Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования «Алтайский государственный университет»

Кафедра прикладной информатики в экономике, государственном и
муниципальном управлении

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине «Разработка отраслевых прикладных программных решений»

Уровень образовательной программы *магистратура*
Направление подготовки 09.04.03. Прикладная информатика
Профиль «Управление информационными системами в бизнесе»

Разработчик:
к.ф.-м.н., доцент кафедры
прикладной информатики в
экономике, государственном и
муниципальном управлении

_____ /О. В. Журенков/

Барнаул, 2020

Визирование ФОС для исполнения в очередном учебном году

Фонд оценочных средств пересмотрен, обсуждён и одобрен для исполнения в 2020–2020 учебном году на заседании кафедры прикладной информатики в экономике, государственном и муниципальном управлении.

Внесены следующие изменения и дополнения:

Протокол от 14.05.2020 № 10
Зав. кафедрой А. Ю. Юдинцев

1. Перечень компетенций, с указанием этапов их формирования в процессе освоения образовательной программы

Компетенции	**Показатели**	**Наименование оценочного средства**
ПК-2: Способен проектировать архитектуру ИС предприятий и организаций в прикладной области	Знает: • терминологию (понятийный аппарат) объектно-ориентированного анализа, проектирования, программирования и тестирования программных систем. Умеет: • выполнять системный анализ предметной области для построения концептуальных схем разрабатываемого проекта. Владеет: • методами объектно-ориентированного анализа требований к программному обеспечению корпоративной информационной системе предприятия.	Практические задания к лабораторным работам. Контрольные вопросы. Тест.
ПК-5: Способен использовать передовые методы оценки качества,надежности и информационной безопасности ИС в процессе эксплуатации прикладных ИС	Знает: • этапы и стадии разработки программных продуктов; • базовые принципы объектно-ориентированного моделирования систем и принципы проектирования сложных систем. Умеет: • читать и анализировать требования к ПО, представленные в виде диаграмм UML. Владеет: • методами объектно-ориентированного анализа при проектировании программных продуктов для различных предметных областей (в рамках направления обучения).	Практические задания к лабораторным работам. Контрольные вопросы. Тест.
ПК-6: Способен использовать информационные сервисы для автоматизации прикладных и информационных процессов	Знает: • технологию разработки проекта программной системы на базе унифицированного языка UML. Умеет: • создавать модели и диаграммы согласно концепции MDA в современной прикладной среде проектирования. Владеет: • технологией анализа проектов программных продуктов.	Практические задания к лабораторным работам. Контрольные вопросы. Тест.

ПК-7: Способен интегрировать компоненты и сервисы ИС	Знает: • способы (шаблоны) и этапы построения проекта в среде моделирования. Умеет: • создавать UML-диаграммы согласно объектно-ориентированному подходу к разработке ПО; • применять на практике основные шаблоны построения проекта в современной среде проектирования. Владеет: • современным инструментарием проектирования программных продуктов.	Практические задания к лабораторным работам. Контрольные вопросы. Тест.

2. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Показатели выявляются путём соотнесения критериев: когнитивный (знания), инструментальный (умения, навыки), праксеологический (опыт), с этапами процесса формирования компетенций, охарактеризованными выше.

Поскольку для промежуточного контроля используется такая форма контроля, как зачёт, по итоговому баллу (в 100-балльной шкале) ставится отметка, в соответствии с табл. 2.

Таблица 2.

Сопоставление шкал оценивания

100-балльная шкала	0–49	50–69	70–89	90–100
Бинарная шкала	Не зачтено	Зачтено		

Для оценивания выполнения практических заданий применяются следующие показатели:

1. полнота выполнения задания;
2. своевременность выполнения задания;
3. логическая последовательность и рациональность выполнения задания;
4. уровень самостоятельности выполнения задания;
5. уровень творчества и новаторства при выполнении задания.

В соответствии с этими показателями, на основе табл. 3 выставляется оценка за каждое выполненное задание.

При выполнении тестирования обеспечивается самостоятельность выполнения тестов: из аудитории удаляются посторонние, преподавателем контролируется неиспользование слушателем интернет-источников, учебников и иных пособий, за исключением личного конспекта слушателя (допускается, как рукописный, так и электронный вариант). Тестирование проводится в ЭУМК на базе образовательного портала АлтГУ, за ограниченное время. Оценка выставляется автоматически по окончанию теста или отведённого времени.

Таблица 3.

Оценивание выполнения практических заданий

4-балльная шкала (уровень освоения)	**100-балльная шкала**	**Критерии**
Отлично (повышенный уровень)	90–100	Студентом задание выполнено самостоятельно и в срок. При этом составлен правильный алгоритм выполнения задания, в логических рассуждениях, в выборе ПО и методах его применения нет ошибок, получен верный ответ, задание выполнено рациональным способом.
Хорошо (базовый уровень)	70–89	Студентом задание выполнено с небольшими подсказками преподавателя, возможно, с небольшой задержкой сроков. При этом составлен правильный алгоритм выполнения задания, в логическом рассуждении и решении нет существенных ошибок; правильно сделан выбор ПО и методов его применения для выполнения задания; есть объяснение выполнения, но задание выполнено не рациональным способом или допущено не более двух несущественных ошибок, получен верный ответ.
Удовлетворительно (пороговый уровень)	50–69	Студентом задание выполнено с подсказками преподавателя, с большой задержкой сроков. При этом задание понято правильно, в логическом рассуждении нет существенных ошибок, но, возможно, допущены существенные ошибки в выборе ПО и методов его применения или в составлении документации; задание выполнено не полностью или в общем виде.
Неудовлетворительно (уровень не сформирован)	0–49	Студентом задание не выполнено.

3. Типовые контрольные задания или иные материалы, необходимые для оценки планируемых результатов обучения по дисциплине, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

3.1. Примерные темы лабораторных работ, практических, индивидуальных заданий

Лабораторная работа №1

Описание функциональности проектируемой информационной системы с использованием диаграмм прецедентов

Цель работы: Описать весь функционал организации, имеющий значение для проектируемой ИС. Описать сценарии для наиболее значимых прецедентов.

Функциональные требования к системе (т. е. функциональность, которую должна предоставлять система) документируются во время разработки с помощью модели *прецедентов*, которая иллюстрирует планируемые функции системы (прецеденты), их окружение (актёры) и связи между прецедентами и актёрами (диаграммы прецедентов).

Модель прецедентов описывает функции системы с точки зрения внешнего пользователя, т. е. отражает взгляд на деятельность организации извне.

1. Запустите IBM Rational Software Architect Designer. Выберите рабочую область на диске U: (например, в папке «IBM»). Создайте новый проект.

2. По технологии RUP, первым делом надо построить *модель прецедентов*. Для создания такой модели выберите в контекстном меню окна Структура проектов ▷ Создать▶ Проект. . . ▶ Моделирование▶ Проект UML, задайте имя своего проекта (название разрабатываемой системы,

можно на русском языке). Далее, выберите модель: в категории Требования (Requirements) — Пакет прецедента (Use Case Model). Остальные настройки оставьте по умолчанию.

3. IBM Rational Software Architect Designer применяет при создании новых моделей *шаблоны моделей*. Шаблон содержит следующие элементы модели:

- Обзорный пакет `Overviews`, который содержит две диаграммы — `Actors overview` (визуализация актёров в модели) и `Context Diagram` (демонстрирует «важнейшие» прецеденты в модели).
- Пакет `Use-Case Building Blocks` со строительными блоками для прецедентов.
- Универсальный пакет актёров `Versatile Actors`.

  Пакет `Use-Case Building Blocks` содержит элементы модели, которые могут быть скопированы в модели прецедентов, позволяя создавать элементы для собственного применения. Первый элемент в пакете называется `${functional.area}`, и этот пакет содержит диаграмму прецедентов. Для большой модели с помощью данного пакета можно сгруппировать прецеденты в функциональные области. Второй элемент модели называется `${use.case}` и является прототипом прецедента, содержащим опциональные диаграммы деятельности и последовательностей. Этот элемент модели может быть использован в любых типах моделей.

  Пакет `Versatile Actors` содержит диаграмму прецедентов, которая демонстрирует всех актёров, общающихся с прецедентами, выходящими за пределы функциональных областей.
- Скопируйте шаблон *функциональной области*(`${functional.area}`) в свою модель. Выберите в контекстном меню ▷ Найти/Заменить... (Find/Replace...) и замените `${functional.area}` на Ваше новое имя функциональной области (например, `Оформление заказа`).

  Аналогично создайте все остальные необходимые области.

4. Используя контекстную панель инструментов (контекстное меню), создайте актёров (бизнес-актёров, сотрудников).

Актёры не являются частью системы — они представляют что угодно (или кого угодно), что взаимодействует с системой. Актёр может:

- только вводить в систему информацию;
- только получать информацию от системы;
- вводить в систему информацию и получать из неё информацию.

Обычно актёры выявляются на основе изучения предметной области или по результатам общения с заказчиком и экспертами. Для определения актёров в системе можно использовать следующие вопросы;

- Кто заинтересован в данных требованиях?
- Где в организации будет применяться данная система?
- Кто выигрывает от использования системы?
- Кто обеспечивает систему информацией и применяет эту информацию, а также удаляет информацию?
- Кто занимается поддержкой системы?
- Использует ли система внешние ресурсы?
- Выполняет ли один человек несколько ролей?
- Выполняют ли несколько людей одну и ту же роль?
- Взаимодействует ли система с уже действующими системами?

Для каждого актёра, в окне *Properties* на закладке Documentation, можно задать описание.

5. Скопируйте шаблон `${use.case}` в свою функциональную область. Выберите в контекстном меню ▷ Найти/Заменить... (Find/Replace...) и замените `${use.case}` на Ваше новое имя прецедента (например, `консультирование`).

6. Создайте все необходимые *прецеденты* (Use Cases), задайте их имена и стереотипы.

   Для выявления прецедентов, определите, представляет ли он некую законченную функциональность, которую инициирует один актёр. Для включения в диаграмму прецеденты должны удовлетворять следующим критериям:

   - прецедент должен описывать, **что** нужно делать, а не **как**;
   - прецедент должен описывать действия с точки зрения **исполнителя** (организации-заказчика);
   - прецедент должен возвращать исполнителю некоторое **сообщение**;
   - последовательность действий внутри прецедента должна представлять собой одну **неделимую** цепочку.

   При выявлении прецедентов руководствуйтесь следующими правилами:

   6.1. Избегайте «множества» прецедентов (руководствуйтесь правилом «5 ± 2»). Даже очень сложные системы редко имеют больше 50 прецедентов. Для таких систем используют несколько диаграмм прецедентов (с различным уровнем детализации и с разбиением на функциональные области).

   6.2. Избегайте «мелких» прецедентов — прецедентов, чьи имена означают отдельные части функциональности. Например, «Ввод в базу данных . . . » не может быть отдельным прецедентом. На бизнес уровне вообще не стоит выражаться в терминах баз данных и программного кода (используйте бизнес термины предметной области).

7. Для использования классического оформления всех диаграмм следует зайти в меню ▷ Окно ▶ Параметры ▶ Моделирование ▶ Внешний вид и поставить в качестве Темы по умолчанию тему Моделирование UML - Классическая.

8. Выделите *включаемые* и *расширяющие* прецеденты. Свяжите их соответствующими отношениями.

9. Установите необходимые *отношения* между объектами диаграммы.

10. Снабдите диаграмму *заметками*.

11. Для основных прецендентов (не менее двух) напишите текстовый *сценарий*:

    - опишите «Главный раздел», включающий *Имя прецедента*, *Идентификатор прецедента*, *Краткое описание*, *Актёров*;
    - опишите «Предусловия» — состояние системы до начала прецедента (условия, которые должны выполниться, чтобы прецедент мог осуществиться).
    - опишите «Типичный ход событий» (основной поток), указав последовательность действий актёров и системы, а также возможные *исключения* (отклонения от основного сценария);
    - опишите «Постусловия» — состояние системы после окончания прецедента (условия, которые должны выполниться по окончанию прецедента).
    - опишите «Альтернативные потоки» (исключения), если они есть, используя аналогичный шаблон;
    - при необходимости, напишите «Примечания».

    Разместите сценарии в разделе документация соответствующих прецедентов.

Оформите отчёт, содержащий *диаграмму прецедентов* и *сценарии*.

Лабораторная работа №2

Описание логики реализации функционала проектируемой информационной системы с использованием UML диаграмм деятельности

Цель работы: Описать алгоритм выполнения основных прецедентов (функционала), имеющих значение для проектируемой ИС.

Диаграмма деятельности демонстрирует поток управления и поток данных внутри прецедента.

В данном задании необходимо построить диаграммы деятельности для основных *прецедентов* (не менее двух).

1. Запустите IBM Rational Software Architect Designer. Откройте свою рабочую область.

2. Откройте диаграмму прецедентов. Для этого можно в дереве проектов выбрать ▶ Диаграммы ▶ Диаграммы прецедентов. В открывшемся списке диаграмм найдите диаграмму, созданную в предыдущей работе (по названию рабочей области).

3. Выберите интересующий Вас прецедент в дереве проектов (в ветке ▶ Модели ▶ Модель прецедента ▶ <Название рабочей области>). Через контекстное меню откройте соответствующую *диаграмму деятельности* (Activity Diagram). Если нужный Вам прецедент не содержит *диаграмму деятельности*, то создайте пустую *диаграмму деятельности* для этого варианта использования, используя контекстное меню: ▷ Добавить диаграмму ▶ Диаграмма видов деятельности (▷ Add Diagram ▶ Activity Diagram).

4. Обязательно задайте *исполнителей*, используя области — инструмент Раздел (Partition). Учтите, что часть работ должна выполнять *система*. Для каждого элемента диаграммы свойства задаются в нижнем окне, на закладке Свойства (Properties).

5. Используя соответствующую палитру инструментов или контекстное меню, создайте диаграмму деятельности. За основу возьмите *сценарии* к прецедентам, однако диаграмма деятельности не обязана в точности повторять сценарий, она должна детализировать и уточнять ранее разработанные артефакты (в данном случае таким артефактом является прецедент).

6. Часто возникает необходимость визуализации потока данных наряду с потоком управления. В этом случае используются инструменты группы Узел объекта (Object Node).

7. Выполните проверку построенной диаграммы (через меню ▷ Моделирование, контекстное меню или соответствующей пиктограммой). Исправьте обнаруженные ошибки.

8. Проверьте работу алгоритма (через меню, контекстное меню или соответствующей пиктограммой). Если обнаружите не соответствующее Вашим замыслам выполнение алгоритма, внесите необходимые корректировки.

Оформите отчёт, содержащий *сценарии* из предыдущей работы и *диаграммы деятельности*.

Лабораторная работа №3

Модель анализа информационной системы: диаграммы классов

Цель работы: Описать все необходимые классы ИС и взаимосвязи между ними.

Следующим этапом проектирования ИС (по методологии RUP) является разработка модели анализа. Модель анализа характеризуется выявлением бизнес-объектов и показывает выполнение бизнес-процессов организации её внутренними исполнителями. Основными компонентами моделей бизнес-объектов являются *внешние* и *внутренние исполнители*, а также *бизнес-сущности*, отображающие всё, что используют внутренние исполнители для реализации бизнес-процессов. В модели анализа также задаются граничные классы (прототипы интерфейсов) и управляющие классы.

Диаграмма классов является основным логическим представлением модели и содержит детальную информацию об архитектуре программной системы.

1. Откройте в IBM Rational Software Architect Designer свою рабочую область (workspace). При этом должен быть доступен проект, созданный в предыдущей работе.

2. Следуя технологии RUP, надо создать *модель анализа*, в которой будет создаваться диаграмма классов в первом приближении (необходимом для анализа проектируемой ИС). Для создания такой модели выберите в контекстном меню своего проекта ▷ Создать ▶ Модель UML, в мастере выберите Стандартный шаблон, категория: Анализ и планирование, шаблон: Пакет анализа RUP, задайте имя модели.

3. IBM Rational Software Architect Designer применяет при создании новых моделей *шаблоны моделей*. Шаблон содержит следующие элементы модели:

   - Обзорный пакет `Overviews`, который содержит 4 диаграммы — `Domain Model` (содержит все классы-сущности), `Key Abstraction` (содержит важнейшие классы в системе), `Key Controllers` (содержит все управляющие классы), `UI` (содержит все классы пользовательского интерфейса).
   - Пакет `Analysis Building Blocks` со строительными блоками, содержит элементы, которые можно копировать в модель анализа. Работа со строительными блоками аналогична построению диаграмм прецедентов.

4. Скопируйте шаблон *функциональной области*(`${functional.area}`) в свою модель. Выберите в контекстном меню Find/Replace... и замените `${functional.area}` на Ваше новое имя функциональной области (например, `оформление заказа`).

5. Используя *проводник проекта*, по вышеописанной технологии, добавьте в созданную функциональную область классы (из соответствующих шаблонов), задействованные в реализации функционала, описанного на диаграмме прецедентов.

   Начните создание классов с *классов-сущностей* («Entity»). Для этого выпишите все существительные из описаний прецедентов и рассмотрите каждое на предмет кандидата в классы. *Классы-сущности* отделены от внешней среды и поведения системы, они могут использоваться во многих прецедентах, с ними связаны информационные объекты (например, клиенты, сотрудники). *Классы-сущности* обычно не имеют операций (как сущности в информационной модели).

   Далее, добавьте *граничные классы* («Boundary»). *Граничные классы* могут появиться при взаимодействии актёра с прецедентом. *Граничные классы* описывают интерфейсы между внешним миром и внутренними классами системы. Внешний мир могут представлять объекты трёх типов: люди (тогда граничный класс представляет класс пользовательского интерфейса), сторонние программные системы (тогда граничный класс представляет класс системного интерфейса), внешние устройства (тогда граничный класс представляет класс аппаратного интерфейса). *Граничные классы* обычно не имеют атрибутов (они используют атрибуты *классов-сущностей*). Рассмотрите все возможные кандидатуры.

   Затем создайте *управляющие классы* («Control») — активные классы, чьи экземпляры координируют поведение системы, соответствующее одному или более прецедентам. Управляющие классы выявляются при рассмотрении поведения системы, описанного прецедентами. Простое поведение можно распределить между граничными классами и классами-сущностями, но для более сложного поведения (например, обработка заказов), лучше добавить управляющий класс (например, `МенеджерЗаказов`). Управляющими классы должны естественным образом вытекать из самой предметной области. Хороший способ выявления управляющих классов — представить себя в роли такого класса. *Управляющие классы* обычно охватывают несколько прецедентов, могут не иметь атрибутов.

   Разделение всех классов на 3 категории — *классы-сущности*, *граничные классы* и *классы управления* — представляет методологию проектирования *MVC* (model-view-control) — *модель-представление-управление*.

6. Для каждого класса, в окне Свойства (Properties) на вкладке Общие (General) задайте *имя* (Name) и Видимость (Visibility). Свойство Abstract (Абстрактный) специфицирует класс как абстрактный (не имеющий экземпляров), а Листовой — не имеющий потомков. В секцию документации данного класса можно ввести поясняющий текст.

7. Добавьте необходимые атрибуты к созданным классам. Добавить атрибут можно через контекстное меню ▷ Добавить UML ▶ Атрибут (▷ Add UML ▶ Attribute) для выбранного класса или в области вкладки Атрибуты в окне свойств соответствующего класса.

8. Добавьте необходимые операции к созданным классам. Добавить операцию можно через контекстное меню ▷ Добавить UML ▶ Операция (▷ Add UML ▶ Operation) для выделенного класса или в области открытой вкладки Операции в окне свойств соответствующего класса.

9. Откройте диаграмму классов Вашей функциональной области. Из дерева проектов перетащите все созданные классы.

10. Добавьте на диаграмму необходимые отношения.

    Для этого можно использовать соответствующие инструменты (стрелки) на контекстной панели инструментов или через контекстное меню.

    - На вкладке Общие (General) в свойствах отношения задайте метку (Label), при необходимости, роли и множественность.
    - На вкладке Стереотипы (Stereotypes) можно задать *стереотип* отношения.
    - Область Ограничения (Constraints) служит для задания *свойств ассоциации* (в виде ограничений).
    - На вкладке Дополнительно (Advanced) можно задать все основные и дополнительные параметры.

11. Выполните проверку построенной диаграммы. Исправьте обнаруженные ошибки.

12. Используя *проводник проекта*, добавьте в созданную функциональную область *реализации* прецедентов (хотя бы 2), используя соответствующий шаблон `${use-case}` в `Analysis Building Blocks`. Назовите созданные реализации (через Find/Replace...) также, как в модели прецедентов.

13. Раскройте внутренний состав первой реализации. Откройте *диаграмму участников* (Participants) и перетащите на неё только те классы, которые участвуют в реализации этого прецедента.

14. Сделайте *диаграмму участников* для второй реализации прецедента.

Оформите отчёт, содержащий все (3) *диаграммы классов*.

Лабораторная работа №4

Описание взаимодействия объектов проектируемой информационной системы с использованием UML диаграмм последовательности

Цель работы: Описать все последовательности реализации основных прецедентов (имеющих наибольшее значение для проектируемой ИС).

В данной работе необходимо построить диаграммы последовательности для основных *прецедентов* (минимум — две). Для этого выполните следующие действия.

1. Запустите IBM Rational Software Architect Designer. Откройте свою рабочую область. Выберите свой проект.

2. Раскройте дерево проектов. Выберите в дереве проектов Модель анализа RUP, а в ней реализацию интересующего Вас прецедента и раскройте её. Из шаблона было создано две диаграммы последовательности: основная (Basic Flow) и альтернативная (Alternative Flow).

3. Откройте диаграмму последовательности для основного потока.

4. Используя палитру инструментов, создайте диаграмму последовательности. Обратите внимание на то, что линии жизни можно создавать «перетягиванием» объектов (из любой области) из дерева проектов на диаграмму.

5. При создании сообщений можно связать его с новой операцией или выбрать операцию, уже заданную для используемого класса.

6. Обязательно используйте комбинированные фрагменты (*Альтернативы*, *Циклы*, и др.), согласно логике выполнения прецедента. Для создания комбинированных фрагментов задайте на диаграмме соответствующие области из набора составной фрагмент.

   Порядок линий жизни и состав линий жизни в комбинированных фрагментах можно изменять через контекстное меню.

Оформите отчёт, содержащий *диаграммы деятельностей* (2), *диаграммы классов* для соответствующих реализаций прецедентов (2) и *диаграммы последовательности* (2).

Лабораторная работа №5

Описание поведения объектов с использованием UML диаграмм состояния

В среде IBM Rational Software Architect Designer постройте две диаграммы состояния (диаграммы конечных автоматов). Одну — для описания поведения одного бизнес-элемента — класса-сущности, важного для бизнеса (например: заявки, заказа, договора, и т. п.). Вторую — для описания поведения главного окна графического интерфейса ИС (как граничного класса).

1. Запустите IBM Rational Software Architect Designer. Откройте свой проект.
2. Выберите в контекстном меню проекта ▷ Создать ▶ Модель UML, в мастере выберите Стандартный шаблон, категория: Анализ и планирование, шаблон: Пустой пакет анализа. На странице детального описания пакета («Сведения о пакете») задайте «Тип пакета» — Модель, Точка наблюдения (viewpoint) — «Бизнес-аналитик».

   Далее, на странице «Функции модели» в разделе «Структурные блоки диаграмм UML» следует добавить «Диаграмма конечного автомата», а в разделе «Структурные блоки элементов UML» следует добавить «Конечный_автомат_1 UML» и «Конечный_автомат_2 UML».
3. В дереве проекта (в ветке «Модель анализа RUP») выберите класс бизнес-элемента и скопируйте его в созданную «Пустую модель анализа». Создайте для этого класса *диаграмму состояния* (State Machine Diagram). Для этого можно использовать контекстное меню: ▷ Добавить диаграмму ▶ Диаграмма конечного автомата.
4. Используя палитру инструментов, опишите состояния этого класса и переходы между ними. Для каждого состояния опишите свойства,
5. Скопируйте в рабочую модель граничный класс главного окна ИС (если такого нет, то создайте его). Далее, аналогичным образом, постройте его диаграмму состояния, задействую внешние (по отношению к нему) объекты (другие окна — граничные классы), не раскрывая их диаграмм.
6. Оформите отчёт, содержащий *диаграмму прецедентов*, *диаграмму классов* и построенные *диаграммы конечных автоматов*.

Лабораторная работа №6

Проектирование подсистем с использованием UML диаграмм компонентов

В среде IBM Rational Software Architect Designer постройте диаграмму компонентов проектируемой ИС.

1. Запустите IBM Rational Software Architect Designer. Откройте свой проект.
2. Выберите в контекстном меню проекта ▷ Создать ▶ Модель UML, в мастере выберите Стандартный шаблон, категория: Анализ и планирование, шаблон: Пустой пакет анализа. Задайте Имя файла: «Модель компонентов». На странице детального описания пакета («Сведения о пакете») задайте «Тип пакета» — Модель, Точка наблюдения (viewpoint) — «Системный аналитик».

   Далее, на странице «Функции модели» в разделе «Структурные блоки диаграмм UML» следует добавить «Диаграмма структуры», а в разделе «Структурные блоки элементов UML» следует добавить «Стереотипный компонент UML», «Составная структура 1 UML» и «Составная структура 2 UML».
3. Создайте *диаграмму компонентов* в этой модели и откройте её в редакторе. Создайте в ней *компоненты*, необходимые для реализации основного функционала ИС. Используйте доступные стереотипы. При необходимости, напишите пояснения в Документации.
4. Постройте *обзорную диаграмму компонентов*. Для этого, используя палитру инструментов или контекстное меню, установите зависимости между компонентами. При этом следите за элементами проекта (особенно при удалении элементов). Отобразить элемент из дерева проекта можно через контекстное меню ▷ Визуализация ▶ Добавить в текущую диаграмму. Найти на диаграмме интересующий элемент из дерева проекта можно через контекстное меню ▷ Навигация ▶ Перейти к диаграмме.

5. Для ключевых компонентов, имеющих структуру, создайте *диаграммы комбинированной структуры* (Composite structure diagram). На этих диаграммах изобразите внутреннюю структуру компонентов, *порты* и *интерфейсы*. Соедините связанные порты (порты, связанные с требуемым и соответствующим предоставляемым интерфейсом).

6. Оформите отчёт, содержащий построенные *диаграмму компонентов* и *диаграммы комбинированной структуры*.

Лабораторная работа №7

Проектирование ИТ-архитектуры ИС

Цель работы: определить наиболее важные (с точки зрения архитектуры) компоненты и спроецировать их на физическое оборудование.

В ООАП, с точки зрения аналитика/проектировщика основная деятельность «Реализации архитектуры» — создание одной (чаще всего) или более *диаграмм развёртывания*.

Диаграмма развёртывания объединяет *компоненты*, *артефакты* и *узлы* для определения физической архитектуры системы.

1. Запустите IBM Rational Software Architect Designer. Откройте свою рабочую область.

2. Создайте *модель развёртывания* (*Deployment Model*), в которой будет создаваться диаграмма развёртывания. Для создания такой модели выберите в контекстном меню проекта ▷ Создать ▶ Модель UML, в мастере выберите Стандартный шаблон, категория: Анализ и планирование, шаблон: Пустой пакет развёртывания, задайте имя модели (со словами `модель развёртывания`). При этом в дереве проектов появится соответствующая модель с пустой диаграммой развёртывания с именем `Main`.

3. Используя контекстное меню диаграммы, или дерева проектов, или палитру инструментов, создайте:

   - *узлы* (*устройства* и *среды исполнения*);
   - *артефакты*.

   Для каждого элемента задайте свойства.

4. Постройте *диаграмму развёртывания*. При связывании устройств коммуникациями, желательно указать стереотип (виды протоколов следует записывать в ключевых словах коммуникации).

   При необходимости, можно разбить *диаграмму развёртывания* на части, добавив соответствующие диаграммы в модель через контекстное меню ▷ Добавить диаграмму.

Оформите отчёт, содержащий полную *диаграмму развёртывания* проектируемой ИС.

3.2. Контрольные вопросы

1. Предпосылки UML. Методология объектно-ориентированного программирования.

2. Методология объектно-ориентированного анализа и проектирования. Общие характеристики UML.

3. Синтаксис и семантика основных объектов UML. Канонические диаграммы языка UML. Механизмы расширения UML

4. Диаграммы прецедентов (вариантов использования) UML. Отношения.

5. Дополнительные обозначения для моделирования бизнес-систем в UML диаграммах прецедентов. Формализация функциональных требований к системе с помощью диаграммы прецедентов (примечания, сценарий).

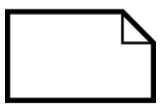
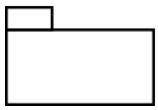
6. Диаграммы деятельности UML. Базовые элементы. Семантика деятельности.

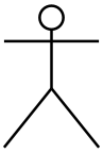
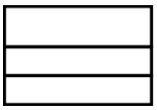
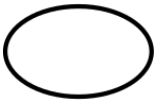
7. Пакеты UML. Модель в UML. Подсистема в UML.

8. Классы UML. Атрибуты класса. Операции класса.
9. Диаграммы классов UML. Отношения между классами.
10. Расширения UML для моделирования бизнес-систем и программного обеспечения (профиль для процесса разработки программного обеспечения, профиль для бизнес-моделирования).
11. Диаграммы последовательности. Элементы диаграммы последовательности (линия жизни, сообщения и сигналы).
12. Комбинированный фрагмент на диаграмме последовательности.
13. Конечные автоматы. Поведенческие и протокольные автоматы. Конечные автоматы и классы. Конечные автоматы в проектировании.
14. Диаграммы состояний: состояния.
15. Диаграммы состояний: переходы.
16. Диаграммы состояний: композитные состояния и регионы.
17. Диаграммы компонентов, основные понятия. Компонент.
18. Элементы диаграммы компонентов: интерфейс, порт, соединитель.
19. Элементы диаграммы компонентов: зависимость, реализация, стереотипы компонентов.
20. UML диаграммы развёртывания: основные понятия. Узел.
21. UML диаграммы развёртывания: артефакт. Спецификация развёртывания.
22. Отношения на диаграмме развёртывания. Стереотипы узлов.

3.3. Тестовые задания

Примерный перечень заданий и вопросов (образец одной попытки):

1. OCL это язык действий для UML, выражения OCL позволяют определить поведение.
   Выберите один ответ:
   - Верно
   - Неверно
2. Укажите простые типы OCL.
   Выберите один или несколько ответов:
   - Boolean
   - Real
   - Char
   - OclVoid
   - Tuple
   - String
   - Byte
   - Integer
3. Выберите название элемента, соответствующего графическому изображению.
   -  Выберите...
   -  Выберите...

-  Выберите...
-  Выберите...
-  Выберите...

4. Действие исполняется, если одновременно на все входящие рёбра поступили маркеры и входящие маркеры удовлетворяют всем локальным предусловиям узла действия.
Выберите один ответ:

- Верно
- Неверно

5. Как называется модельно-ориентированный подход к разработке архитектуры через построение абстрактной метамодели управления и обмена моделями, а также задании способов её трансформации в поддерживаемые технологии программирования?
Выберите один ответ:

- объектно-ориентированное программирование
- предметная область
- архитектура, управляемая моделью
- объектно-ориентированный анализ и проектирование
- разработка, управляемая моделями

6. Как называется специальный элемент модели, который содержит состояния и переходы и является частью композитного состояния или конечного автомата?
Ответ:

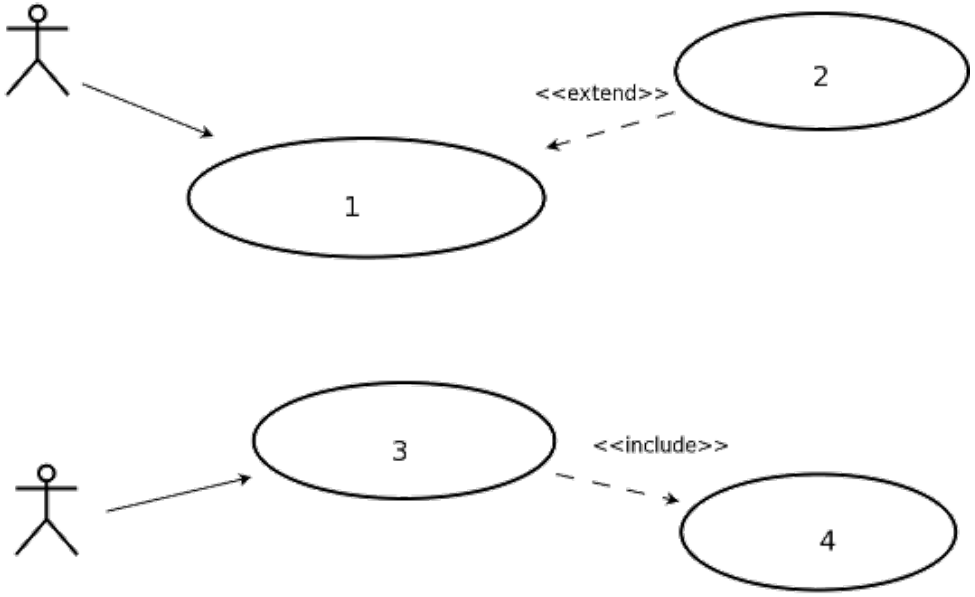
7. Действие исполняется, если хотя бы на одно входящее ребро поступил маркер и он удовлетворяет всем локальным предусловиям узла действия.
Выберите один ответ:

- Верно
- Неверно

8. Как называется отношение, которое обозначает тот факт, что один прецедент использует поведение, определённое в другом, но только при соблюдении определённых условий?
Выберите один ответ:

- Расширение (extend relationship)
- Зависимость (dependency relationship)
- Обобщение (generalization relationship)
- Ассоциация (association relationship)
- Включение (include relationship)

9. Прецедент, изображённый на диаграмме под номером 2 выполняется всегда.



Выберите один ответ:

- Верно
- Неверно

10. Как называется в UML группировка элементов модели, которые специфицируют простейшее поведение физической системы?
Выберите один ответ:

- подсистема (subsystem)
- класс (class)
- подпакет (subpackage)
- подкласс (subclass)
- пакет (package)

4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Практические занятия проводятся в компьютерных классах. Цель практических занятий — закрепление теоретических основ дисциплины, излагаемых в лекционном курсе, а также формирование навыков, умений и соответствующих компетенций у слушателей.

Текущий контроль осуществляется при выполнении практических работ (которые выполняются самостоятельно) — по конечному результату, с учётом вышеописанных показателей, на основе табл. 3. Результат выполнения каждой работы оценивается по 100-балльной системе.

Контроль посещения лекции осуществляется во время теоретических занятий. В случае пропуска лекции слушатель может восстановить (прочитать и написать конспект) и предъявить лекционный материал (возможно, ответив на несколько вопросов).

Промежуточная аттестация осуществляется в виде зачёта. Для подготовки к зачёту даётся ещё 36 ч. самостоятельной работы.

Для расчёта итогового балла B применяется следующая формула:

$$B = T_{\text{итог.}} + D + 0{,}7\frac{\sum\limits_{i=1}^{N} w_i l_i}{N} + a_i, \tag{1}$$

где $T_{\text{итог.}} \leqslant 20$ — балл за итоговый тест; D — балл за прохождение дистанционного курса (при предъявлении сертификата $D = 10$); $l_i \leqslant 100$ — балл за i-ю выполненную слушателем работу;

w_i — вес за i-ю работу (0,5 или 1); N — полное число практических работ в дисциплине; a_i — дополнительный (бонусный) балл, который слушатель может получить за активность (не более, чем 1 балл за одну лекцию).

Проведение зачёта возможно в виде тестирования, оценка за тест выражается в 100-балльной шкале. Тест состоит из 10 вопросов по изученному теоретическому материалу, ограничение по времени: 10 мин, метод оценивания: Последняя попытка. Слушателю разрешено сделать 3 попытки. Слушатель должен ответить на вопросы теста самостоятельно, без использования интернет-источников, учебников и иных пособий, за исключением личного конспекта слушателя. Личный конспект слушателя (допускается, как рукописный, так и электронный вариант) должен быть предъявлен преподавателю (для проверки подлинности авторства) перед проведением тестирования. Тестирование проводится в ЭУМК на образовательном портале АлтГУ. Оценка выставляется автоматически по окончанию теста или отведённого времени.

Окончательная оценка выводится из итогового балла на основании табл. 2.