Федеральное государственное бюджетное образовательное учреждение высшего

образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа № 6

**Модель анализа**

Вариант 9

Выполнил: Иванов В.С.

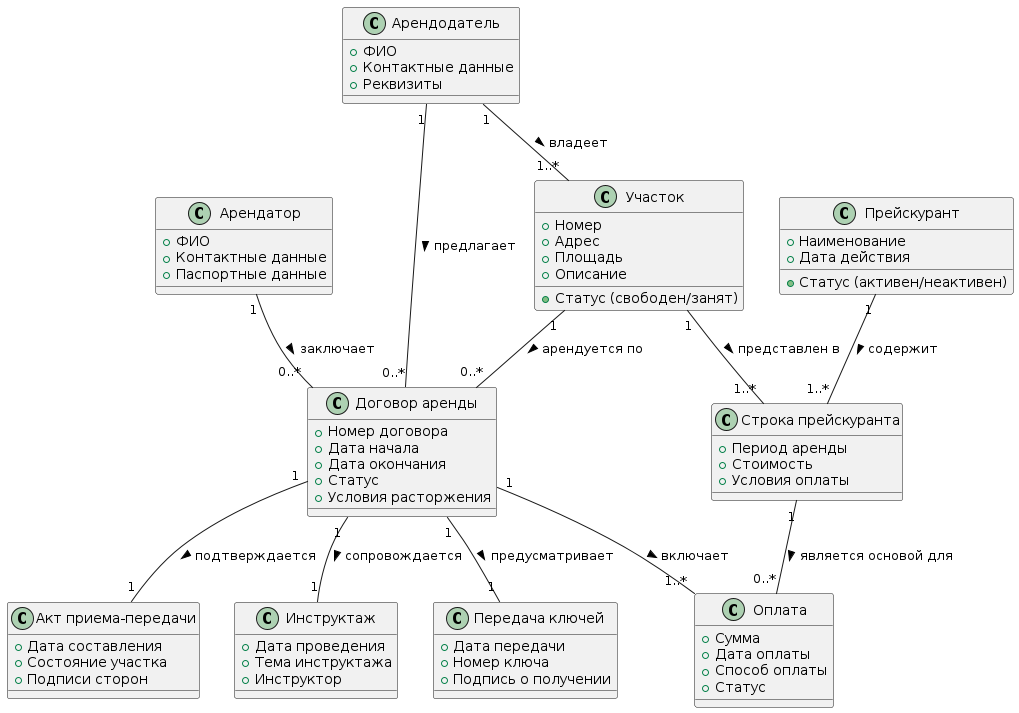
студент группы ИВТ-41-22

Проверил: кандидат технических наук

Ржавин Вячеслав Валентинович

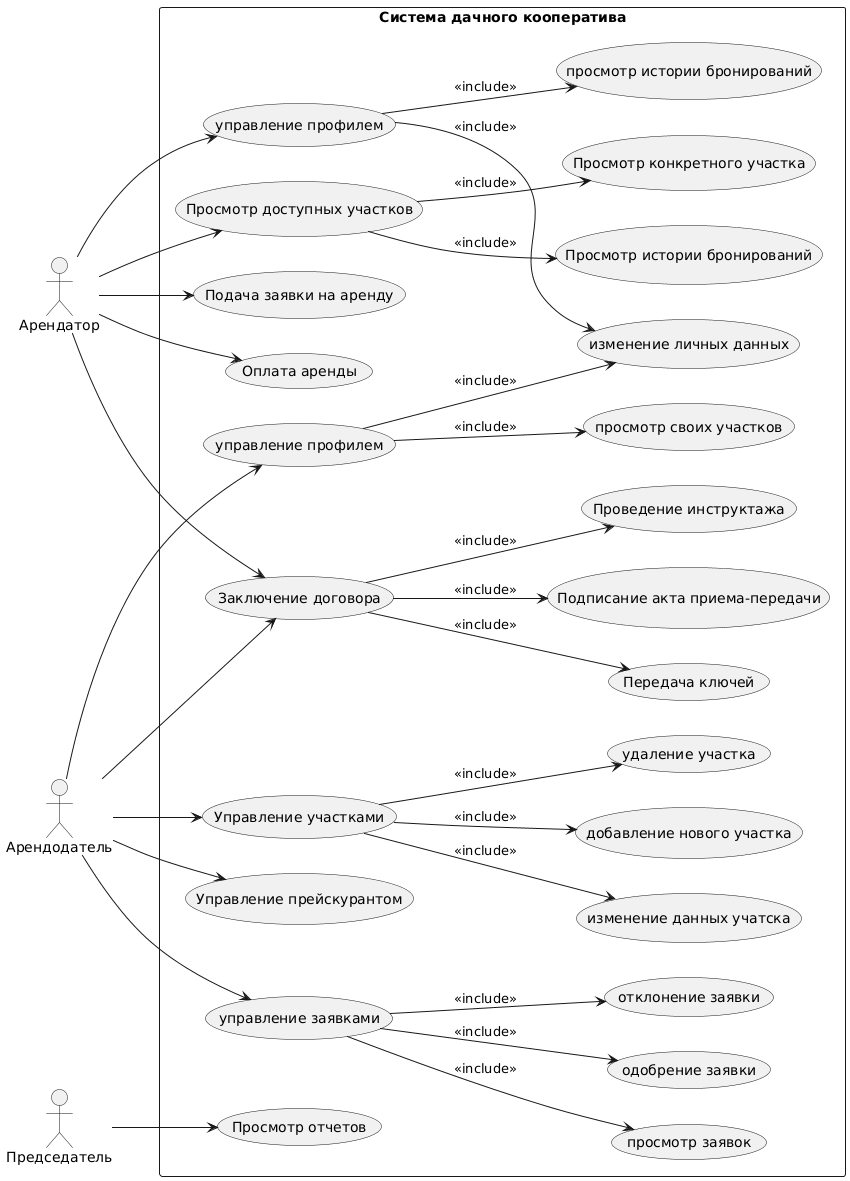
Чебоксары, 2025

Модель предметной области

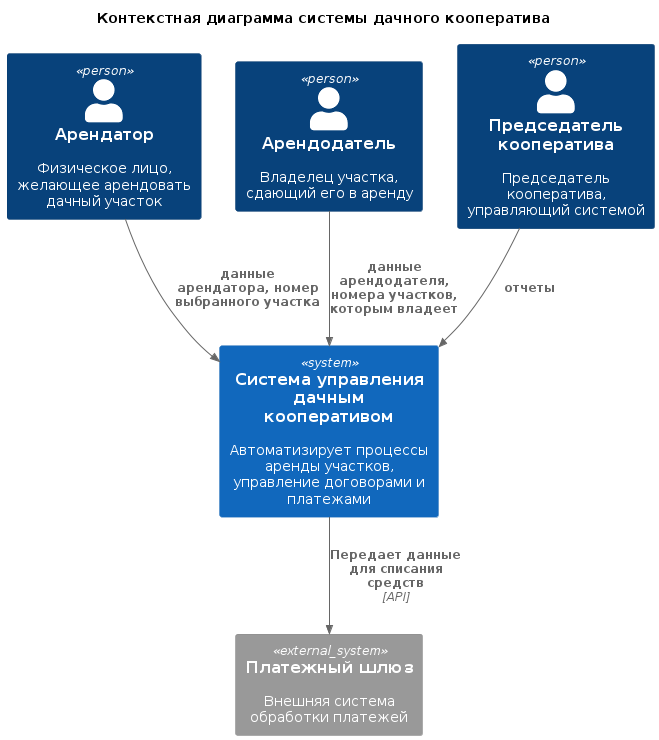


Строка прейс -> композиция

Диаграмма вариантов использования



Контекстная диаграмма (C4)



Сценарии вариантов использования

Основной поток

|  |
| --- |
| Вариант использования : заключение договора аренды |
| ID: UC-014 |
| Краткое описание: процесс оформления договора аренды участка между арендатором и арендодателем |
| Главное действующее лицо: арендатор |
| Второстепенные действующие лица: нет |
| Предусловия:   1. Арендатор авторизован в системе 2. Участок доступен для аренды на выбранные даты 3. Заявка на аренду подтверждена арендодателем 4. Прейскурант актуален и содержит стоимость аренды |
| Основной поток:   1. Арендодатель просматривает заявку 2. Арендодатель принимает заявку на аренду, генерируется шаблон договора 3. Арендатор смотрит договор и подписывает его 4. Система регистрирует договор, присваивает уникальный номер 5. Система сохраняет подписанный договор в базе данных 6. Система отправляет копии договора обеим сторонам 7. Статус договора изменяется на "Активен" |
| Постусловия:   1. Договор аренды зарегистрирован в системе 2. Участок помечен как "Занят" на период аренды 3. Обе стороны имеют доступ к электронной копии договора |
| Альтернативные потоки: нет |

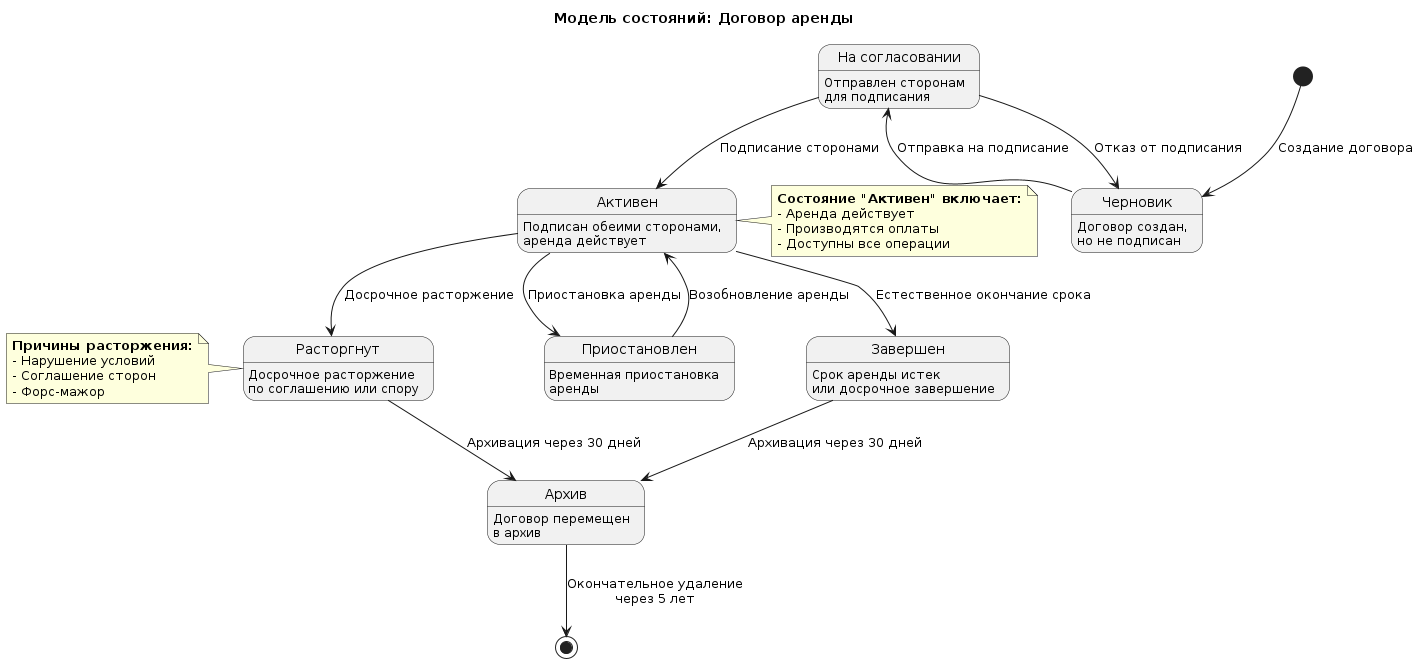
Альтернативные потоки

|  |
| --- |
| Вариант использования : заключение договора аренды |
| ID: UC-014 |
| Краткое описание: процесс оформления договора аренды участка между арендатором и арендодателем |
| Главное действующее лицо: арендатор |
| Второстепенные действующие лица: нет |
| Предусловия:   1. Арендатор авторизован в системе 2. Участок доступен для аренды на выбранные даты 3. Заявка на аренду подтверждена арендодателем 4. Прейскурант актуален и содержит стоимость аренды |
| Альтернативные потоки:  А1: отказ от арендодателя  Точка входа 1 шаг основного потока   1. Арендодатель проверяет условия договора 2. Арендодатель обнаруживает конфликт расписания или иные причины для отклонения 3. Арендодатель отклоняет заявку через интерфейс системы 4. Система фиксирует причину отклонения и время операции 5. Система автоматически освобождает забронированные даты участка 6. Система изменяет статус участка на "Доступен" 7. Система отправляет арендатору уведомление об отклонении заявки   А2: ошибка генерации договора  Точка входа 2 шаг основного потока   1. Система пытается сгенерировать договор аренды 2. Возникает ошибка генерации (отсутствует шаблон, некорректные данные) 3. Система сохраняет лог ошибки для технической поддержки 4. Система отображает арендатору сообщение о временной технической проблеме 5. Система предлагает повторить попытку через 15 минут   А3: отказ арендатора от аренды  Точка входа 3 шаг основного потока   1. Арендатор просматривает подготовленный договор 2. Арендатор обнаруживает несоответствие условий или принимает решение об отказе 3. Арендатор отказывается от подписания договора 4. Система запрашивает подтверждение отказа 5. Система уведомляет арендодателя об отказе арендатора 6. Система освобождает даты аренды |
| Постусловия: нет |

Поток с включением

|  |
| --- |
| Вариант использования : проведение инструктажа |
| ID: UC6 |
| Краткое описание:ознакомление арендатора с правилами и особенностями участка |
| Главное действующее лицо: арендодатель |
| Второстепенные действующие лица: арендатор |
| Предусловия:   1. Акт приема передачи подписан |
| Основной поток:   1. Арендодатель проводит инструктаж по правилам пользования участком 2. Арендодатель объясняет правила пользования общими зонами кооператива 3. Арендодатель знакомит с правилами противопожарной безопасности 4. Арендатор задает вопросы при необходимости 5. Арендатор подтверждает ознакомление с правилами в системе 6. Система фиксирует дату и время проведения инструктажа 7. Система сохраняет запись о проведении инструктажа |
| Постусловия:   1. Инструктаж проведен и зафиксирован в системе 2. Арендатор ознакомлен с правилами пользования участком |
| Альтернативные потоки:  А1: арендатор не понял инструктаж   1. Повторное проведение инструктажа |

Модель состояний предметной области



**Ответы на вопросы по UML**

1. Назовите типы сущностей в UML.

Представь, что ты строишь дом из конструктора. В UML есть четыре основных типа «деталей»:

\* Структурные — это как кирпичи и блоки (стены, окна). Из них собирается каркас.

\* Поведенческие — это как инструкция, что делать с домом (включить свет, открыть дверь).

\* Группирующие — это как коробки, в которые ты складываешь детали конструктора, чтобы не потерялись.

\* Аннотационные — это как стикеры с надписями «это кухня», которые ты клеишь на стены для пояснений.

2. Что относится к структурным сущностям?

Это основные «кирпичики» нашей системы. Самые главные:

\* Класс — шаблон объекта (например, шаблон «Студент»).

\* Интерфейс — список правил, что класс \*должен\* уметь делать.

\* Use Case (Прецедент) — описание одной цели пользователя (например, «Оплатить заказ»).

\* Компонент — готовый кусок программы (например, файл calculator.dll).

3. Что относится к поведенческим сущностям?

Это то, что описывает, как система \*ведет себя\*, что происходит \*во времени\*.

\* Взаимодействие — кто, кому и какое сообщение отправляет (как переписка в мессенджере).

\* Состояние — как объект меняется после каких-то событий (например, статус заказа: «создан» -> «оплачен» -> «доставлен»).

\* Деятельность — последовательность шагов для достижения цели (как рецепт приготовления блюда).

4. Назовите семь наиболее важных свойств классификаторов.

Классификатор (например, класс «Автомобиль») может иметь:

1. Имя (например, Car).
2. Атрибуты — \*данные\* (цвет, модель, год выпуска).
3. Операции — \*действия\*, которые он может выполнять (завести(), затормозить()).
4. Обязанности — короткое описание, за что он отвечает.
5. Связи с другими классами (например, «Автомобиль» \*имеет\* «Двигатель»).
6. Видимость — что видно другим классам, а что скрыто (кары public и private).
7. Множественность — сколько таких объектов может быть (например, у одного «Владельца» может быть от 1 до 5 «Автомобилей»).

5. Что в UML относится к классификаторам?

Классификатор — это просто умное слово для «шаблона» или «чертежа», по которому создаются объекты. К ним относятся:

\* Класс

\* Интерфейс

\* Компонент

\* Use Case (Прецедент)

\* Актер (действующее лицо)

6. Что такое абстрактный классификатор?

Это «недокласс». Он описывает общие черты, но сам по себе не может использоваться. Его имя пишут \*курсивом\*.

\* Пример: Есть абстрактный класс \*ТранспортноеСредство\* с методом двигаться(). Мы не можем создать просто «транспортное средство», но можем создать по его чертежам \*конкретные\* классы «Автомобиль» или «Самолет», которые уже реализуют \*как\* именно они двигаются.

7. Что такое классификатор?

Смотри ответ №5. Это «чертеж» или «форма» для печенья. А само печенье — это уже объект.

8. Что такое дескриптор?

Это и есть сам классификатор («чертеж»). Он один, он описывает общие для всех правила.

9. Что такое литерал? Чем отличается от дескриптора?

Литерал — это конкретный объект, созданный по чертежу.

\* Дескриптор (чертеж): Класс Student { String name; }.

\* Литерал (объект): student1 = new Student(); student1.name = "Мария";.

10. Дайте понятия состояния, деятельности, действия.

\* Действие — это один маленький шаг (например, «нажать кнопку»).

\* Деятельность — это целый процесс из нескольких действий (например, «Приготовить завтрак»: разбить яйца, взболтать, пожарить).

\* Состояние — это положение объекта \*между\* действиями (например, после действия «включить духовку» её состояние меняется на «разогрета»).

11. Является ли сущность Use Case классификатором? Обоснуйте ответ.

Да. Use Case «Оплатить заказ» — это \*шаблон\* (чертеж) для всех возможных сценариев оплаты: картой, наличными, через PayPal. Конкретная оплата картой — это уже \*объект\* (литерал), созданный по этому шаблону.

12. Что относится к группирующим сущностям?

В основном, это Пакет (Папка). Просто папка, в которую ты складываешь связанные классы и диаграммы, чтобы навести порядок в проекте.

13. Что такое программный интерфейс? Что такое сигнатура?

\* Интерфейс — это \*контракт\* или \*должностная инструкция\*. В нем написано что нужно сделать (например, рассчитатьЦену()), но не написано как это делать.

\* Сигнатура — это \*паспорт\* операции. Её полное описание: имя (рассчитатьЦену), параметры (скидка: int) и что возвращает (float).

14. Дайте определение компонента UML.

Компонент — это готовый, самостоятельный кусок программы, который можно взять и использовать в другом месте. Как деталь Лего. Например, библиотека для работы с базой данных или модуль расчета доставки.

15. Что такое артефакт? Приведите примеры.

Артефакт — это любой файл, который получается в результате работы программы или который нужен для её работы.

\* Примеры: program.exe, myapp.jar, index.html, database.sql, config.xml, отчет.pdf.

16-17. Какие типы отношений определены в UML?

Это то, как наши «кирпичики» связаны друг с другом. Основные типы связей:

1. Ассоциация (сплошная линия): Просто факт связи. «Преподаватель — \*ведет\* — Курс».

\* Агрегация (линия с пустым ромбом): Связь «часть-целое», где часть может жить без целого. «Компьютер — \*содержит\* — Монитор» (монитор можно отключить и подключить к другому компьютеру).

\* Композиция (линия с закрашенным ромбом): Более жесткая связь. Часть не может жить без целого. «Дом — \*имеет\* — Комнату» ( если разрушить дом, комнаты не станет).

2. Зависимость (пунктир со стрелкой): Один класс \*использует\* другой временно. «Класс Заказ использует класс Logger для записи в журнал».

3. Обобщение (сплошная линия с треугольной стрелкой): Это наследование. «Класс Студент наследует классу Человек».

4. Реализация (пунктир с треугольной стрелкой): Класс \*реализует\* интерфейс. «Класс MySQLDatabase реализует интерфейс Database».

18. Назовите основные типы диаграмм.

Их много, но все делятся на два типа:

\* Структурные диаграммы (как статичная фотография):

\* Диаграмма классов — самая главная. Показывает из каких классов состоит система и как они связаны.

\* Диаграмма компонентов — показывает из каких готовых кусков (библиотек) собрана программа.

\* Диаграмма развертывания — показывает на каком компьютере какая программа работает.

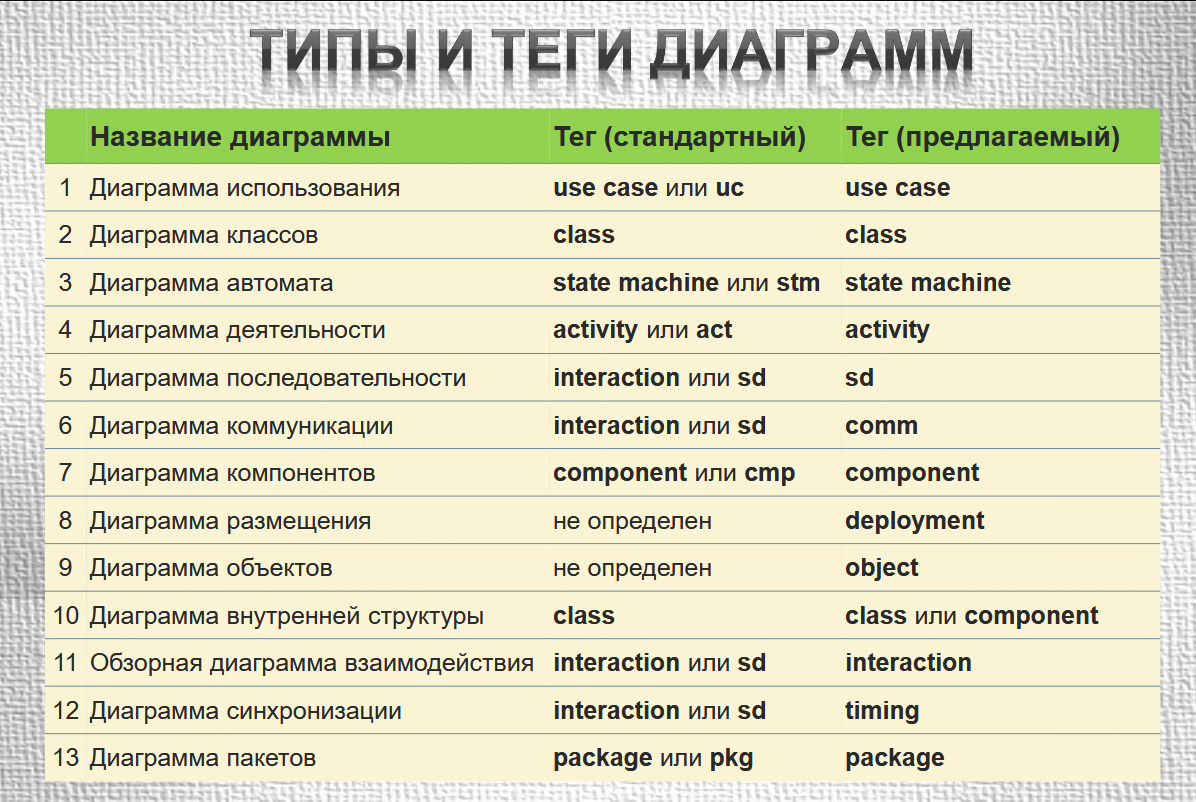
\* Поведенческие диаграммы (как видео):

\* Диаграмма прецедентов — показывает, \*что\* система может делать для пользователя.

\* Диаграмма последовательностей — показывает \*как\* объекты обмениваются сообщениями \*вовремени\*.

\* Диаграмма состояний — показывает, как меняется состояние \*одного\* объекта.

\* Диаграмма деятельностей — похожа на блок-схему, показывает поток работ.



19. Какие представление включает в себя модель «4+1»?

Это способ посмотреть на одну и ту же систему с разных сторон, чтобы понять её полностью.

1. Логическое (логика программы): Диаграммы классов.

2. Реализации (как всё собрано): Диаграммы компонентов.

3. Процессов (как всё работает одновременно): Диаграммы взаимодействия.

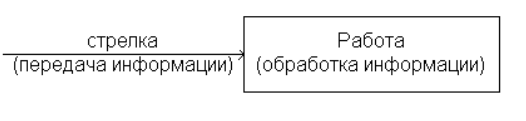
4. Развертывания (где всё работает): Диаграммы развертывания.

+1. Use Case (что система делает для пользователя): Эта view связывает все остальные, потому что всё начинается с потребностей пользователя.

**Контрольные вопросы**

1. Что такое модель домена и с какой целью она создаётся?

Часто используемые комбинации свойств можно именовать (дать имя). Такая комбинация свойств называется доменом, может наследоваться. Например, можно определить домен DATA для отображения всех колонок с датами для отображения в одном стиле. Домен DATA родителя ребенка наследует все атрибуты домена и может вносить свои. Связывание данных - входные/выходные данные - обработка данных



1. Поясните сценарий использования. Где и кем используется?

Сценарий представляет собой последовательность шагов, описывающих взаимодействие между пользователем и системой.

ВИ представляет собой множество сценариев, объединенных вместе некоторой общей целью пользователя.

Цель варианта использования заключается в том, чтобы определить законченный аспект или фрагмент поведения некоторой сущности без раскрытия внутренней структуры этой сущности.

1. Какие виды отношений используются в Use Case?

|  |  |
| --- | --- |
| Ассоциация | Указывает, что субъект принимает участие в варианте использования. |
| Включение | Включающий вариант использования вызывает включенный. Включение используется, чтобы показать, как разбить вариант использования на несколько более мелких шагов. Включенный вариант использования находится на окончании с наконечником стрелки. |
| Расширение | Расширяющий вариант использования добавляет цели и шаги в расширяемый вариант использования. Расширения работают только при определенных условиях. Расширенный вариант использования находится на окончании с наконечником стрелки. |
| Наследование | Устанавливает отношение между специализированным и обобщенным элементом. Обобщенный элемент находится на окончании с наконечником стрелки. Специализированный вариант использования наследует цели и субъекты своего обобщения и может добавлять более конкретные цели и шаги для их достижения. Специализированный субъект наследует варианты использования, атрибуты и ассоциации своего обобщения и может добавлять дополнительные объекты. |
| Зависимость | Указывает, что конструкция источника зависит от конструкции целевого объекта. |

1. Что такое точка расширения (extension point), условие активации (activationcondition)иветка расширения (extension branch)?

Точка расширения (Extension Point)

Что это: Конкретное место в основном Use Case, где может быть подключено дополнительное поведение из расширяющего Use Case.

Аналогия: Это как "розетка" в основном сценарии, в которую можно "включить" дополнительную функциональность.

Условие активации (Activation Condition)

Что это: Булево выражение, которое определяет, когда именно должно сработать расширение.

Аналогия: Это "выключатель", который решает, будет ли использована "розетка".

Формат: Логическое условие, которое должно быть истинно для активации расширения.

Ветка расширения (Extension Branch)

Что это: Последовательность шагов из расширяющего Use Case, которая выполняется когда условие активации истинно.

Аналогия: Это "удлинитель с прибором", который включается в розетку когда нужно.

Содержит: Конкретные действия, которые добавляются к основному потоку.

1. Какие другие элементы UML допустимо использовать на диаграмме UC, кроме основных?

Пакеты, артефакты, комментарии, зависимости, интерфейсы

1. Назовите основные типы актёров.

Пользователи, программные системы, аппаратные средства, время.

1. Что такое абстрактное действующее лицо?