1. **Цель работы**
2. **Постановка задачи** (в краткой форме)
3. **Видение проекта**. Краткое описание целей проекта и проектных ограничений (бюджетных, временных и т.д.), которые важны для управления проектом
4. **Отчет об обследовании предприятия**:

- *Организационная структура объекта (итоговое словесное описание и схема)*
- *Кадровая структура объекта (итоговое словесное описание в формате: сотрудник::основные функции::зависимость (взаимодействие с другими сотрудниками и схема)*

- *Основные бизнес-процессы объекта (итоговое словесное описание в формате: процесс::участники процесса::входные данные и диаграмма IDF0 с необходимым уровнем декомпозиции)*
- *Основные информационные потоки предприятия (DFD – диаграмма + документы::выходные данные + документы*
- *Основные проблемы в организации (в формате: бизнес-процесс::участники::проблема)*

5. **Анализ рынка**

- *Перечень похожих программных продуктов (2 – 3)*
- *Сравнительный анализ выбранного ПО: общие функции, уникальные функции, достоинства и недостатки.*

Анализ можно выполнить в виде сравнительной таблицы, вида:

	Аналог 1	*Аналог 2*	..	
Функция 1	*есть*	*нет*	*частично*	
Функция 2	..			
…				

6. **Заключение о возможности реализации проекта**

- задачи, решаемые системой (диаграмма use case)
- круг заинтересованных лиц в разработке системы (предварительный перечень экторов системы с указанием уровня доступа)
- границы использования системы (ограничения на разработку)
- *основные* свойства системы в формате**:** *бизнес-процесс::участники::выгода для предприятия*.

<u>Остальные участники группы</u>

Остальные участники группы формируют отчеты, ориентируясь на указания руководителя. В качестве ориентира предлагается следующая структура.

Структура отчетов сотрудников

1. Цель разработки
2. Отчет об обследовании предприятия:

На основании стенограммы мозгового штурма сформулировать предложение по автоматизации в виде:

- Организационная структура объекта (словесное и схема)
- Кадровая структура объекта (словесное описание)

в формате

Сотрудник	Деятельность	Бизнес-процесс	В подчинении у:

- Основные функции объекта (use case)
- Основные бизнес-процессы объекта (таблица + IDF0)

Указать взаимодействие процессов, сотрудников и сотрудников и процессов в формате

Бизнес-процесс	Участники	Входные данные	Выходные данные	Документы

3. Анализ похожего программного обеспечения
4. Предложения по автоматизации бизнес-процессов предприятия (перечень)
5. Ограничения по автоматизации
6. Описание преимуществ для организации в результате реализации проекта

Секретарь

Отчет студента, выполняющего в данной лабораторной роль секретаря, должен быть сформирован в первую очередь, по крайней мере, его первая часть, так как он должен содержать максимально полную информацию о проведенном «мозговом штурме». Стенограмма «мозгового штурма» должна

быть представлена для ознакомления и использования всем участникам группы.

Структура отчета секретаря

1. Постановка задачи (в краткой форме)
2. Стенограмма мозгового штурма
3. Краткое описание проектных ограничений (бюджетных, временных и т.д.), которые важны для управления проектом (на основании мозгового штурма)
4. Основные функции объекта (use case)
5. Основные бизнес-процессы объекта (таблица + IDF0)
6. Предложения по автоматизации бизнес-процессов предприятия (перечень)

По итогам проведенных мероприятий и выполненных работ вся группа формирует доклад о результатах выполнения задания, с которым участвует в устном отчете перед преподавателем и остальными студентами. Для отчета рекомендуется сформировать тезисы либо презентацию по итоговому отчету руководителя. Основной докладчик – руководитель. На дополнительные вопросы отвечает вся группа, либо автор предложения.

Рекомендации по оформлению выступления

В презентации не должно быть ничего лишнего. Каждый слайд должен представлять собой необходимое звено повествования и работать на общую идею презентации. При создании мультимедийных презентаций необходимо учитывать особенности восприятия информации с экрана. Важно проверять презентацию на удобство чтения с экран. Тексты презентации не должны быть большими. Рекомендуется использовать сжатый, информационный стиль изложения материала.

Наиболее важные элементы мультимедийной презентации должны иметь подсказки или пояснения. Справочный материал презентации содержит основные определения, наиболее важные даты, таблицы для сравнения определенных характеристик объектов и т. п. Про каждую решённую задачу необходимо говорить конспективно: постановка, зачем нужна, что получилось. В самых интересных случаях, которыми группа гордится особо, можно сказать примерно следующее: «Идея метода (решения, алгоритма) заключается в том, чтобы…». Хорошим тоном в этой ситуации является экономия времени зрителей, а по сути, конкурентов. Поэтому слайды и структуру выступления необходимо тщательно продумать так, чтобы провести их шаг за шагом, от простого к сложному, по всем деталям презентуемого проекта. Здесь неуместны рассуждения о пользе полезного, так как в общих вопросах слушатели, скорее всего, осведомлены не хуже выступающих.

Подача материала на слайдах должна быть профессиональной с точки зрения дизайна. Диаграммы, схемы, графики и прочие виды деловой графики приветствуются. Слайды должны демонстрировать, что группа накопила достаточный опыт. Излишнее наукообразие, наоборот, не приветствуется. Перегруженность текстом и мелкий шрифт тяжелы для восприятия. Оптимальное число строк на слайде — от 6 до 11. Оптимальное количество цветов в презентации — от пяти до семи. На одну презентацию не более двух шрифтов. Любые данные представлять в виде инфографики, любые сравнения – в виде диаграмм или таблиц. Визуализируйте по максимуму!

Речь должна быть максимально доходчивой; впечатление, что Вы хотите запутать слушателя, равносильно провалу. Имеет смысл быть аккуратными, придерживаться единого стиля. Неряшливо сделанные слайды (разнобой в шрифтах и отступах, опечатки, типографические ошибки в формулах) вызывают подозрение, что и к содержательным вопросам докладчик подошёл спустя рукава.

Титульная страница необходима. Так же обязателен слайд с постановкой задачи и целью выполненной работы. Распространённая ошибка – читать слайд дословно. Лучше всего, если на слайде будет написана подробная информация (определения, теоремы, формулы), а словами будет рассказываться их содержательный смысл. Информация на слайде может быть более формальной и строго изложенной, чем в речи.

Пункты перечней должны быть короткими фразами; максимум – две строки на фразу, оптимально – одна строка. Чтение длинной фразы отвлекает внимание от речи. Короткая фраза легче запоминается визуально. Не проговаривайте формулы словами – это долго и безумно скучно. Оптимальная скорость переключения – один слайд за 1–2 минуты. Используйте изображения вместо и вместе с текстом везде, где это возможно. Изображения работают лучше слов, потому что мозг видит слова как несколько маленьких картинок вместо одной.

Успешная презентация – это доставленное по адресу и правильно воспринятое аудиторией сообщение. Не отвлекайтесь, но и не упускайте важного. Представьте, что вы пилот пассажирского самолета и вам нужно доставить людей из аэропорта А в аэропорт Б. Никому в салоне неинтересно почему вы стали пилотом или сколько кошек у вас дома. Пассажиром важно безопасно переместиться из точки А в точку Б. Вот и расскажите им именно об этом. Когда у презентатора хорошее настроение, его энергетика обязательно передастся аудитории. К сожалению, тоже самое произойдет, если выступающий говорит монотонно, при этом, не выражая не единой эмоции по поводу того, о чем он вещает. Получайте удовольствие от того, что вы делаете!

«Настоящие презентации фокусируют внимание аудитории на выступающем, на его идеях», – Стив Джобс.

Методические указания к лабораторной работе №2

Тема: «Сбор и анализ требований к ИС»

Цель работы:

Изучение и практическое применение методов сбора требований на разработку ИС.

Ход лабораторного занятия

В начале занятия выбираются руководитель группы и секретарь.

В течение занятия, используя методологию «мозгового штурма», необходимо выполнить следующие этапы:

1. **Выявить требования к ИС**.
2. **Сгруппировать требования по группам, согласно ГОСТ.**

ГОСТ разделяет все требования к системе на три класса:

- требования к системе в целом;
- требования к функциям (задачам), выполняемым системой;
- требования к видам обеспечения.

3. **Определить приоритет требований** (необходимые, желательные, дополнительные).
4. **Выделить экторов** – пользователей системы.
5. **Выделить основные варианты использования системы (не менее 5).**
6. **Определить архитектуру ПО.**

Вторая часть лабораторной работы выполняется индивидуально. Необходимо:

- расписать сценарии по каждому выделенному варианту использования системы (с учетом выделенных требований),
- представить их графически с помощью диаграмм прецедентов и диаграмм последовательности.

Руководитель имеет право распределить выделенные варианты использования ИС для выполнения описания между участниками группы поровну, либо продублировать все варианты для каждого участника. *Рекомендация для руководителя*: при распределении требований целесообразно дублировать требования с высоким приоритетом между участниками группы для достижения наилучшего результата.

Рекомендации по выполнению работы

При сборе требований в качестве основного применить метод опорных точек. А именно, попытаться выявить максимальное количество требований к системе (*не менее тридцати*), отождествляя себя с разными сотрудниками предприятия – будущими пользователями системы.

a. Требования должны касаться как функционала будущей системы, так и программного обеспечения, безопасности, пользователей, дизайна, интерфейса и др.
b. Требования должны быть сформулированы по каждому этапу автоматизируемых бизнес-процессов.
c. Должны быть выявлены требования к документации: количество документов, входная и выходная информация, вид, дизайн и т.д.

Если виды требований могут быть различными (зависит от целей проекта), то со свойствами все проще, их 3:

- Требование должно быть понятным;
- Требование должно быть конкретным;
- Требование должно быть тестируемым.

ГОСТ разделяет все требования к системе на три класса:

- требования к системе в целом;
- требования к функциям (задачам), выполняемым системой;
- требования к видам обеспечения.

Среди *требований к системе в целом* (системные требования) указываются требования:

- к структуре системы (здесь закладываются высокоуровневые архитектурные решения, либо структурные ограничения, вводится деление на подсистемы, комплексы и модули, решаются вопросы коммуникации компонент системы и системы с внешним миром),
- к режимам функционирования системы;
- персоналу (указывается численность, требуемая квалификация и режим работы);
- надежности;
- безопасности;
- эргономике и технической эстетике;
- транспортабельности для подвижных АС;
- эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;
- защите информации от несанкционированного доступа;
- сохранности информации при авариях;
- защите от влияния внешних воздействий;
- патентной чистоте;
- стандартизации и унификации,

а также показатели назначения (параметры, характеризующие степень соответствия системы ее назначению) и дополнительные требования (распространяются на обучающие подсистемы, средства контроля работоспособности системы и др.).

Требования ГОСТ к функциям (задачам), в переводе на современный язык, подразделяются на:

- перечень функциональных требований в привязке к подсистемам и очередям автоматизации;
- временной регламент реализации функциональных требований;

- требования к качеству реализации каждого из функциональных требований (в том числе – форме представления выходной информации, характеристики необходимой точности и времени выполнения, требования одновременности выполнения группы функций, достоверности выдачи результатов);
- перечень и критерии отказов для каждого функционального требования, по которому были заданы требования по надежности.

Требования к видам обеспечения. Среди видов обеспечения ГОСТ указывает математическое, информационное, лингвистическое, программное, техническое, метрологическое, организационное, методическое.

Требования к оформлению отчетов

Структура отчета руководителя:

1) формализованная постановка задачи
2) требования к ИС, выделенные для реализации ИС
3) общая диаграмма вариантов использования системы (use case)
4) итоговый вариант описания вариантов использования (с указанием авторства):
    a) текстовый сценарий
    b) диаграмма прецедентов
    c) диаграмма последовательности
5) итоговое описание документов (с указанием авторства):
    a) наименование
    b) входная, выходная информация
    c) вид.
6) Описание выбранной архитектуры ПО.

Структура отчетов сотрудников:

1) постановка задачи

2) требования к ИС, сгруппированные по ГОСТ с указанием приоритета

3) описания вариантов использования:

   a) текстовый сценарий

   b) диаграмма прецедентов

   c) диаграмма последовательности

4) описания документов, формируемых системой:

   a) наименование

   b) входная, выходная информация

   c) вид.

Структура отчета секретаря:

1) постановка задачи

2) стенограмма «мозгового штурма»

3) требования к ИС, сгруппированные по ГОСТ с указанием приоритета

4) DFD-диаграмма потоков данных ПО с указанием документов

5) компонентная диаграмма ПО.

По итогам проведенных мероприятий и выполненных работ вся группа формирует доклад о результатах выполнения задания, с которым участвует в устном отчете. Для отчета рекомендуется сформировать тезисы либо презентацию по итоговому отчету руководителя. Основной докладчик – руководитель. На дополнительные вопросы отвечает вся группа, либо автор предложения.

Методические указания к лабораторной работе №3

Тема: «Трассировка требований к ПС. Научно-исследовательские работы (ГОСТ – 19.102-77)»

Цель работы:

Изучение процесса трассирования требований и разработка матрицы трассировки требований. Изучить и применить практически навыки реализации НИС, согласно ГОСТ 19.102-77

Ход лабораторного занятия

В начале занятия выбираются руководитель группы и секретарь.

В течение занятия, используя методологию «мозгового штурма», необходимо выполнить следующие этапы:

1. **Определить программные и аппаратные средства для проектирования и разработки ПО.** Обосновать свой выбор.
2. **Определить характеристики и атрибуты качества ПО.**
3. **Сформировать перечень системных требований к ПО.**
4. **Уточнить список вариантов использования ПО и перечень прецедентов для каждого варианта.**
5. **Сформировать список предложений по тестированию ПО.**
6. **Составить матрицу трассировки требований.** Для этого использовать шаблон

ID Матрицы	Бизнес-требования (пользовательские)	Функциональные требования	Вариант использования	Сценарий тестирования	Комментарии
1					

ID Матрицы — уникальная последовательность для идентификации комбинации требований и связанных с ними вариантов использования.

Бизнес-требования – номер бизнес-требования (атрибута качества ИС), который идентифицирует критерии успеха, на основе которых будут выполняться тесты.

В качестве бизнес-требования может выступать атрибут качества ИС, определенный ранее, либо пользовательское требование, сформулированное на естественном языке заказчиком.

Функциональные требования – идентификационный номер функционального требования (в соответствии с ТЗ), которое исполняет указанное бизнес-требование.

Вариант использования – описание варианта использования, который будет использоваться для проверки соответствия бизнес-требований с функциональными требованиями.

Сценарий тестирования – предлагаемый вариант сценария тестирования реализации функционального требования на соответствие бизнес-требованию

Руководитель группы должен распределить выбранные требования между участниками группы. Подробное составление матрицы выполняется каждым участником группы самостоятельно.

Рекомендация для руководителя: при распределении требований целесообразно дублировать требования с высоким приоритетом между участниками группы для достижения наилучшего результата.

Вторая часть лабораторной работы выполняется индивидуально. Необходимо:

- Подробно заполнить матрицы трассировки требований для пользовательских и системных требований.
- Сформировать список вариантов использования
- Смоделировать каждый вариант использования

Рекомендации по выполнению работы

Как проверить, что требования определены достаточно полно и описывают все, что ожидается от будущей программной системы? Это можно сделать, проследив, все ли необходимые аспекты *качества ПО* отражены в них. Именно понятие *качественного ПО* соответствует представлению о том, что *программа* достаточно успешно справляется со всеми возложенными на нее задачами и не приносит проблем ни конечным пользователям, ни их начальству, ни службе поддержки, ни специалистам по продажам.

Стандарт *ISO* 9126 предлагает использовать для описания внутреннего и внешнего *качества ПО* многоуровневую модель. На верхнем уровне выделено 6 основных характеристик *качества ПО*. Каждая характеристика описывается при помощи нескольких входящих в нее атрибутов. Для каждого атрибута определяется набор метрик, позволяющих его оценить. Множество характеристик и атрибутов качества согласно *ISO* 9126 показано на рисунке 1.

Характеристики и атрибуты качества ПО позволяют систематически описывать требования к нему, определяя, какие свойства *ПО* по данной характеристике хотят видеть заинтересованные стороны. Таким образом, требования должны определять следующее:

- что *ПО* должно делать, например: позволять клиенту оформить заказы и обеспечить их доставку; обеспечивать контроль качества строительства и отслеживать проблемные места; поддерживать нужные характеристики автоматизированного процесса производства, предотвращая аварии и оптимальным образом используя имеющиеся ресурсы;
- насколько *ПО* должно быть надежно, например: работать 7 дней в неделю и 24 часа в сутки; допускается неработоспособность в течение не более 3 часов в год; никакие введенные пользователями

данные при отказе не должны теряться;

Рисунок 1. Многоуровневая модель качества ПО

- насколько должно быть удобно пользоваться *ПО*, например: покупатель должен, зная название товара и имея средние навыки работы в Интернет, находить нужный ему товар не более, чем за 2 минуты; инженер по специальности «строительство мостов» должен в течение одного дня уметь разобраться в 80% функций системы;
- насколько *ПО* должно быть эффективно, например: поддерживать обслуживание до 10000 запросов в секунду; время отклика на запрос при максимальной загрузке не должно превышать 3 с; время реакции на изменение параметров процесса производства не должно превышать 0.1 с; на обработку одного запроса не должно тратиться более 1 MB оперативной памяти;
- насколько удобно должно быть сопровождение, например: добавление в систему нового вида запросов не должно требовать

более 3 человеко-дней; добавление поддержки нового этапа процесса производства не должно стоить более $20000;

- насколько оно должно быть переносимо, например: *ПО* должно работать на операционных системах Linux, Windows XP и MacOS X; *ПО* должно работать с документами в форматах MS Word 97 и HTML; *ПО* должно сохранять файлы отчетов в форматах MS Word 2000, MS Excel 2000, HTML, RTF и в виде обычного текста; *ПО* должно сопрягаться с существующей системой записи данных о заказах.

Приведенные атрибуты качества закреплены в стандартах, но это не значит, что они вполне исчерпывают понятие качества *ПО*. Так, например, в стандарте ИСО 9126 отсутствуют характеристики, связанные с мобильностью *ПО* (mobility), т.е. способностью программы работать на любой аппаратуре в различных операционных системах. Так же, вместо надежности многие исследователи предпочитают рассматривать более общее понятие добротности (dependability), которое описывает способность *ПО* поддерживать определенные показатели качества по основным характеристикам (функциональности, производительности, удобству использования) с заданными вероятностями выхода за их рамки и определенным максимальным ущербом от возможных нарушений.

Тестирование – наиболее широко применяемый *метод контроля качества*. Для оценки многих атрибутов качества не существует других эффективных способов, кроме *тестирования*.

Организация *тестирования ПО* регулируется следующими стандартами:

- IEEE 829-1998 Standard for Software Test Documentation – описывает виды документов, служащих для подготовки тестов.
- IEEE 1008-1987 (R1993, R2002) Standard for Software *Unit Testing* - описывает организацию модульного *тестирования*.

- ISO/IEC 12119:1994 (аналог AS/NZS 4366:1996 и ГОСТ Р-2000, также принят IEEE под номером IEEE 1465-1998) Information Technology. *Software packages* – Quality requirements and testing – описывает требования к процедурам *тестирования* программных систем.

Тестировать можно соблюдение любых требований, соответствие которым выявляется во время работы *ПО*. Из характеристик качества по ISO 9126 этим свойством не обладают только атрибуты *удобства сопровождения*. Поэтому выделяют виды *тестирования*, связанные с проверкой определенных характеристик и атрибутов качества – *тестирование функциональности*, *надежности*, *удобства использования*, *переносимости* и *производительности*, а также *тестирование* защищенности, функциональной пригодности и пр. Кроме того, особо выделяют *нагрузочное* или *стрессовое тестирование*, проверяющее работоспособность *ПО* и показатели его *производительности* в условиях повышенных нагрузок – при большом количестве пользователей, интенсивном обмене данными с другими системами, большом объеме передаваемых или используемых данных и пр.

Требования к оформлению отчетов

Структура отчета руководителя:

1) формализованная постановка задачи
2) ТЗ (итоговая версия)
3) диаграммы ВИ
4) итоговая матрица трассировки пользовательских требований (с указанием авторов методик испытаний)
5) итоговая матрица трассировки системных требований (с указанием авторов методик испытаний)
6) глоссарий проекта.

Структуры отчетов сотрудников:

1) постановка задачи
2) диаграммы use case для каждого варианта использования (ВИ)
3) матрица трассировки пользовательских требований
4) матрица трассировки системных требований
5) подробные методики испытаний.

Структура отчета секретаря:

1) постановка задачи
2) стенограмма «мозгового штурма»
3) матрица трассировки пользовательских требований
4) матрица трассировки системных требований
5) стратегия тестирования *ПО*.

По итогам проведенных мероприятий и выполненных работ вся группа формирует доклад о результатах выполнения задания, с которым участвует в устном отчете. Для отчета рекомендуется сформировать тезисы либо презентацию по итоговому отчету руководителя. Основной докладчик – руководитель. На дополнительные вопросы отвечает вся группа, либо автор предложения.

Методические указания к лабораторной работе №4

Тема: «Проектирование интерфейса и разработка дизайн - макета»

Цель работы:

Изучение и практическое применение методов сбора требований к пользовательскому интерфейсу ИС. Проектирование и выбор дизайнерских решений.

Ход лабораторного занятия

В начале занятия выбираются руководитель группы и секретарь.

В течение занятия, используя методологию «мозгового штурма», необходимо выполнить следующие этапы:

1. **Выявить требования к пользовательскому интерфейсу ИС**. При сборе требований в качестве основного применить метод опорных точек. А именно, попытаться выявить максимальное количество требований к пользовательскому интерфейсу (ПИ) (*не менее тридцати*), отождествляя себя с разными сотрудниками предприятия – будущими пользователями системы.

   a. Требования должны определять структуру ПИ:

   - основное и вложенные меню (текстовое и графическое описание),
   - количество страниц (для веб-систем) и их назначение,
   - структуры главной и вложенных страниц,
   - количество и структуру диалоговых окон,
   - количество и структуру системных сообщений (об ошибках, о завершении процесса и т.п.),
   - Наличие обратной связи и др.

   b. Требования должны быть сформулированы по каждому этапу автоматизируемых бизнес-процессов.

2. **Определить приоритет требований** (необходимые, желательные, дополнительные)

3. Выделить основные требования к дизайну.

Вторая часть лабораторной работы выполняется индивидуально. Необходимо:

- Построить схему структуры ПИ.
- Разработать дизайн-макеты главной страницы системы, дополнительных страниц (вкладок), диалоговых окон и сообщений.
- Обосновать выбор дизайна.
- Сформировать стратегию тестирования ПИ.

Рекомендации по выполнению работы

НАПОМИНАНИЕ:

- ТРЕБОВАНИЕ ДОЛЖНО БЫТЬ ПОНЯТНЫМ;
- ТРЕБОВАНИЕ ДОЛЖНО БЫТЬ КОНКРЕТНЫМ;
- ТРЕБОВАНИЕ ДОЛЖНО БЫТЬ ТЕСТИРУЕМЫМ.

Требования к пользовательскому интерфейсу могут быть разбиты на две группы:

2. Требования к внешнему виду пользовательского интерфейса и формам взаимодействия с пользователем;

3. Требования по доступу к внутренней функциональности системы при помощи пользовательского интерфейса.

Другими словами, первая группа требований описывает взаимодействие подсистемы интерфейса с пользователем, а вторая – с внутренней логикой системы.

К первой группе можно отнести следующие типы требований:

Требования к размещению элементов управления на экранных формах

Определяют общие принципы размещения элементов пользовательского интерфейса или требования к размещению конкретных элементов. Например:

Каждое окно приложения должно быть разбито на три части: строка меню, рабочая область и статусная строка. Строка меню должна быть горизонтальной и прижатой к верхней части окна, статусная строка должна быть горизонтальной и прижата к нижней части окна, рабочая область должна находиться между строкой меню и статусной строкой и занимать всю оставшуюся площадь окна.

При тестировании этого требования достаточно определить, что в каждом окне системы действительно присутствуют три части, которые расположены и прижаты согласно требованиям, причем сохраняют свое расположение даже при изменении размеров окна, его сворачивании/разворачивании, перемещении по экрану, при перекрытии его другими окнами.

Примером требований по размещению конкретного элемента может служить следующее:

Кнопка «Начать передачу» должна находиться непосредственно под строкой меню в левой части рабочей области окна.

При тестировании этого требования необходимо так же проверить, сохраняется ли расположение элемента при изменении размера окна, а также при использовании элемента (в данном случае – при нажатии).

Требования к содержанию и оформлению выводимых сообщений

Требования к содержанию и оформлению выводимых сообщений определяют текст сообщений, выводимых системой, его шрифтовое и цветовое оформление. Также часто в таких требованиях определяется, в каких случаях выводится то или иное сообщение. Лучше всего, если каждое требование будет сопровождаться визуализацией, например, эскизом.

Например, требование:

Сообщение «Невозможно открыть файл» должно выводиться в статусную строку прижатым к левому краю красным цветом полужирным шрифтом в случае недоступности открываемого файла по чтению.

При тестировании этого требования должно быть проверено, что при возникновении указанной ситуации сообщение действительно выводится, как указано.

Однако, в случае тестирования требования вида

Сообщения об ошибках должны выводиться в статусную строку, прижатыми к левому краю красным цветом полужирным шрифтом,

необходимо будет уже проверять форматы всех возможных сообщений об ошибках программы во всех возможных ошибочных ситуациях.

Таким образом, можно видеть, что при тестировании пользовательского интерфейса не всегда можно однозначно определить количество тестовых примеров, которые понадобятся для тестирования требования. Эта проблема вызвана тем, что требования к пользовательскому интерфейсу зачастую кажутся слишком очевидными для их точной формулировки. Учитывайте это: все требование должны быть задокументированы и персонифицированы. Приводимые требования должны описывать **именно разрабатываемую систему**, а не выражать общий взгляд на подобные системы. Такая неконкретность требований вызывает большое количество тестов для каждого требования.

Требования к форматам ввода

Данная группа требований определяет, в каком виде информация поступает от пользователя в систему. При этом кроме собственно требований, определяющих корректный формат, к этой группе относятся требования, определяющие реакцию системы на некорректный ввод. Для проверки таких требований необходимо планировать как позитивные, так и негативные тесты. Желательно при этом разбивать различные варианты ввода на классы эквивалентности (как минимум на два – корректные и некорректные).

Ко второй группе относятся следующие типы требований.

Требования к реакции системы на ввод пользователя

Данный тип требований определяет связь внутренней логики системы и интерфейсных элементов. Например,

При нажатии кнопки «Сброс» значение таймера синхронизации передачи должно сбрасываться в 0.

Для проверки такого требования в тестовом примере должно быть сымитировано нажатие на кнопку «Сброс», после чего должна проводиться проверка значения таймера. Однако некоторые требования определяют в качестве реакции системы не то, как меняется ее внутреннее состояние, а реакцию пользовательского интерфейса. Например, в требовании

При нажатии кнопки «Отложенный сброс» должно выводиться окно «Ввод значения времени для отложенного сброса».

В качестве реакции на использование одного интерфейсного элемента определяется появление другого интерфейсного элемента. Такие требования проверяются при помощи имитации ввода пользователя и анализа появляющихся интерфейсных элементов.

Требования к времени отклика на команды пользователя

Подсознательно пользователь воспринимает операции продолжительностью более 1 секунды как длительные. Если в этот момент система не сообщает пользователю о том, что она выполняет какую-либо операцию, пользователь начнет считать, что система зависла или работает в неверном режиме. Поэтому в качестве отдельного типа требований можно выделить требования ко времени отклика системы на различные пользовательские операции, в которых все предельные времена отклика должны быть точно определены, либо указано, что во время длительных операций должны выводиться информационные сообщения (например, индикатор прогресса). Значения предельного времени и равномерность увеличения значений индикатора прогресса должны проверяться соответствующими тестами.

Некоторые требования к пользовательскому интерфейсу могут оказаться не пригодными для тестирования, либо их тестирование будет

значительно затруднено. К таким требованиям, в первую очередь, относятся требования, описывающие субъективные характеристики интерфейса, которые не могут быть точно определены или измерены при выполнении тестовых примеров. Важно: при анализе требований к пользовательскому интерфейсу необходимо четко представлять, какой элемент интерфейса и каким образом будет проверяться, какая его характеристика и как будет измеряться в ходе тестирования.

Примером не пригодного для тестирования требования может служить классическое:

Пользовательский интерфейс должен быть интуитивно понятным.

Без определения четких критериев интуитивной понятности проверка такого требования невозможна. При этом необходимо понимать, что критерий в данном случае может быть двух видов: детерминированным или вероятностным. Примером детерминированного критерия может быть дополнение к требованию вида

Под интуитивной понятностью интерфейса понимается доступность любой функции системы при помощи не более чем 5 щелчков мыши по интерфейсным элементам.

Требование с таким описанием критерия поддается как ручному, так и автоматизированному тестированию, более того, результат такого тестирования не будет зависеть от субъективного мнения тестировщика (понятия об интуитивной понятности у всех разные).

Примером вероятностного критерия может служить следующее дополнение:

Под интуитивной понятностью интерфейса понимается, что пользователь обращается к руководству пользователя не чаще, чем раз в пять минут на этапе обучения и не чаще, чем раз в 2 часа на этапе активного использования системы. Значения должны быть получены на репрезентативной выборке пользователей не менее 1000 человек.

Требования к оформлению отчета

Структура отчета руководителя:

1) формализованная постановка задачи
2) требования к ПИ ИС, выделенные для реализации ИС
3) итоговая схема структуры ПИ
4) итоговые дизайн-макеты главной страницы системы, дополнительных страниц (вкладок), диалоговых окон и сообщений (с указанием авторства идей)
5) обоснование выбранного дизайна
6) стратегия по тестированию ПИ.

Структура отчетов сотрудников:

1) постановка задачи
2) требования к ПИ ИС, сгруппированные с указанием приоритета
3) дизайн-макеты главной страницы системы, дополнительных страниц (вкладок), диалоговых окон и сообщений
4) обоснование предложенного дизайна.

Структура отчета секретаря:

1) постановка задачи
2) стенограмма «мозгового штурма»
3) требования к ПИ, сгруппированные с указанием приоритета
4) схема структуры ПИ
5) предложения по тестированию ПИ.

По итогам проведенных мероприятий и выполненных работ вся группа формирует доклад о результатах выполнения задания, с которым участвует в устном отчете. Для отчета рекомендуется сформировать тезисы либо презентацию по итоговому отчету руководителя. Основной докладчик – руководитель. На дополнительные вопросы отвечает вся группа, либо автор предложения.

Методические указания к лабораторной работе №5

Тема: Анализ и проектирование входных и выходных данных ПО. Проектирование БД (ГОСТ 34.321 – 96).

Цель работы:

изучить и применить на практике различные методы сбора требований и проектирования и описания БД.

Ход лабораторного занятия

В начале занятия выбираются руководитель группы и секретарь.

В течение занятия, используя методологию «мозгового штурма», необходимо выполнить следующие этапы:

1. **Сформировать набросок диаграммы последовательности для системы.**
2. **Определить и уточнить список входных и выходных данных.**
3. **Уточнить список документов.**
4. **Выявить требования к БД ИС**.
    - Определить сущности
    - Определить атрибуты сущностей
    - Определить ограничения
    - Определить взаимосвязи между сущностями
    - Определить уровни доступа к данным

Вторая часть лабораторной работы выполняется индивидуально. Необходимо сформировать:

1) ER-диаграмма логической модели базы данных и необходимые комментарии к ней;
2) диаграмма физической модели базы данных;

3) полное описание таблиц базы данных ИС с указанием типов и назначения данных, а так же примерами значений в случае необходимости;

4) программный каркас ИС, формирующий смоделированную БД в виде диаграммы последовательности.

Руководитель имеет право распределить задачи между участниками группы. Рекомендуется дублирование задач для лучшего результата.

Рекомендации по выполнению работы

При проектировании базы данных решаются две основных проблемы:

- Каким образом отобразить объекты предметной области в абстрактные объекты модели данных, чтобы это отображение не противоречило семантике предметной области и было, по возможности, лучшим (эффективным, удобным и т.д.)? Часто эту проблему называют проблемой логического проектирования баз данных.
- Как обеспечить эффективность выполнения запросов к базе данных, т.е. каким образом, имея в виду особенности конкретной СУБД, расположить данные во внешней памяти, создание каких дополнительных структур (например, индексов) потребовать и т.д.? Эту проблему называют проблемой физического проектирования баз данных.

В случае реляционных баз данных трудно представить какие-либо общие рецепты по части физического проектирования. Здесь слишком много зависит от используемой СУБД.

Основные понятия модели Entity-Relationship (Сущность-Связи)

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна. Имя сущности – это имя типа,

для большей выразительности и лучшего понимания имя сущности может сопровождаться примерами конкретных объектов этого типа.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибутом сущности является любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности. Уникальным идентификатором сущности является атрибут, комбинация атрибутов, комбинация связей или комбинация связей и атрибутов, уникально отличающая любой экземпляр сущности от других экземпляров сущности того же типа.

Получение реляционной схемы из ER-схемы

Шаг 1. Каждая простая сущность превращается в таблицу. Простая сущность – сущность, не являющаяся подтипом и не имеющая подтипов. Имя сущности становится именем таблицы.

Шаг 2. Каждый атрибут становится возможным столбцом с тем же именем; может выбираться более точный формат. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы, соответствующие обязательным атрибутам, – не могут.

Шаг 3. Компоненты уникального идентификатора сущности превращаются в первичный ключ таблицы. Если имеется несколько возможных уникальных идентификатора, выбирается наиболее используемый. Если в состав уникального идентификатора входят связи, к числу столбцов первичного ключа добавляется копия уникального идентификатора сущности, находящейся на дальнем конце связи (этот процесс может продолжаться рекурсивно). Для именования этих столбцов используются имена концов связей и/или имена сущностей.

Шаг 4. Связи многие-к-одному (и один-к-одному) становятся внешними ключами. Т.е. делается копия уникального идентификатора с конца связи "один", и соответствующие столбцы составляют внешний ключ. Необязательные связи соответствуют столбцам, допускающим неопределенные значения; обязательные связи – столбцам, не допускающим неопределенные значения.

Шаг 5. Индексы создаются для первичного ключа (уникальный индекс), внешних ключей и тех атрибутов, на которых предполагается в основном базировать запросы.

Шаг 6. Если в концептуальной схеме присутствовали подтипы, то возможны два способа:

1. все подтипы в одной таблице (а)
2. для каждого подтипа - отдельная таблица (б)

При применении способа (а) таблица создается для наиболее внешнего супертипа, а для подтипов могут создаваться представления. В таблицу добавляется, по крайней мере, один столбец, содержащий код ТИПА; он становится частью первичного ключа.

При использовании метода (б) для каждого подтипа первого уровня (для более нижних – представления) супертип воссоздается с помощью представления UNION (из всех таблиц подтипов выбираются общие столбцы – столбцы супертипа).

Шаг 7. Имеется два способа работы при наличии исключающих связей:

1) общий домен (а)
2) явные внешние ключи (б)

Если остающиеся внешние ключи все в одном домене, т.е. имеют общий формат (способ (а)), то создаются два столбца: идентификатор связи и идентификатор сущности. Столбец идентификатора связи используется для различения связей, покрываемых дугой исключения. Столбец

идентификатора сущности используется для хранения значений уникального идентификатора сущности на дальнем конце соответствующей связи.

Если результирующие внешние ключи не относятся к одному домену, то для каждой связи, покрываемой дугой исключения, создаются явные столбцы внешних ключей; все эти столбцы могут содержать неопределенные значения.

Требование к оформлению отчета

Структура отчета руководителя:

1) формализованная постановка задачи
2) требования к БД, сгруппированные с указанием приоритета
3) итоговая ER-диаграмма логической модели базы данных и необходимые комментарии к ней
4) итоговая диаграмма физической модели базы данных
5) полное описание таблиц базы данных ИС с указанием типов и назначения данных, а так же примерами значений в случае необходимости;
6) итоговая диаграмма последовательности для ИС.

Структура отчетов сотрудников:

1) постановка задачи
2) построить логическую схему БД
3) построить физическую схему БД
4) построить диаграмму последовательности для всей системы с указанием взаимодействия между модулями посредством передачи сообщений (данных).
5) сформировать макеты документов.

Структура отчета секретаря:

1) постановка задачи;

2) стенограмма «мозгового штурма»;

3) требования к БД, сгруппированные с указанием приоритета;

4) диаграмма последовательности для всей системы с указанием взаимодействия между модулями посредством передачи сообщений (данных).

По итогам проведенных мероприятий и выполненных работ вся группа формирует доклад о результатах выполнения задания, с которым участвует в устном отчете. Для отчета рекомендуется сформировать тезисы либо презентацию по итоговому отчету руководителя. Основной докладчик – руководитель. На дополнительные вопросы отвечает вся группа, либо автор предложения.

Методические указания к лабораторной работе №6

Тема: «Разработка ПО. Моделирование физической реализации ПО»

Цель работы:

Изучение и практическое применение методов ООП для реализации ПО.

Ход лабораторного занятия

В начале занятия выбираются руководитель группы и секретарь.

В течение занятия, используя методологию «мозгового штурма», необходимо выполнить следующие этапы:

1. На основании ТЗ:
   a) **уточнить архитектуру ИС**
   b) **смоделировать компонентную диаграмму ИС**
   c) **выделить классы и их атрибуты, а так же списки методов**.
2. На основании ТЗ и выделенных вариантов использования **определить набор функций (и модулей) с высоким приоритетом**, подлежащих обязательной реализации (базовая комплектация)**.**
3. **Сформировать прототип разрабатываемого ПО в виде диаграммы состояний**
4. **Распределить функционал для реализации между членами группы.**

Вторая часть лабораторной работы выполняется индивидуально. Необходимо:

- Разработать диаграмму классов ИС
- Разработать алгоритмы реализации выделенных функций базовой комплектации
- Отобразить алгоритмы графически в виде диаграмм активности
- Уточнить методику испытаний для выделенных функций
- Разработать диаграмму состояний для ИС

Рекомендации по выполнению работы

Диаграммы активности

Главное: диаграмма активности должна иметь только одно начальное и только одно конечное состояние. На практике, для лучшего представления возможных вариантов выхода из прецедента, например, допустимо использовать несколько конечных состояний. Но важно помнить, что все они являются лишь вариациями, предлагаемыми для улучшения понимания алгоритма действий системы.

Располагать диаграмму активности принято вертикально, отображая начальное состояние системы вверху, а конечное, соответственно, внизу. Такая конструкция легче читается, но при необходимости, можно использовать и горизонтальную развертку (рис.2).

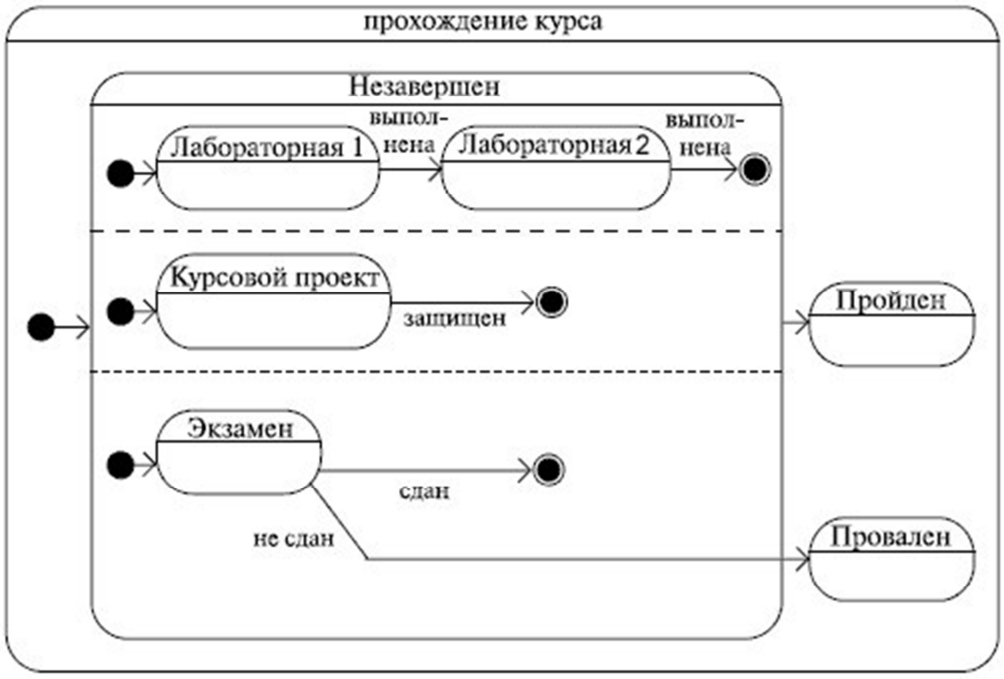

Рисунок 2 Пример диаграммы активности

Диаграмма может быть разделена на несколько вертикальных или горизонтальных областей, которые называют ***дорожками*** *(swimlanes),* которые используют не логической группировки деятельностей. В частности, на рисунке 2 указаны три дорожки, позволяющие акцентировать внимание на составной природе отдельных состояний. На рисунке 3 другая ситуация,

дорожки использованы для визуализации деятельностей отделов компании в рамках рассматриваемого варианта использования.

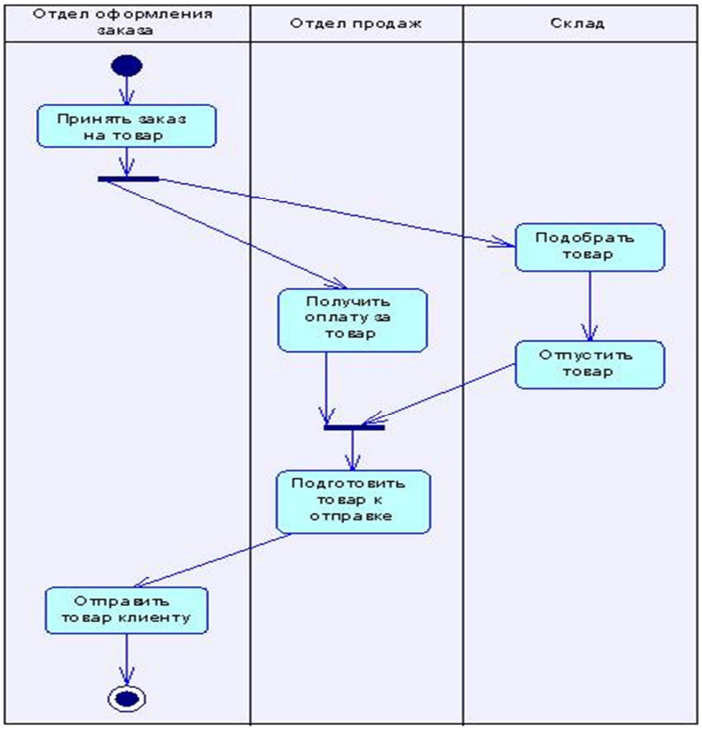

Рисунок 3 Пример отображения дорожек на диаграмме активности

В качестве основных элементов диаграммы деятельности можно указать:

начальное состояние – вход в диаграмму; графическая реализация – закрашенный круг (рис. 4);

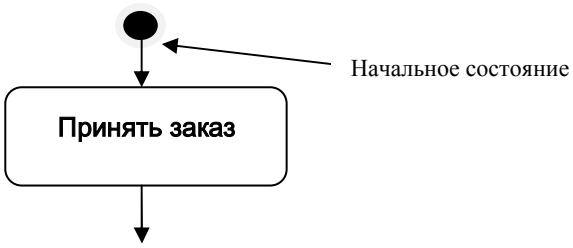

Рисунок 4 Пример визуализации начального состояния

состояние действия (*actionstate)* – атомарное действие (например, вызов операции, передача сигнала, присваивание значений атрибутам и т.п.).

графическая реализация – прямоугольник с закругленными краями, внутри которого записывается уникальное для данной диаграммы имя действия (рис. 5);

Рисунок 5 Пример отображения действия

Возможны два случая графической нотации для состояния: просто прямоугольник, и тогда там указывается только имя действия, и прямоугольник, разделенный на две части горизонтальной линией. Второй способ используется для визуализации действий, которые система может одновременно выполнять, но при этом каждое из них – независимо. Тогда в верхней секции указывается имя действия, а в нижней – список внутренних состояний или переходов в виде:

<метка / выражение>,

где <метка> – это встроенное имя, которое не может быть использовано в качестве идентификатора для события или деятельности (рис. 6). Список меток:

entry – входное действие, выполняется в момент входа в данное состояние;

exit – выходное действие, выполняется в момент выхода из данного состояния;

do – внутренняя деятельность (doactivity), определение для объекта операций или процедур, которые требуют некоторого времени для выполнения;

include – обращение к автомату с именем, указанным после данной метки.

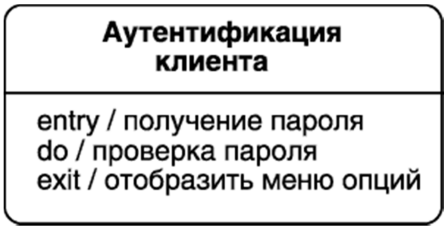

Рисунок 6 Отображение действия с использованием меток

состояние деятельности (*Activity, Process)* – неатомарный шаг, который может быть в дальнейшем подвергнут декомпозиции; графическая

реализация – прямоугольник с закругленными краями, внутри которого записывается уникальное для данной диаграммы имя деятельности (рис. 7);

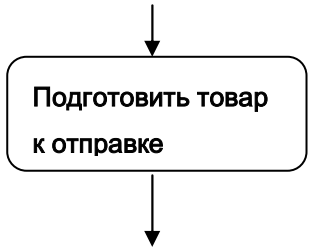

Рисунок 7 Пример отображения деятельности

переход (*Transitions)* – отображение потока данных между двумя состояниями объекта;графическая реализация – линия со стрелкой (рис. 8);

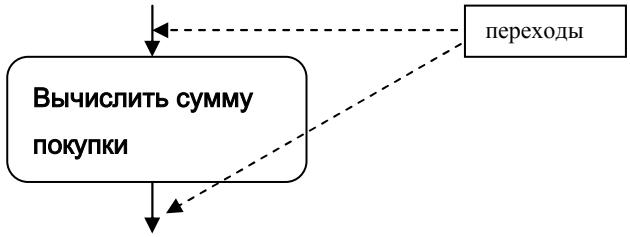

Рисунок 8 Пример отображения переходов

Каждый переход может быть снабжен строкой текста. Если из состояния действия выходит несколько переходов, то необходимо для каждого указывать сторожевые условия, истинным из которых на один момент времени может быть только одно.

ветвления – описания различных путей выполнения алгоритма, в зависимости от некоторого сторожевого условия, при этом в точку ветвления может входить только один переход, а выходить – два или более. Графическая реализация – символ решения (decision) в виде незакрашенного ромба (рис. 9). Входящим ветвление может быть только один переход, а выходящим – несколько. Каждый из выходящих переходов должен быть снабжен сторожевыми условиями, только одно из которых может быть истинно на один момент времени.

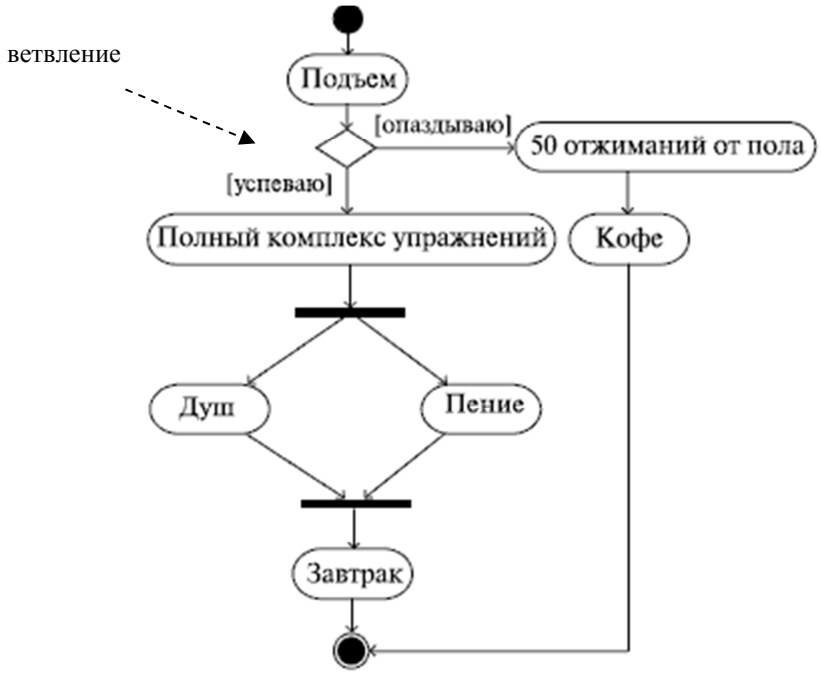

Рисунок 9 Пример ветвления на диаграмме активности

сторожевое условие (*guardcondition*) – булевское выражение, от которого зависит выполнение одного из переходов (потоков); указывается в прямых скобках рядом с символов ветвления (рис. 9);

разделения (*concurrentfork*) – отображение входа в параллельные потоки, при этом разделение может иметь только один переход на входе, а на выходе – два или более; графическая реализация – жирная вертикальная линия (рис. 10);

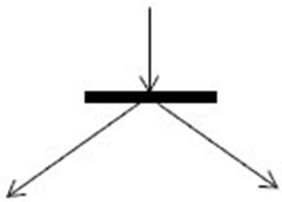

Рисунок 10 Пример визуализации разделения

слияния (*concurrentjoin)* – отображение выхода из параллельных потоков, при этом слияние может иметь только несколько переходов на входе, но только один на выходе; графическая реализация – жирная вертикальная линия (рис. 11);

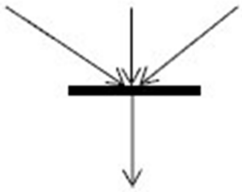

Рисунок 11 Пример визуализации слияния

Для иллюстрации рассмотренных видов переходов рассмотрим пример диаграммы активности, отображающей процесс регистрации пассажиров в аэропорту (рис. 12).

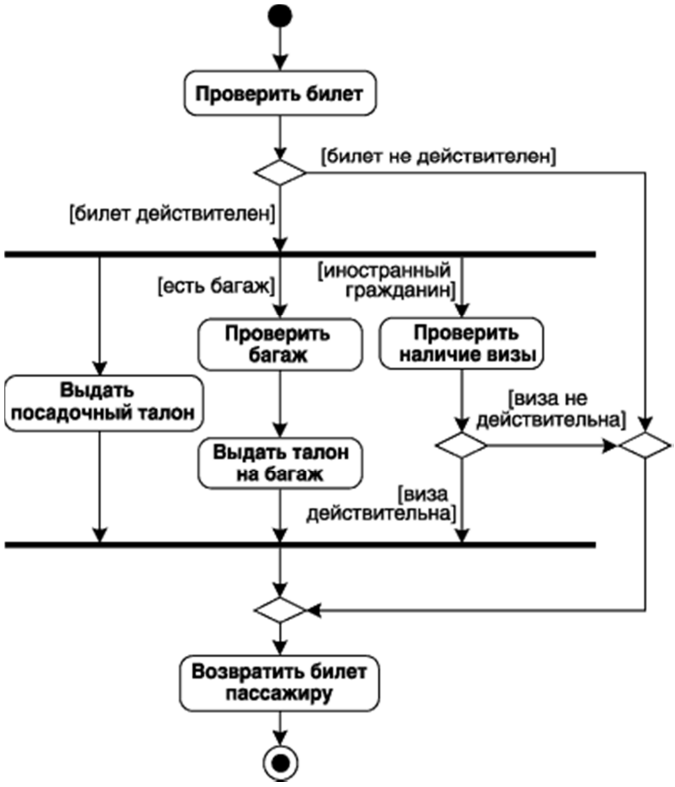

Рисунок 12 Пример различных видов переходов на диаграмме активности

Данная диаграмма примечательна еще и использованием символа для отображения объединения альтернативных ветвей, которые выходят из символов ветвления потока, что упрощает понимание последовательности действий и соответствует негласным правилам хорошего тона при создании диаграмм;

объекты – на диаграмме можно отображать объекты, являющиеся входными данными или результатом выполнения какого-либо действия. Часто объект, выступающий в роли результата выполнения одного действия, является одновременно и входным параметром для другого. В том случае,

когда объект используется на диаграмме несколько раз, необходимо указывать его характеристики состояния, исключающие несогласованность и запутывание диаграммы. Графическая реализация – прямоугольник, внутри которого указывается имя объекта; связь потока объектов с потоком деятельностей происходит через переходы с помощью пунктирной линии со стрелкой (рис. 13). При этом, если связь между объектом и переходом достаточно четко указывает последовательность действий, дублировать ее переходом (сплошной линией со стрелкой) не обязательно;

состояние класса – довольно часто один и тот же объект может использоваться в некоторой последовательности деятельностей, каждая из которых каким-то образом меняет состояние этого объекта. В таких случаях рядом с именем объекта допускается указание состояния объекта (в квадратных скобках), что позволит избежать путаницы (рис. 13);

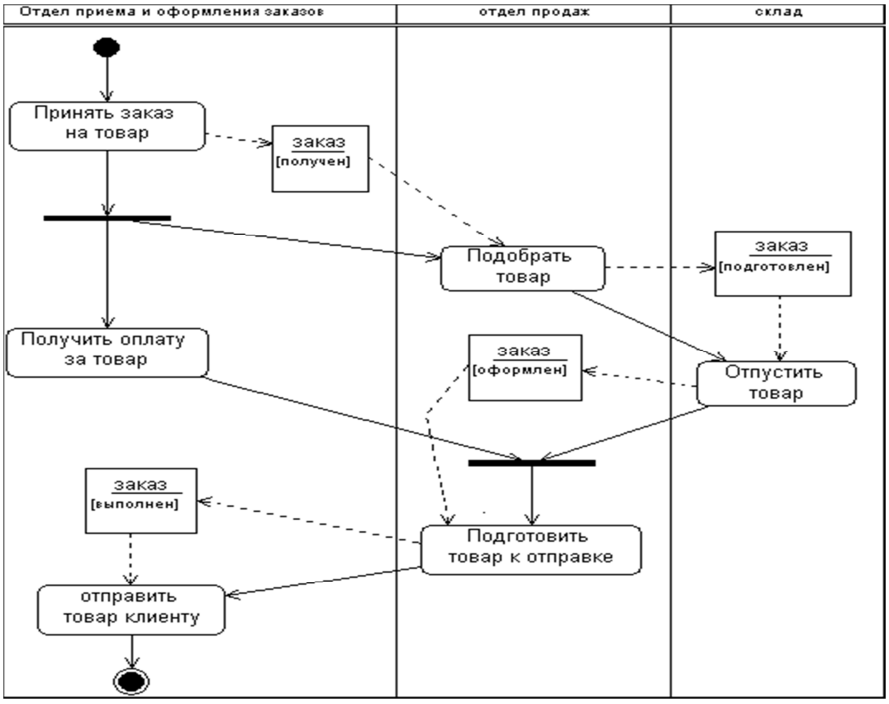

Рисунок 13 Пример диаграммы деятельности с указанием объектов

конечное состояние – выход из диаграммы; графическая реализация – закрашенный круг внутри незакрашенной окружности (рис.14);

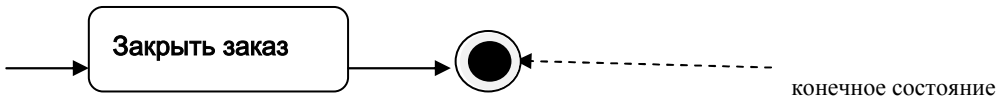

Рисунок 14 Пример отображения конечного состояния

конечное состояние потока – завершение конкретного потока, не влияющее на другие потоки на диаграмме (рис. 15); графически отображается в виде незакрашенного перечеркнутого круга.

Рисунок 15 Конечное состояние потока

Важно учитывать, что англоязычные case-средства оперируют понятием *узел* для обозначения основных компонент диаграмм деятельности. Например, узел управления (control node) – некий абстрактный узел действия, координирующий потоки. Соответственно, о каждом виде деятельности на диаграмме говорят как об узле: начальный узел деятельности (activity initial node, рис. 4), конечный узел деятельности (activity final node, рис. 15), конечный узел потока (flow final node, рис. 14), узел расширения (ветвление, разделение, слияние) и т.д.

Требования к оформлению отчета

Структура отчета руководителя:

1) постановка общей задачи
2) архитектура ИС (описание с обоснованием)
3) компонентная диаграмма ИС
4) общая диаграмма активности ПО на базе диаграмм для выделенных функций
5) диаграмма классов ПО итоговая с описанием
6) диаграмма состояний ПО итоговая с описанием
7) итоговый тестовый журнал

Структура отчетов сотрудников:

1) постановка индивидуальных подзадач
2) определение входных и выходных данных для каждой функции (модуля)
3) реализация индивидуальных алгоритмов:
    a. в виде диаграмм активности
    b. в виде диаграммы состояний
4) описание функционального (модульного) тестирования
5) разработка диаграммы классов ИС с описанием.

Структура отчета секретаря:

1) постановка общих задач
2) стенограмма «мозгового штурма»
3) таблица распределения подзадач между участниками группы
4) определение входных и выходных данных для каждой функции (модуля)
5) реализация индивидуальных алгоритмов:
    a) в виде диаграмм активности
    b) в виде диаграммы состояний
6) описание функционального (модульного) тестирования.

По итогам проведенных мероприятий и выполненных работ вся группа формирует доклад о результатах выполнения задания, с которым участвует в устном отчете. Для отчета рекомендуется сформировать тезисы либо презентацию по итоговому отчету руководителя. Основной докладчик – руководитель. На дополнительные вопросы отвечает вся группа, либо автор предложения.

Методические указания к лабораторной работе №7

Тема: «Разработка ПО. Подведение итогов»

Цель работы:

Изучение и практическое применение методов представления алгоритмов реализуемого ПО; оформления сопутствующей документации.

Ход лабораторного занятия

В начале занятия выбираются руководитель группы и секретарь.

В течение занятия, используя методологию «мозгового штурма», необходимо выполнить следующие этапы:

1. **Сформировать единое представление о реализованном ПО.**
2. **Разработать общую диаграмму функциональности для реализованного ПО.**
3. **Определить несоответствия функционала реализованного ПО с ТЗ.**
4. **Сформировать план руководств** (пользователя, системного администратора) **к реализованной АС.**
5. **Уточнить глоссарий проекта.**

Руководитель должен распределить нагрузку по составлению руководств между участниками группы и выполнению дополнительных работ по реализации АС.

Вторая часть лабораторной работы выполняется индивидуально. Необходимо:

- Выполнить повторную трассировку требований
- Доработать (при необходимости) АС в соответствии с ТЗ
- Разработать общую диаграмму последовательности для ПО
- Разработать общую диаграмму активности для ПО
- Сформировать руководство:

- Пользователя (по ролям)
- Администратора (инсталляция/деинсталляция)

Рекомендации по выполнению работы

Одним из инструментов установления зависимости между сформулированными требованиями и их изменениями является трассировка, т.е. поддержка развития и обработки требований с прослеживанием идентифицированных связей, которые должны быть зафиксированы по двум направлениям – от источника требований к реализации и наоборот.

В случае трассирования требований от продукта, движение в обратном направлении, т.е. к требованиям, можно выяснить, как написана каждая строка этого продукта и соответствует ли она отдельным атрибутам требований. Связи трассируемости требований помогают найти незапланированные и реализованные некоторые функции или фрагменты программ, не соответствующие заданным требованиям. И, наоборот, выявить нереализованные требования к функциональности. Взаимосвязи и зависимости между отдельными требованиями сохраняются, например, в таблице трассируемости, удаляются или модифицируются при различных изменениях.

Процедура трассирования состоит в следующем:

- выбирается элемент (функция, фрагмент или некоторая часть) из матрицы трассирования требований, за которым проводится прослеживание на этапах ЖЦ;
- составляется список вопросов, по которым на каждом этапе проверяются связи при *реализации требований* в продукте, и если изменяется какое-то звено в цепочке требований, то может модифицироваться процедура разработки этого элемента на последующем этапе ЖЦ;
- проводится мониторинг статуса каждого требования на соответствие выполнения согласно принятому плану;

- уточнение ресурсов выполнения проекта при необходимости проведения изменений в требования и в элементы проекта.

Условием принятия решения о возможных модификациях требований и результатов промежуточного проектирования, является обновленная информация о связях между различными частями системы и первоначально заданными требованиями к ним.

Требования к оформлению отчета

Структура отчета руководителя:

1) постановка общей задачи
2) общая функциональная схема ПО
3) таблица трассировки требований из разделов «Общие требования к системе» и «Требования к функциям» в ТЗ в виде (пример):

Требование	**Отметка о выполнении**	**Функция (описание)**	**Функция (определение)**
Система должна предоставлять возможность загружать текстовый документ	выполнено	В системе реализован модуль, выполняющий загрузку и хранение данных в формате .doc и .docx	Модуль Textgr Функция txtgrr(…)

4) интеграционное (сборочное) тестирование реализованных участниками модулей ПО и ведение общего журнала тестирования с подробным описанием обнаруженных ошибок и внесенных исправлений
5) Руководство пользователя (итоговый вариант)
6) Руководство администратора.

Структура отчетов сотрудников:

1) постановка индивидуальных подзадач

2) программная реализация согласно разработанной ранее модели деятельности
3) проведение тестирования реализованных функций и заполнение журнала тестирования
4) выполнение индивидуальных задач по разработке руководств.

Структура отчета секретаря:

1) постановка общих задач
2) стенограмма «мозгового штурма»
3) распределение индивидуальных подзадач
4) программная реализация согласно разработанной ранее модели деятельности
5) проведение тестирования реализованных функций и заполнение журнала тестирования
6) выполнение индивидуальных задач по разработке руководств.

По итогам проведенных мероприятий и выполненных работ вся группа формирует доклад о результатах выполнения задания, с которым участвует в устном отчете. Для отчета рекомендуется сформировать тезисы либо презентацию по итоговому отчету руководителя. Основной докладчик – руководитель. На дополнительные вопросы отвечает вся группа, либо автор предложения.

СПИСОК ЛИТЕРАТУРЫ

1) Инженерный подход к подготовке специалистов по программной инженерии [Электронный ресурс]// http://habrahabr.ru/post/158637/.
2) Абрамова О.Ф. CASE-технологии: изучать или исключить? / Абрамова О.Ф. // Alma mater (Вестник высшей школы). - 2012. - № 9. - С. 109-110.
3) Абрамова О.Ф. CASE-технологии: нужны ли они высшей школе? [Электронный ресурс] / Абрамова О.Ф. // 11-я научно-практическая конференция профессорско-преподавательского состава ВПИ (филиал) ВолгГТУ (г. Волжский, 27-28 янв. 2012 г.) : сб. матер. конф. / ВПИ (филиал) ВолгГТУ. - Волгоград, 2012. - 1 электрон. опт. диск (CD-ROM). - С. 323-325.
4) Абрамова О. Ф. Формирование образа мышления современного специалиста с помощью CASE-технологий/ Абрамова О. Ф.// Известия ВолгГТУ. Серия "Новые образовательные системы и технологии обучения в вузе". Вып. 10 : межвуз. сб. науч. ст. / ВолгГТУ. - Волгоград, 2013. - № 13 (116). - С. 10-12
5) Валеева Н. А. Дискуссионная технология "Аквариум" в школьном обучении// Режим доступа: https://открытыйурок.рф/статьи/532111/
6) Абрамова О.Ф. Введение в программную инженерию: методические указания к лабораторной работе на тему "Основные сведения о UML и BOUML. Диаграммы вариантов использования" [Электронный ресурс] Сборник «Методические указания». Выпуск 2. [Электронный ресурс] / О.Ф. Абрамова, Д.Н. Лясин. - Волгоград: ВолгГТУ, 2013. - номер гос. регистрации 0321301999- 1 электрон. опт. диск (CD-ROM).
7) Лясин Д.Н. Объектно-ориентированный анализ и программирование [Электронный ресурс] Сборник "Учебные пособия". Выпуск 1. [Электронный ресурс] / Д.Н. Лясин, О.Ф. Абрамова. - Волгоград: ВолгГТУ, 2014. - номер гос. регистрации 0321400870- 1 электрон. опт. диск (CD-ROM).

8) Абрамова О.Ф. Введение в программную инженерию: методические указания к лабораторной работе на тему "Основные сведения о UML и BOUML. Диаграммы активности [Электронный ресурс] Сборник «Методические указания». Выпуск 1. [Электронный ресурс] / О.Ф. Абрамова, Д.Н. Лясин. - Волгоград: ВолгГТУ, 2014. - номер гос. регистрации 0321400872- 1 электрон. опт. диск (CD-ROM).

9) Лясин Д.Н. Основы объектно-ориентированного анализа программных систем [Электронный ресурс] Сборник «Методические указания». Выпуск 4. [Электронный ресурс] / Д.Н. Лясин, О.Ф. Абрамова. - Волгоград: ВолгГТУ, 2014. - номер гос. регистрации 0321402233- 1 электрон. опт. диск (CD-ROM).

10) Абрамова О.Ф. Введение в программную инженерию: методические указания к лабораторной работе на тему "Основные сведения о UML и BOUML. Диаграммы взаимодействия" [Электронный ресурс] «Методические указания». Выпуск 2. [Электронный ресурс] / О.Ф. Абрамова, Д.Н. Лясин. - Волгоград: ВолгГТУ, 2015. - свид. о регистрации № 20915- 1 электрон. опт. диск (CD-ROM).

11) Лясин Д. Н. Объектно-ориентированный анализ и проектирование программных систем учебное пособие. / Д. Н. Лясин, О.Ф. Абрамова. - Волгоград: ВПИ (филиал) ВолгГТУ, 2015. - 100с.

12) Абрамова, О.Ф. Методические указания по курсовому проектированию по дисциплине "Конструирование программного обеспечения" [Электронный ресурс] «Методические указания». Выпуск 2. [Электронный ресурс] / О.Ф. Абрамова, Д.Н. Лясин. - Волгоград: ВолгГТУ, 2015. - свид. о регистрации № 20915- 1 электрон. опт. диск (CD-ROM).

13) Лясин Д.Н. Основы объектно-ориентированного программирования на языках C++ и C# [Электронный ресурс] «Методические указания». Выпуск 2. [Электронный ресурс] / Д.Н. Лясин, О.Ф. Абрамова. - Волгоград: ВолгГТУ, 2015. - свид. о регистрации № 20915- 1 электрон. опт. диск (CD-ROM).

14) Лясин Д. Н. Структурный анализ и проектирование программных систем методические указания. Вып. 3. / Д. Н. Лясин, О. Ф. Абрамова. - Волгоград: ВолгГТУ, 2015. - 24с.

15) Лясин Д.Н. Наследование классов и полиморфизм в объектно-ориентированном программировании [Электронный ресурс] методические указания. [Электронный ресурс] / Д.Н. Лясин, О.Ф. Абрамова. - Волжский, 2016. - 38 с. (Режим доступа: http://library.volpi.ru/csp/library/PDF/31413.pdf)

Электронное учебное издание

Оксана Федоровна **Абрамова**

Дмитрий Николаевич **Лясин**

МЕТОДИКА ПРИМЕНЕНИЯ ПРИНЦИПОВ РОЛЕВОГО ПОДХОДА ДЛЯ ОРГАНИЗАЦИИ ЦИКЛА ЛАБОРАТОРНЫХ ЗАНЯТИЙ

Учебно-методическое пособие

Электронное издание сетевого распространения

Редактор Матвеева Н.И.

Темплан 2019 г. Поз. № 19.
Подписано к использованию 06.05.2019. Формат 60x84 1/16.
Гарнитура Times. Усл. печ. л. 3,94.

Волгоградский государственный технический университет.
400005, г. Волгоград, пр. Ленина, 28, корп. 1.

ВПИ (филиал) ВолгГТУ.
404121, г. Волжский, ул. Энгельса, 42а.