Методические указания к лабораторному практикуму   
по курсу: “**Проектирование программ в интеллектуальных системах**”   
5 семестр

# Структура лабораторного практикума

Лабораторный практикум состоит из двух частей:

1. Спецификация объектной модели программной системы. Разработка спецификации объектной модели программной системы согласно варианту с использованием средств языка UML.
2. Реализация программных систем на основе модели. Реализация спроектированной программную систему на выбранном языке программирования высокого уровня.
3. Экспертиза программных систем. Провести экспертизу результатов анализа, проектирования и программирования

Разрабатываемая объектная модель и реализуемая объектная модель – это две различные модели, т.е. после выполнения первой части практикума происходит замена на другую объектную модель

# Лабораторная работа 1. Проектирование объектной модели

## Варианты индивидуального задания

Типовые варианты индивидуального задания приведены в приложении 1. Можно предложить свою формулировку индивидуального задания. Объем работы по индивидуальному заданию корректируется преподавателем.

## Структура описания объектной модели

1. Текстовое описание разрабатываемой программной системы с указанием трех направлений развития системы.
2. Диаграмма классов и объектов (или ER-диаграмма), описывающая концептуальный уровень.
3. Диаграмма (бизнес) процессов, описывающая концептуальный уровень
4. Диаграмма вариантов использования.
5. Диаграммы последовательности.
6. Базовая диаграмма классов.
7. Диаграммы классов, описывающие варианты развития.
8. Диаграмма состояния.
9. Описание пользовательского интерфейса. Диаграмма состояний пользовательского интерфейса.
10. Описание привязки объектной модели к языку программирования и библиотекам.

Результатом разработки объектной модели является набор диаграмм языка UML и текстового описания, который оформляется одним документом в Google Docs.

Для рисования диаграмм можно выбрать любой графический редактор, но рекомендуется использовать редактор, который поддерживает UML версии 2 и выше.

## Текстовое описание разрабатываемой программы

Текстовое описание является зафиксированной информацией о разрабатываемой программе, полученной от “заказчика”. В роли заказчика программного продукта выступает студент, он определяет функциональное содержание программного продукта, преподаватель контролирует объем.

Текстовое описание разрабатываемой программы должно представлять логически связное повествование о программе на русском языке в стиле “школьного сочинения”. В описании должны быть три логических компонента:

* описание структуры программы и ее подсистем;
* описание взаимодействие между подсистемами программы;
* описание основных сценариев работы пользователя с программой.

В конце описания необходимо перечислить **три** направления развития программной системы.

## Описание концептуального уровня

Концептуальный уровень – “верхний” уровень абстракции объектной модели программной системы. Строится на основании текстового описания, разработанного на предыдущем шаге, состоит из трех частей:

* диаграмма классов и объектов;
* диаграмма бизнес процессов;
* диаграмма вариантов использования.

Диаграмма классов и объектов концептуального уровня отображает объекты, классы, отношения между объектами и между классами.

В концептуальном уровне должны присутствовать только те сущности, которые описаны в текстовом описании.

## Разработка диаграмм последовательности

Выбираем один главный вариант использования и для него составляем диаграмму последовательности.

Если выбранный вариант использования имеет зависимые второстепенные варианты использования их надо обозначить на составленной диаграмме классов.

## Разработка диаграммы классов и направлений развития

Создаем одну базовую диаграмму классов, не содержащую реализацию направлений развитий, и для каждого описанного направления создаем отдельную диаграмму классов, которая показывает каким способом, будет изменяться программная система в данном направлении.

## Разработка диаграммы состояний и активности

Создаем для всех методов описанных в диаграмме классов диаграммы состояний или активности.

## Разработка структуры пользовательского интерфейса

Рисуем макеты (wireframe) окон пользовательского интерфейса. Диаграммой состояний описываем переходы между окнами.

Если предполагается только интерфейс командной строки, то описываем входные параметры и формат выводимой информации.

# Этапы разработки объектной модели

|  | Что нужно сделать | Что должно быть в отчете |
| --- | --- | --- |
| Этап 1 | * Определиться с темой и что будет делать программная система * Разработать текстовое описание программной системы * Выделить три направления развития программной системы | * Текстовое описание * Три направления развития |
| Этап 2 | * Продумать концептуальный уровень программной системы * Разработать ER-диаграмму (или диаграмму классов и объектов) для концептуального уровня программной системы * Разработать BPMN-диаграмму бизнес-процесса(ов) концептуального уровня программной системы | * Текстовое описание * Три направления развития * ER-диаграмма * BPMN-диаграмма |
| Этап 3 | * Разработать диаграмму вариантов использования * Продумать логический уровень программной системы * Разработать диаграмму последовательности одного из главного варианта использования для логического уровня программной системы | * Текстовое описание * Три направления развития * ER-диаграмма * BPMN-диаграмма * Use Case диаграмма * Sequence диаграмма(ы) |
| Этап 4 | * Разработать базовую диаграмму классов для программной системы с учетом выделенных направлений развития * Разработать для каждого направления развития отдельной диаграммы классов, на которой будет показано (выделено) какие будут происходить изменения | * Текстовое описание * Три направления развития * ER-диаграмма * BPMN-диаграмма * Use Case диаграмма * Sequence диаграмма(ы) * Базовая диаграмма классов * 3 диаграммы классов по направления развития |
| Этап 5 | * Если есть сложное поведение, то его описываем отдельной диаграммой состояний или активности * Разработать макеты окон пользовательского интерфейса * Разработать диаграмму состояний, описывающую переходы между окнами | * Текстовое описание * Три направления развития * ER-диаграмма * BPMN-диаграмма * Use Case диаграмма * Sequence диаграмма(ы) * Базовая диаграмма классов * 3 диаграммы классов по направления развития * Макеты окон * Диаграмма состояний пользовательского интерфейса * Рекомендации по реализации объектной модели |

# Лабораторная работа 2. Реализация объектной модели

Для выданной преподавателем объектной модели вам надо сделать следующие задания:

1. Составить список ошибок. Какие именно ошибки можно встретить описано в приложении 3
2. Сделать реализацию описанной объектной модели. Залить исходники на github и в отчете вставить ссылку.

Язык реализации Python / Java / C++. Реализация должны быть совпадать с описание объектной модели, все, что не описано (не детализировано) можно придумывать самостоятельно. Возможно отклонения от описания, но при обязательном согласовании с преподавателем.

Реализация должна точно содержать описанные классы и пользовательский интерфейс. Приложение должно запускаться и должен осуществляться переход между окнами. Внутренняя логика может отсутствовать.

# Лабораторная работа 3. Экспертиза программных систем

# Провести экспертизу результатов анализа, проектирования и программирования. Подготовить отчет с описанием:

1. Проблем проведенного анализа. Что было не учтено на этапе анализа.
2. Проблем спроектированной объектной модели. Какие ошибки и неточности содержат разработанные диаграммы.
3. Проблем реализации модели. Описать несоответствия с объектной модели и обосновать невозможность привести реализацию с моделью

# 

# Задача

Задача №1. Построение BPMN диаграммы для бизнес-процесса

Дано: в приложении 2 приведены названия процессов из нашей жизни

Надо: декомпозировать процесс на действия, выделить акторов, определить артефакты и нарисовать диаграмму этого процесса на языке BPMN

# Приложение 1. Типовые варианты индивидуального задания для ЛР №1

1. Платформа учета и анализа клиентского опыта и отзывов
2. Электронная система бронирования для ресторанов и кафе
3. Система мониторинга здоровья с использованием носимых гаджетов
4. Интерактивная образовательная платформа для дистанционного обучения
5. Система автоматизированного учета и анализа энергопотребления в доме
6. Система управления инвентарем для розничных магазинов
7. Платформа для виртуальных выставок и ярмарок
8. Система для автоматизированного контроля качества продукции
9. Умный домашний ассистент с голосовым управлением
10. Платформа для создания и управления корпоративными блогами
11. Электронная система учета рабочего времени сотрудников
12. Платформа для проведения виртуальных экскурсий и путешествий
13. Электронная система для управления и бронирования гостиничных номеров
14. Платформа для проведения онлайн-конференций и вебинаров
15. Система автоматизации учета и отслеживания посылок в логистических компаниях
16. Система управления и бронирования парковочных мест
17. Система мониторинга состояния растений для сельского хозяйства
18. Интерактивная система для детского обучения и развития
19. Платформа для создания и редактирования видеоконтента
20. Система автоматизированного управления клиентской базой для бизнеса
21. Платформа для создания и управления мероприятиями
22. Система контроля доступа и безопасности на предприятиях
23. Платформа для проведения онлайн-аукционов
24. Система учета и мониторинга движения финансов
25. Платформа для обмена и продажи цифровых товаров
26. Система автоматического детектирования и реагирования на аварийные ситуации
27. Платформа для обмена знаниями и опытом между специалистами
28. Платформа для создания и управления корпоративными блогами
29. Электронная система для учета и управления финансами личного бюджета
30. Платформа для совместного создания и редактирования документов
31. Система автоматического учета трафика на сайтах и анализа поведения пользователей

# Приложение 2. Темы для задачи №1 “Построение BPMN диаграммы для бизнес-процесса”

1. Бронирование столика в кафе с внесением аванса
2. Вакцинация и получение сертификата
3. Вызов скорой помощи для друга и сопровождение его в больницу
4. Занятие в тренажерном зале с тренером
5. Заселение в общежитие
6. Лечение болезни
7. Организация вечеринки
8. Организация выставки
9. Организация конференции
10. Организация концерта
11. Организация своего дня рождения
12. Организация субботника
13. Оформление банковской карточки
14. Пересдача экзамена
15. Подготовка и сдача курсового проекта
16. Поездка автостопом
17. Поиск и заселение в квартиру с другом/подругой
18. Поиск работы, прохождение собеседований
19. Покупка билета в кино
20. Покупка и постановка на учет автомобиля
21. Покупка путевки, бронирование авиабилета и гостиницы
22. Получение кредита
23. Получение нового паспорта
24. Получение справки о состоянии здоровья
25. Посадочно-контрольные процедуры в аэропорту
26. Поход в магазин за продуктами
27. Поход к врачу
28. Предложение руки и сердца девушке, дарение кольца
29. Ремонт в комнате
30. Сдача в ремонт одежды или обуви
31. Сдача на право вождения автомобиля
32. Сдача ПЦР-теста
33. Стрижка в парикмахерской с предварительной записью
34. Эвакуация из застрявшего лифта

# Приложение 3. Список для проверки объектной модели

## 1. Общие требования

1.1. Разделы в пояснительной записке должны идти в следующем порядке:

1. текстовое описание,
2. описание трех направлений развития,
3. ER-диаграмма,
4. BPMN-диаграмма(ы)
5. диаграмма вариантов использования,
6. диаграмма(ы) последовательности,
7. базовая диаграмма классов,
8. три диаграммы классов,
9. диаграммы состояний или деятельности для сложной логики (может отсутствовать),
10. макеты окон пользовательского интерфейса,
11. диаграмма состояний пользовательского интерфейса,
12. рекомендации по реализации

1.2. Размер текста на диаграммах должен читабельный.

1.3. Цветовое выделение на диаграмме должно иметь смысл.

1.4. Имена сущностей в тексте и на диаграммах должны быть на одном языке (или на русском или на английском)

1.5. Одинаковые сущности должны иметь одинаковые имена

## 2. Текстовое описание

2.1. Текстовое описание должно содержать:

1. описание структуры предметной области,
2. описание динамики (логики, процессов) предметной области,
3. описание сценариев поведения пользователей программной системы

2.2. Три направления развития программной системы

## 3. Диаграмма сущность-связь

3.0. Можно заменить на диаграмму объектов/классов

3.1. Должны быть представлены все сущности, которые упоминаются в текстовом описании

3.2. На отношения должны быть указаны множители и название связей

## 4. Диаграмма бизнес-процессов

## 5. Диаграмма вариантов использования

5.1. Акторы должны быть внешними сущностями по отношению к программной системе

5.2. Должны быть обозначены границы системы и система должна иметь название, которое соответствует текстовому описанию

5.3. Варианты использования связанные напрямую с акторами должны включать действия от начала использование системы до завершения и иметь завершенную значимость для системы.

5.4. Название актора должно быть конкретным. Никаких “пользовать” или “человек”

## 6. Диаграмма последовательности

6.1. Для одного варианта использования, который связан напрямую с акторами, должен быть нарисована диаграмма последовательности

6.2. Акторы могут только отправлять асинхронные сообщения

6.3. Все объекты должны относиться к классам с базовой диаграммы классов

6.4. Сообщения к объектами должны соответствовать методам соответствующих классов

## 7. Диаграммы классов

7.1. Базовая диаграмма классов не должна содержать функционал из направлений развития

7.2. Для каждого направления должна быть отдельная диаграмма классов, которая содержит только новые классы и те которые связаны с новыми. Новые классы нужно выделить, например, цветом.

7.3. Для каждого отношения ассоциации должны быть проставлены множители

7.4 Для каждого отношения ассоциации должны быть указана навигация и роль

7.5. Методы get и set для атрибутов класса указывать не нужно

7.6. Если между классами есть наследование, то должно быть понятно какие методы переопределяются. Исключение для классов данных

7.7. На базовой диаграмме классов должен присутствовать класс со стереотипом <<main>> и являться точкой входа.

7.8. На диаграмме классов нужно выделить класс, который реализует dependency injection. Класс должен быть со стереотипом <<injector>>. От этого класса должны идти отношения зависимости со стереотипом <<create>>. Может совпадать с классом <<main>>

7.9. Должно быть понятно какой класс отвечает за отображение пользовательского интерфейса (указывать стереотип <<UI>>) и какие методы реализуют обработку пользовательских действий

7.10. Отношение агрегации и композиции должно рисовать ромбом к классу, который является “целым”

7.11. Если два класса связанные отношение зависимости <<use>>, должно быть понятно каким образом один класс узнает про другой: через передачу параметра в метод, через отношение ассоциации

7.12. При указании навигации в отношении ассоциации стрелка указывает класс, который доступен (т.е. виден)

7.13. При описании направления развития учитываем принцип открытости/закрытости (из SOLID) - старые классы не меняем, а только от них наследуемся или используем

7.14. Если старый класс должен знать о новом классе, то он должен быть изменен, а это соответственно плохо, см. предыдущий пункт

7.15. Выделенные интерфейсы или базовые классы должны использоваться, т.е. должны быть связаны отношениями зависимости и ассоциации или только ассоциацией

7.16. Не перечислять в классе атрибуты, которые имеют тип одного из разрабатываемых классов.

7.17. Для классов данных указывать стереотип <<data>>. В них не должно быть методов

7.18. “Пустых” классов не должно быть

7.19. Неоправданное использование синглетонов

7.20. Классы <<injector>>, <<main>>, <<UI>> не должен нарушать принципа единственности ответственности

7.21. Если в классе есть атрибут типа A, не должно быть связи зависимости к A - должна быть связь ассоциация

7.22. Класс данных не должен реализовывать никаких интерфейсов

7.23. В диаграммах класса для направлений развития должно быть понятно кто будет создавать новые классы и кто будет вызывать методы новых классов

7.24. При наследовании в производном классе надо указывать только те методы, которые переопределяются и новые, кроме этого данные не должны дублироваться

7.25. Абстрактный класс (<<abstract>>) не может быть связан отношением реализации с другим классом.

7.26. Должно быть понятно кто создает объекты класса.

## 8. Диаграмма состояний

8.1. Основная диаграмма состояний должна описывать переходы между экранами пользовательского интерфейса

8.2. Дополнительные диаграммы состояний могут описывать сложное состояние внутренних объектов системы

8.3. Состояние на диаграмме означает ожидание события: действия пользователя, прихода сообщения, генерации исключительных ситуации

8.4. В состоянии и на переходах указываются действия, которые должны соответствовать методам классов из базовой диаграммы

8.5. Должно быть одно начальное состояние и возможно несколько конечных состояний. У других состояний всегда должна быть хотя бы одна входящая и выходящая связь

## 9. Диаграмма деятельности

9.1. Если диаграмма описывает логику, которая задействует более одного метода, нужно указывать конкретные названия методов

9.2. Логика диаграммы не должны дублировать логику диаграммы состояний

## 10. Макеты пользовательского интерфейса

## 11. Диаграмма состояний пользовательского интерфейса