АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ
ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ
«Международный Колледж Бизнеса и Дизайна»
(АНО ПОО «Международный Колледж Бизнеса и Дизайна»)

УТВЕРЖДАЮ
Директор АНО ПОО «МКБиД»
_____ Н.Н.Репин
«28» августа 2023 г.

КОНСПЕКТ ЛЕКЦИЙ

по учебной дисциплине МДК.03.01 Моделирование и анализ программного обеспечения

программы подготовки специалистов среднего звена
по специальности: 09.02.07 «Информационные системы и программирование»

2023 год

ЛЕКЦИЯ 1

Тема: Понятие моделирования. Способы представления моделей.

Модель — аналог, прототип, **шаблон**, образец, используемый вместо оригинала для решения задач (получения ответов на вопросы). Модель строится на основании ограниченного множества известных нам данных (свойств, поведений) об оригинале. *Построение моделей и использование моделей (решение на них задач) производится с целью:*

- получения неизвестных ранее данных, предсказания новых свойств и будущих поведений,
- извлечения пользы при реализации решений,
- систематизации (обобщения) известных данных.

Моделирование – способ, **процесс** замещения оригинала его аналогом (моделью) с последующим изучением свойств и поведения оригинала на модели.

Процесс моделирования состоит из:

- ***формализации*** (проектирование и настройка модели, систем моделей и моделей систем),
- ***собственно моделирования*** (постановка различных задач и решение их на модели),
- ***интерпретации результатов*** моделирования, комплексирования с уже имеющимися реальными системами.

Модель вместо исходного объекта используется в случаях, когда эксперимент опасен, дорог, происходит в неудобном масштабе пространства и времени невозможен, неповторим, ненагляден .

Использование моделирования в случаях:

- «**эксперимент опасен**» — при деятельности в агрессивной среде вместо человека лучше использовать его макет; примером может служить луноход;
- «**дорог**» — прежде чем использовать идею в реальной экономике страны, лучше опробовать её на математической или имитационной модели экономики, просчитав на ней все «за» и «против» и получив представление о возможных последствиях;
- «**долговременен**» — изучить коррозию — процесс, происходящий десятилетия, — выгоднее и быстрее на модели;
- «**кратковременен**» — изучать детали протекания процесса обработки металлов взрывом лучше на модели, поскольку такой процесс скоротечен во времени;

- «**протяжен в пространстве**» — для изучения космогонических процессов удобны математические модели, поскольку реальные полёты к звёздам (пока) невозможны;
- «**микроскопичен**» — для изучения взаимодействия атомов удобно воспользоваться их моделью;
- «**невозможен**» — часто человек имеет дело с ситуацией, когда объекта нет, он ещё только проектируется. При проектировании важно не только представить себе будущий объект, но и испытать его виртуальный аналог до того, как дефекты проектирования проявятся в оригинале.

Важно: моделирование теснейшим образом связано с проектированием. Обычно сначала проектируют систему, потом её испытывают, потом снова корректируют проект и снова испытывают, и так до тех пор, пока проект не станет удовлетворять предъявляемым к нему требованиям.

Процесс «проектирование-моделирование» цикличен. При этом цикл имеет вид спирали — с каждым повтором проект становится все лучше, так как модель становится все более детальной, а уровень описания точнее;

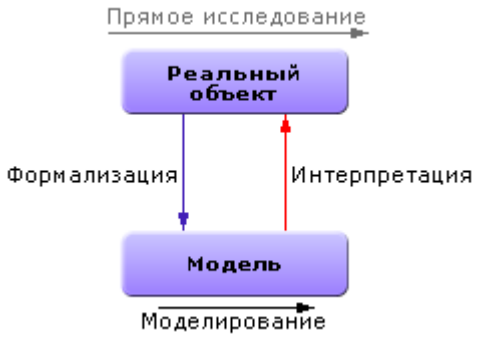
- «**неповторим**» — это достаточно редкий случай, когда эксперимент повторить нельзя; в такой ситуации модель — единственный способ изучения таких явлений. Пример — исторические процессы, — ведь повернуть историю вспять невозможно;
- «**ненагляден**» — модель позволяет заглянуть в детали процесса, в его промежуточные стадии; при построении модели исследователь как бы вынужден описать причинно-следственные связи, позволяющие понять все в единстве, системе. Построение модели дисциплинирует мышление.

Важно: модель играет системообразующую и смыслообразующую роль в научном познании, позволяет понять явление, структуру изучаемого объекта. Не построив модель, вряд ли удастся понять логику действия системы.

Это означает, что модель позволяет разложить систему на элементы, связи, механизмы, требует объяснить действие системы, определить причины явлений, характер взаимодействия составляющих.

Процесс моделирования *есть процесс перехода из реальной области в виртуальную* (модельную) *посредством формализации*, далее происходит изучение модели (собственно моделирование) и, наконец, интерпретация результатов как обратный переход из виртуальной

области в реальную. Этот путь заменяет прямое исследование объекта в реальной области, то есть лобовое или интуитивное решение задачи.



Итак, в самом простом случае **технология моделирования подразумевает 3 этапа:**

1.формализация

2.собственно моделирование

3.интерпретация (**рис. 1.1**).

Рис. 1.1. Процесс моделирования (базовый вариант)

Если требуется уточнение, эти этапы повторяются вновь и вновь: формализация (проектирование), моделирование, интерпретация. *Спираль!* Вверх по кругу.

ЛЕКЦИЯ 2

Тема: Базовый комплекс моделирования. Основные подсистемы при проектировании моделей. Виды моделей.

Из прошлого занятия следует, что моделей может быть несколько: *приближенная, более точная,. Модели как бы образуют ряд. Двигаясь от варианта к варианту*, *исследователь совершенствует модель*. *Для построения* и совершенствования моделей *необходима средства* отслеживания версий и так далее, то есть моделирование требует *инструмента* и опирается на *технологию*.

Инструмент — типовое средство, позволяющее достичь оригинальный результат и обеспечивающее сокращение затрат на выполнение промежуточных операций (стандартные библиотеки, мастера, линейки, резинки).

Это то с помощью чего мы будем моделировать

Технология — набор стандартных способов, приёмов, методов, позволяющий достичь результата гарантированного качества с помощью указанных инструментов за заранее известное время при заданных затратах, но при соблюдении пользователем объявленных требований и порядка.

Это то как мы будем моделировать ,грубо говоря как наклонить сколько раз повторить удар

Среда — совокупность рабочего пространства и инструментов на нем, поддерживающая хранение и изменение, преемственность проектов и интерпретирующая свойства объектов и систем из них.

Это то, с помощью чего мы будем производить эти удары, к примеру молоток

Иногда модели пишут на языках программирования, но это долгий и дорогой процесс. Для моделирования можно использовать математические пакеты, но, как показывает опыт, в них обычно не хватает многих инженерных инструментов. Оптимальным является использование среды моделирования

Любое моделирование состоит из алгоритма действий.

Алгоритм-это любая последовательность действий, которая приведёт к определенному результату.

Какова разница между алгоритмом и моделью?

Алгоритм — это процесс решения задачи путём реализации последовательности шагов, тогда как **модель** — совокупность потенциальных свойств объекта.

Если к *модели* поставить *вопрос* и добавить *дополнительные условия* в виде исходных данных (связь с другими объектами, начальные условия, ограничения), то она может быть разрешена исследователем относительно неизвестных.

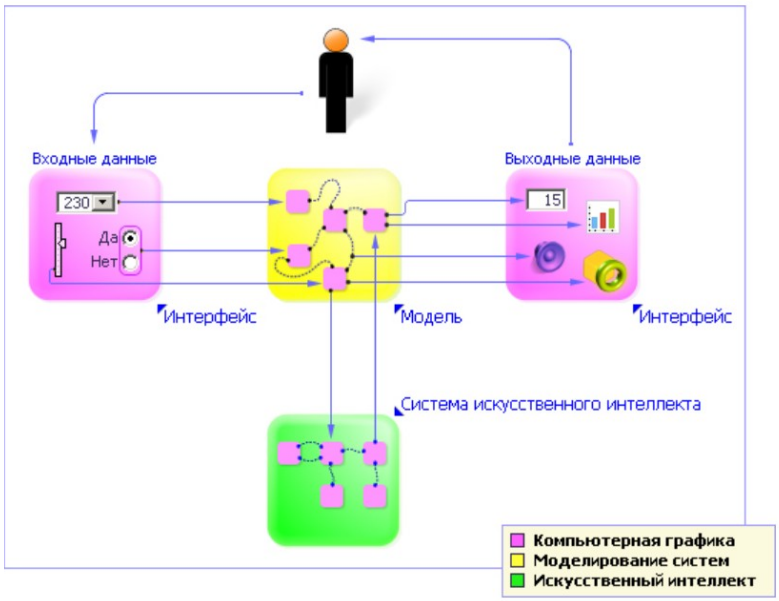
модель + вопрос + дополнительные условия = задача

Процесс решения задачи может быть представлен алгоритмом.

Вообще примеры алгоритмов в природе неизвестны, они суть порождение человеческого мозга, способного к установлению плана.

Собственно алгоритм — это и есть план, развёрнутый в последовательность действий. Следует различать поведение объектов, связанное с естественными причинами, и промысел разума, управляющий ходом движения, предсказывающий результат на основе знания и выбирающий целесообразный вариант поведения

Основные подсистемы при проектировании комплексных моделей



Модели могут быть:

1. **феноменологические и абстрактные;**

привязаны к конкретному явлению. Абстрактная модель воспроизводит систему с точки зрения её внутреннего устройства, копирует её более точно. У неё больше возможностей, шире класс решаемых задач.

2. **активные и пассивные;**

Активные модели взаимодействуют с пользователем; могут не только, как пассивные, выдавать ответы на вопросы пользователя, когда тот об этом попросит, но и сами активируют диалог, меняют его линию, имеют собственные цели. Все это происходит за счёт того, что активные модели могут самоизменяться

3. **статические и динамические;**

Статические модели описывают явления без развития. Динамические модели прослеживают поведение систем, поэтому используют в своей записи, например, дифференциальные уравнения, производные от времени.

4. **дискретные и непрерывные;**

Дискретные модели *изменяют состояние переменных скачком*, потому что не имеют детального описания связи причин и следствий, часть процесса скрыта от исследователя. *Непрерывные модели более точны, содержат в себе информацию о деталях перехода*.

5. **детерминированные и стохастические;**

Если *следствие точно определено причиной*, то модель представляет процесс *детерминировано*. Если из-за неизученности деталей *не удаётся описать точно связь причин и следствий*, а возможно только описание в целом, *статистически*,то модель строится с использованием понятия вероятности.

6. **функциональные и объектные.**

Если описание идёт с точки зрения поведения, то модель построена по *функциональному* признаку. *Если описание каждого объекта отделено от описания другого объекта*, если описываются свойства объекта, из которых вытекает его поведение, то *модель является объектно-ориентированной*.

Каждый подход имеет свои достоинства и недостатки. Разные математические аппараты имеют разные возможности (мощность) для решения задач, разные потребности в вычислительных ресурсах. Один и тот же объект может быть описан различными способами. Инженер должен грамотно применять то или иное представление, исходя из текущих условий и стоящей перед ним проблемы.

Приведённая выше классификация является идеальной. Модели сложных систем обычно имеют комплексный вид, используют в своём составе сразу несколько представлений.

ЛЕКЦИЯ 3

Тема: Назначение метода моделирования в процессе принятия и реализации управленческих решений. Классификация моделей.

Модели обеспечивают исследователям возможность уточнения характеристик и свойств изучаемого явления, представляя это явление в упрощенном и наглядном варианте для изучения.

Для того как чтобы приступить к проектированию моделей, вы должны знать, как себя поведет та или иная модель в различных ситуациях, каков исход ее будет при различных условиях

Основное назначение применения моделей в процессе разработки и реализации управленческих решений *состоит в возможности проведения активных экспериментов*, которые не могут быть проведены с определенным объектом, в отношении которого применяются управленческое воздействие.

Моделирование применяется различными науками, не является исключением государственное и муниципальное управление. Существует несколько определений понятия «модель» и «моделирование».

Модель представляет собой:

1. определенный *образ объекта*, явления или процесса, который *призван заменять реальность* и представление о ней;

может ускорить или замедлить время, наложить определенные условия на объект

2. способ и средства отображения свойств объекта в соотношении с оригиналом;

с помощью чего лучше всего продемонстрировать наглядный и понятный результат

3. имитацию свойства объекта за счет других явлений и предметов.

Более дешёвая вариация воспроизведения результата за счет моделировании шаблона

Модель в системе разработки и реализации управленческих решений обладают следующими свойствами:

1. Находится в состоянии подобности по отношению к исследуемому объекту;

Грубо говоря, ее маленькая копия, ее дешевле и проще и менее затратно реализовать

2. Отличается простотой по сравнению с исследуемым объектом, что позволяет более подробно осуществить ее изучение;

Модели можно постепенно задавать дополнительные стрессовые условия, тем самым постепенно усложнять и видеть этапы.

3. Обладает широтой и глубиной описания, которые признаются достаточными для исследования объекта;

Много технической и математической документации описывающей, каждое поведение объекта

4. Отвечает дополнительным ограничениям по времени, требуемого для принятия управленческого решения.

Тестируем на определенных временных промежутках, для определения и записи резльтатов

Модель находится в тесной взаимосвязи с решаемой проблемой, *давая представление о проблемной ситуации и формируя возможности для моделирования стратегических альтернатив и последствий.*

Моделирование представляет собой оперирование объектом практического или теоретического характера, позволяющего заменять изучаемый предмет искусственным или естественным аналогом для исследования. **В основе моделирования находятся принципы подобия, общности свойств, аналогии, самостоятельности формы.**

Классификация моделей по признаку отображения действительности:

Совокупность моделей, применяемых для разработки и реализации управленческих решений, можно классифицировать на следующие виды по признаку отображения действительности:

1. **физические модели;**
2. **графические модели;**
3. **математические модели,** наиболее часто используемые

Разработка и реализация управленческих решений преимущественно осуществляется с помощью математических моделей.

Процесс принятия управленческих решений также позволяет выделить модели как:

1. **дескриптивная**

2. **описательная** - В основе находятся *эмпирические наблюдения*, данные модели отличаются небольшим количеством элементов и дают объяснения экономическим соотношениям, как в реальном выражении, так и упрощенном. Взаимосвязи элементов такой модели описываются простыми математическими уравнениями.

Недостатком является отсутствие отражения функциональных ограничений и взаимосвязей. Однако описательные модели выступает в качестве основы формирования более сложных моделей.

пример данного вида моделей можно привести модель идеальной конкуренции. Спрос-предложение

3. **аналитическая или нормативная-** обеспечивают выявление наиболее эффективных путей достижения цели. Данные модели выражаются через функциональные уравнения, отражающие взаимосвязь зависимых и независимых переменных. В качестве независимых переменных этих моделей выступают параметры действий, зависимые переменные представляют собой ожидаемые переменные, которые получаются в результате воздействия независимых переменных.

Описательные модели являются выражением целевой функции, описывая определенную технологию или процедуру. *Их назначение состоит в возможности выбора оптимального решения с учетом существующих критериев и ограничений.*

Аналитические модели обеспечивают возможность представления последствий принятых управленческих решений в виде определенных переменных.

Факторы внешней среды оказывает существенное влияние на возможности моделирования. Внешняя среда отражается такими характеристиками как риск, определенность и неопределенность.

Моделирование при принятии и реализации управленческих решений должно учитывать факторы внешней среды, прогнозируя их воздействие на ожидаемые результаты.

Этапы создания моделей при разработке и реализации управленческих решений

При построении модели разработки и реализации управленческих решений можно выделить ряд этапов, к числу которых относятся следующие:

1. **этап постановки задачи**. обеспечивая диагностику проблемы и ее идентификацию;
2. **этап создания модели**. устанавливает такие параметры моделирования как цель, исходная и конечная информация, источники информации;
3. **этап проверка модели**. представляет собой оценку соответствия созданной модели реальной проблеме.

Основу проверки модели на достоверность обеспечивает прошлый опыт относительно состояния объекта, его проблем и последствий в результате осуществления тех или иных управленческих решений;

4. **этап применение модели**. представляет собой вид практической деятельности, подразумевающий использование моделей в конкретной ситуации.
5. **Этап моделирование в системе разработки** моделирование с помощью конкретного ПО

ЛЕКЦИЯ 4-5

Тема: унифицированный язык моделирования UML. Особенности UML. Основные виды диаграмм.

Итак, системы *моделирования процессов разработки строились на качестве выполнения требований к разрабатываемой системе.*

При анализе каждой модели выявлена эффективность только тех, в которых предусмотрено постоянное общение с заказчиком, наличие, качество и полнота документации.

Моделирование может применяться во всех перечисленных моделях разработки(класическая, инкрементная, RAD, XP, спиральная)

Моделирование может производиться и на этапе постановки задачи, как заказчиком, так и разработчиком. Здесь спектр задач моделирования весьма велик.

Например, заказчик предъявил особые требования к дизайну сайта и сделал макеты страниц в системе фотошоп.

Если есть готовая модель web-страниц, то нет необходимости перечислять требования к дизайну страниц в техническом задании, достаточно просто сослаться на модель.

На этапе анализа требований моделирование – хороший способ обезопасить себя как заказчика, ведь некоторые детали можно уточнить у заказчика.

Примером может служить случай, когда дизайн сайта описан в словесной форме и, чтобы уточнить, что именно имел в виду заказчик, можно воспользоваться системой Axure RP Pro 7.0. Также необходимо поступать и с графическим интерфейсом программ, ведь графическая часть плохо описывается словесно. *Если предметом разработки является компьютерная игра, то, возможно необходимы 3D модели элементов игры.*

Моделирование активно применяется на этапе проектирования программного продукта. Простейшим примером моделирования является составление блок-схемы, которая является модель алгоритма решения задачи.

Примером системы для моделирования на этапе проектирования может служить Astah – программа, предназначенная для создания UML диаграмм.

UML (Unified Modeling Language) — унифицированный язык моделирования, активно применяется при объектно-ориентированном анализе. **Цель UML** наглядное проектирование архитектуры программного продукта.

Для чего нужны UML-диаграммы

1. **Для создания «чертежей» программы**, схем, которые показывают, как будет устроено программное обеспечение изнутри, — то есть для проектирования. Это может быть описание связей между компонентами, модулей или сервисов, программных процессов и многого другого.
2. **Для визуализации уже имеющейся программной структуры**. Ряд инструментов позволяет создать UML на основе существующего кода, в таком случае диаграмма сгенерируется автоматически. Это называется реверс-инжинирингом.
3. **Для автоматической генерации кода или технической документации к нему**, так как UML поддерживает возможность создания продукта на основе диаграмм.

автоматически сгенерированный код позже нуждается в доработке. А вот созданная таким образом документация обычно наглядная и понятная.

4. **Для внутренней и внешней коммуникации между сотрудниками, заказчиками** и другими: картинки и диаграммы UML понятнее людям, чем текстовые описания.

Важная особенность UML — этот язык поддерживает объектно-ориентированный подход, где все сущности представлены как объекты с определенными свойствами и методами. В диаграммах UML легко изобразить объекты, связи между ними, наследование и возможности передачи данных от одного объекта к другому.

Как устроена диаграмма UML

Схема UML — концептуальная: это значит, что она оперирует концепциями и связями между ними. Сама диаграмма состоит из фигур, значков, надписей, линий и контуров.

1. **Фигуры обычно обозначают ту или иную концепцию**: например, объект, класс, группу объектов или что-либо еще. Вариантов фигур в языке множество. Внутри одной фигуры могут находиться другие элементы, главное — чтобы они не пересекали границу.
2. **Значки тоже обозначают разные сущности, но отличаются от фигур**: внутрь них нельзя ничего поместить. Это могут быть более мелкие атомизированные структурные единицы, а могут быть служебные сущности, например для описаний.
3. **Надписи могут быть обычными, подчеркнутыми, курсивными**. Они именуют сущности, показывают, что есть что, и могут использоваться для описаний.

4. **Линии могут быть прямыми, ломаными, изогнутыми, направленными и ненаправленными, штриховыми и какими-либо** еще. Обычно они обозначают связи и зависимости сущностей друг от друга.
5. **Контуры — это контейнеры**, внутри которых помещаются концепции и связи между ними.

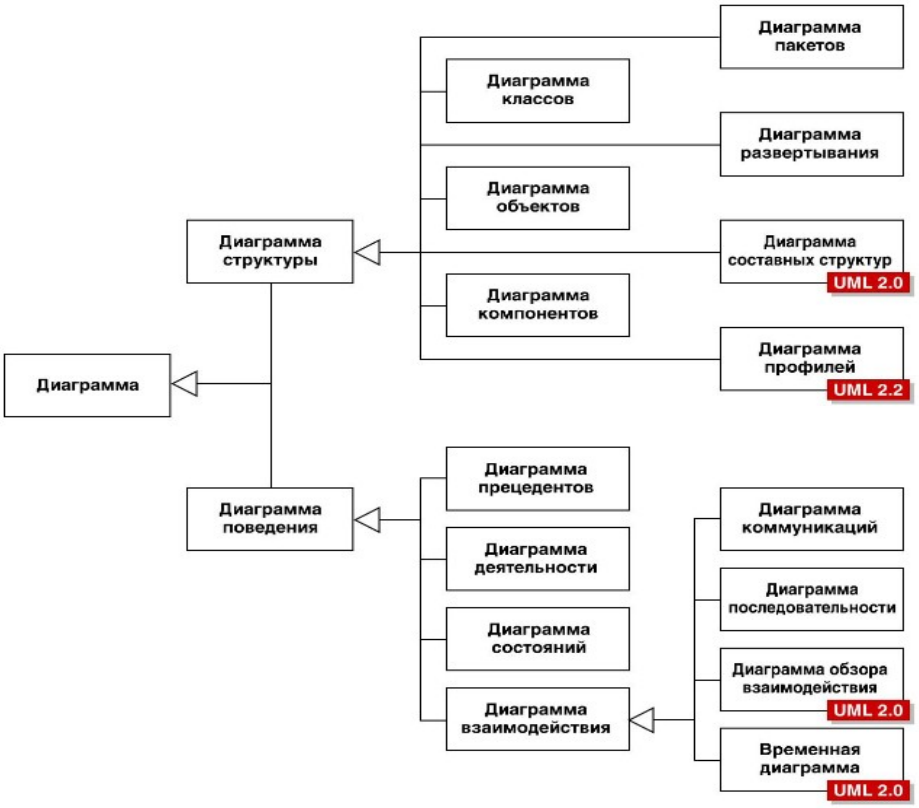


Рис. 1. Виды диаграмм UML

Все виды диаграмм в UML

Среди них выделяют структурные диаграммы, отражающие ***статическую структуру системы:***

1. **диаграммы классов** описывают типы объектов системы (классы), их свойства и статические отношения, которые существуют между ними;
2. **диаграммы объектов** представляют снимок объектов системы в определенный момент времени;
3. **диаграммы компонентов** отображают компоненты системы, различные виды связей между ними и интерфейсы взаимодействия;
4. **диаграммы составных структур** отображают внутреннюю структуру компонентов или подсистем: части, взаимосвязи (коннекторы), интерфейсы и порты:
5. **диаграммы пакетов** позволяют структурировать модель или отобразить структуру системы, указывая разбиение на логические части, содержимое этих частей и взаимосвязи между ними;

6. **диаграммы развертывания** (размещения) представляют структуру аппаратных, коммуникационных средств, а также физическое расположение элементов программного обеспечения (элементов конфигурации) на оборудовании, информационные пути и протоколы взаимодействия между элементами;

7. **диаграммы профилей** описывают динамическую метамодель системы спомошью стереотипов, которые специфицируют изменение свойств объектов (классов, компонентов) в случае применения к системе различных профилей, модифицирующих ее поведение

и поведенческие диаграммы:

8. **диаграммы прецедентов** отображают функциональное назначение и границы системы или ее части в виде набора сценариев использования системы (прецедентов) акторами (заинтересованными лицами или внешними системами);

9. **диаграммы деятельности** описывают логику и процесс выполнения некоторой деятельности, включая основные шаги, переходы между ними и потоки управления;

10. **диаграммы состояний** (конечных автоматов) отображают все возможные состояния системы или отдельных объектов на протяжении их жизненного цикла, события и переходы между состояниями, а также структуру сложных и параллельных состояний;

Распространенные виды диаграмм:

1. **диаграмма вариантов использования** (use case diagram) состоит из эктеров и претендентов. Эктер отражает объект, а претендент – действие.

Диаграмма хорошо подходит для моделирования практически любых процессов.

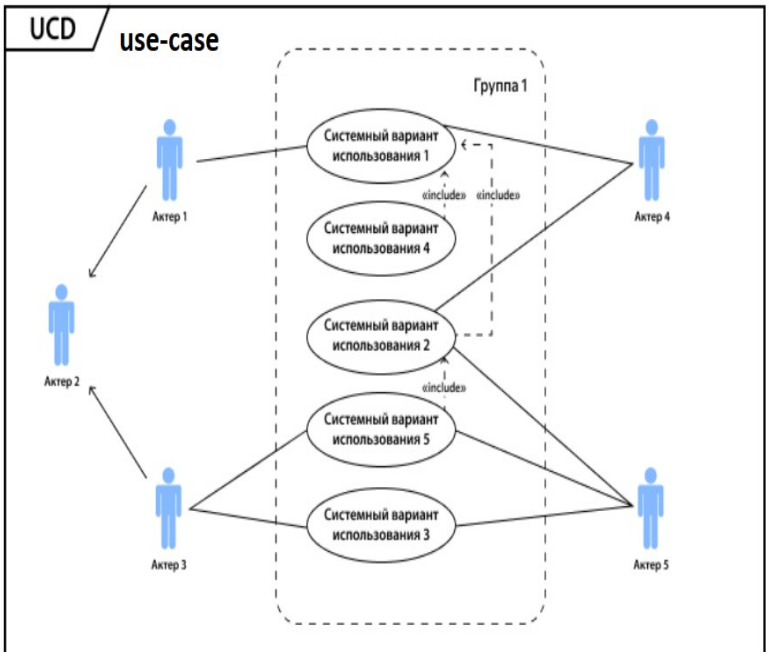
2. **Диаграмма последовательностей-** используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это средство документирования проекта с точки зрения сценариев использования!

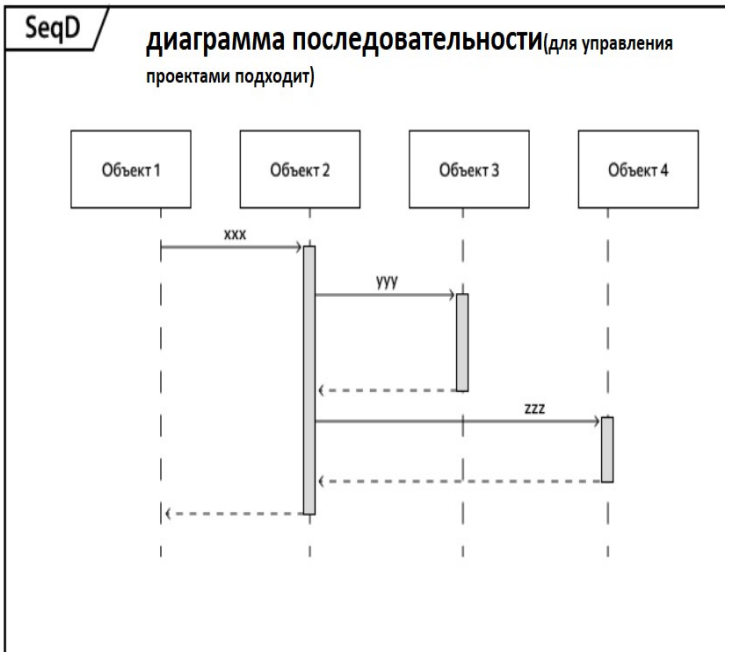
Диаграммы последовательностей обычно содержат *объекты*, которые *взаимодействуют в рамках сценария, сообщения*, которыми они обмениваются, и *возвращаемые результаты*, связанные с сообщениями.

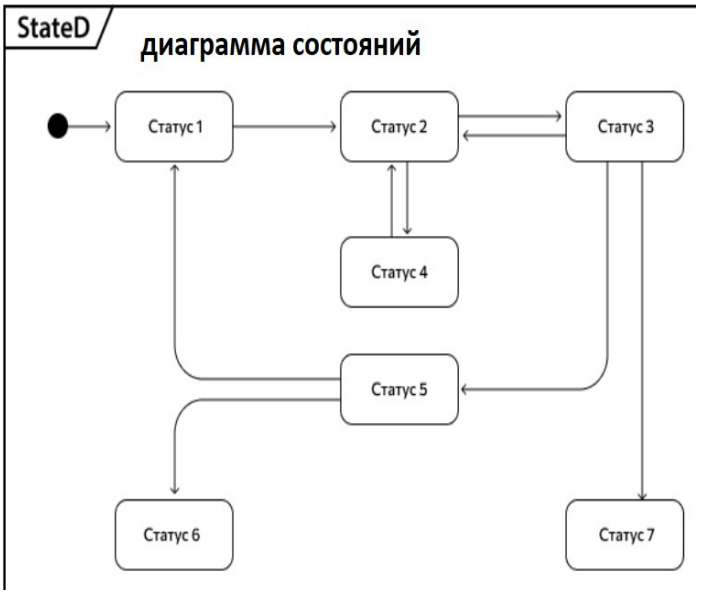
3. **Диаграмма классов-** Диаграммы классов применяются только в объектно-ориентированном программировании. Они являются визуальным представлением структуры классов в программе.

Диаграммы классов позволяют разработчикам правильно спроектировать архитектуру программного продукта.

4.Диаграмма состояний-показывает, как объект переходит из одного состояния в другое. Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса







ЛЕКЦИЯ 6-7

Тема: use case diagram. Принцип работы. Основные виды отношений.

1.диаграмма вариантов использования- (use case diagram)

Основные элементы диаграммы - участник (actor) и прецедент (вариант).

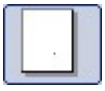
Участник - это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Участником может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности. Графически участник изображается “человечком”.

Прецедент (use case) - описание множества последовательных событий (включая варианты), выполняемых системой, которые приводят к наблюдаемому участником результату. Прецедент представляет поведение сущности, описывая взаимодействие между участниками и системой. Прецедент не показывает, “как” достигается некоторый результат, а только “что” именно выполняется. Прецеденты обозначаются очень простым образом - в виде эллипса, внутри которого указано его название.

Основные элементы диаграммы вариантов использования

На диаграмме вариантов использования можно отобразить следующие элементы нотации UML, доступные в панели элементов. На диаграммах UML для связывания элементов используются различные соединительные линии, которые называются *отношениями*. Каждое такое отношение имеет собственное название и используется для достижения определённой цели.

все виды отношений:

1.  Участник (Actor)
2.  Вариант (Use case)
3.  Граница
4.  Ненаправленная ассоциация
5.  Направленная ассоциация
6.  Обобщение
7.  зависимость (включение и расширение)
8.  комментарий
9.  Коннектор комментария

4,5 Отношение ассоциации

Мы хотим отображать на диаграмме информацию о том, какие варианты использования могут быть использованы каждым актёром.

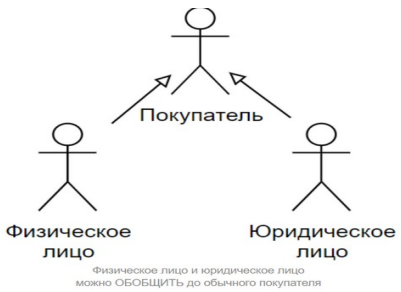
6.Отношение обобщения Заметим, что в нашей системе группы пользователей «Преподаватель» и «Классный руководитель» обладают схожими возможностями. Чтобы изобразить это на диаграмме, мы можем пойти одним из трёх путей:

1. **Дублировать варианты использования**, чтобы связать их с каждым схожим актёром (очевидно, неудачный вариант).
2. **Соединить каждого актёра со всеми нужными вариантами использования**. Это может породить множество пересечений линий, что не самым лучшим образом скажется на читаемости диаграммы.
3. **Показать с помощью одного из видов отношений, что актёры связаны между собой**. Это будет означать, что один из них может пользоваться всеми вариантами использования, с которыми соединён другой актёр.

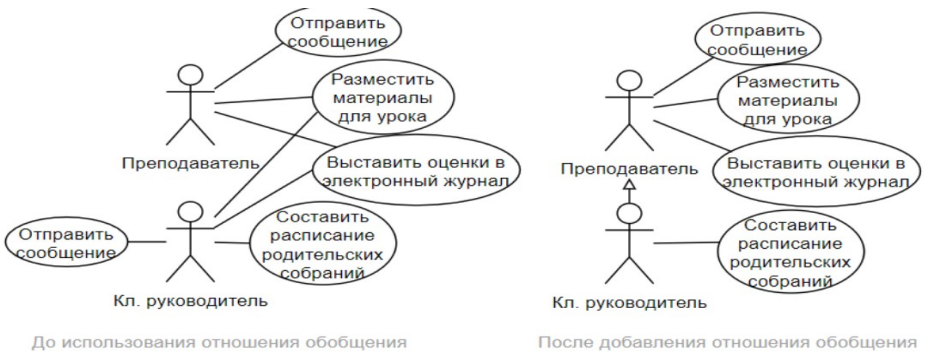
Последний вариант похож на принцип повторного использования кода при написании программ или на наследование классов в ООП (Объектно-ориентированное программирование). Преимущество этого варианта в том, чтобы уменьшить количество связей на диаграмме.

1.

2.



3.



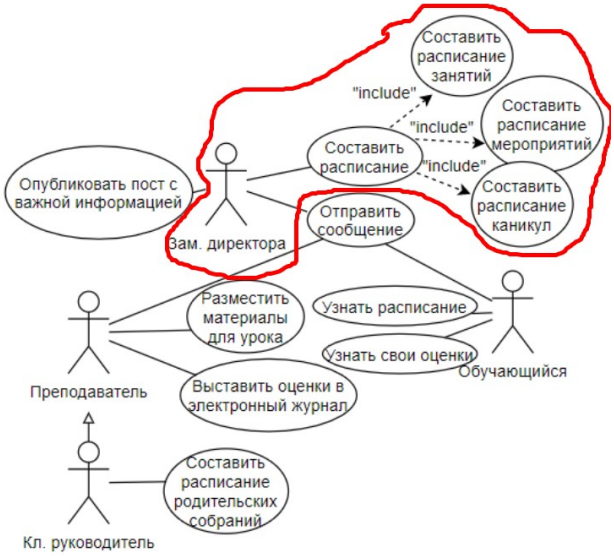
На рисунке сверху сразу видно, насколько понятнее становится диаграмма при использовании отношения обобщения: исчезли все повторы вариантов использования и

пересечения линий. Разумеется, это огромный плюс для тех, кто будет читать эту диаграмму в дальнейшем.

Давайте добавим действие «Узнать свои оценки». Логично предположить, что обучающиеся захотят не только знать список своих оценок, но и знать свою среднюю оценку за некоторый период времени или среднюю оценку по определённому предмету.

Изобразим это на диаграмме. Для этого создадим два варианта использования "Узнать среднюю оценку за некоторый период времени" и "Узнать среднюю оценку по предмету" и соединим их с *вариантом использования "Узнать свои оценки"* отношением обобщения.

7.Отношение включения и расширения



В общем случае, отношение включения используется, чтобы показать, что некоторый вариант использования ***включает*** в себя другой вариант использования в качестве составной части

Для заместителя директора мы отмечали, что ему нужно составлять расписания. *Условно* расписание можно поделить на три категории:

1. Расписание занятий
2. Расписание мероприятий

7.Отношение расширения Можно сказать, что отношение расширения - это выборочное отношение включения. Если *отношение включения* обозначает, что

элемент **обязательно** включается в состав другого элемента, то в случае *отношения расширения это включение* ***необязательно.***

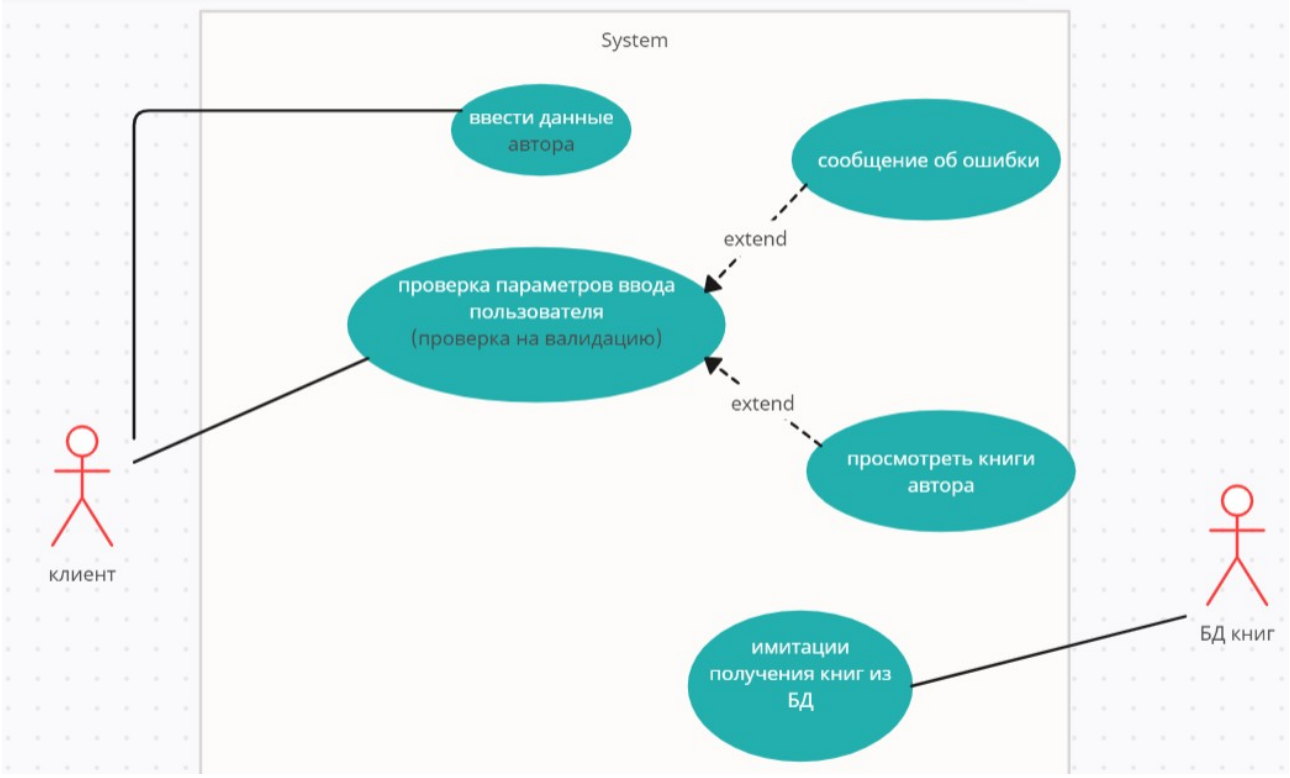
Во время дистанционного обучения обучающемуся необходимо выполнять домашние задания и присылать их в виде архива или фотографий учителям. Получается, нужно добавить возможность прикреплять файл к сообщению в нашей системе.

отношение расширения. *Отношение расширения обозначается пунктирной линией с V-образной стрелкой на конце, над стрелкой добавляется надпись* ***“extend ”.***

Общие рекомендации:

1. **Диаграммы моно изменять в любой** момент. Не нужно пугаться того, что требования к программе могут измениться или что вы что-то забыли отобразить на диаграмме. Вы можете добавить элементы к диаграмме, когда вам угодно.
2. **Не нужно засорять диаграмму слишком мелкими действиями**. Объедините все общие действия в одну группу под общим названием, чтобы было просто читать диаграмму.
3. Старайтесь не допускать пересечений соединительных линий. Это может затруднить чтение диаграммы для вас и для ваших коллег.
4. **Не дублируйте варианты использования на диаграмме**. Если приходится дублировать варианты использования, то элементы диаграммы надо постараться расставить по-другому.
5. **Пользуйтесь специальными компьютерными программами для построения диаграмм**. Это существенно упростит весь процесс моделирования.


ИНТЕРНЕТ-МАГАЗИН
Авторизованный пользователь
Авторизоваться
<<extend>>
Отредактировать профиль
<<include>>
Отменить заказ
Оформить заказ
<<extend>>
Редактировать заказ
<<include>>
Изменить заказ
Оплатить заказ
Неавторизованный пользователь
Посмотреть товары
Зарегистироваться


System
ввести данные автора
сообщение об ошибки
extend
проверка параметров ввода пользователя (проверка на валидацию)
extend
просмотреть книги автора
клиент
БД книг
имитации получения книг из БД

ЛЕКЦИЯ 8-10

Тема: Диаграмма последовательностей. Основные части диаграммы. Принцип работы и основные параметры.

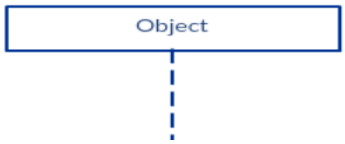
Диаграммы последовательностей, обычно используемые разработчиками, моделируют взаимодействия между объектами в едином сценарии использования. Они иллюстрируют, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования.

Проще говоря, диаграмма последовательности показывает различные части работы системы в “последовательности”, чтобы что-то сделать.

Обозначения диаграмм последовательности

Принцип построения: Схема последовательности построена таким образом, что она представляет собой временную шкалу, которая начинается сверху и постепенно опускается, чтобы отметить последовательность взаимодействий. Каждый объект имеет колонку, а сообщения, которыми обмениваются между собой, представлены стрелками.

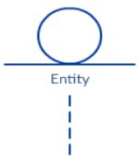
Части диаграммы последовательности



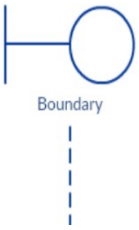
1.Нотация линии жизни обьект последовательность состоит из нескольких таких обозначений линии жизнеобеспечения, которые должны быть расположены горизонтально в верхней части диаграммы.

Никакие две нотации страховочной линии не должны перекрывать друг друга. Они представляют собой различные объекты или части, которые взаимодействуют друг с другом в системе во время последовательности.

2. линия жизни Actor – экземпляр участника процесса (роль на диаграмме прецедентов)



2.линия жизни Entity – Класс-сущность - обычно применяется для обозначения классов, которые хранят некую информацию о бизнес-объектах (соответствует таблице или элементу БД)



3. линия жизни Boundary – Класс-Разграничитель - используется для классов, отделяющих внутреннюю структуру системы от внешней среды (экранная форма, пользовательский интерфейс, устройство ввода-вывода). Объект большей предназначен для вызова методов класса, с которым он связан. Объект boundary показывает именно экранную форму, которая принимает и передает данные обработчику

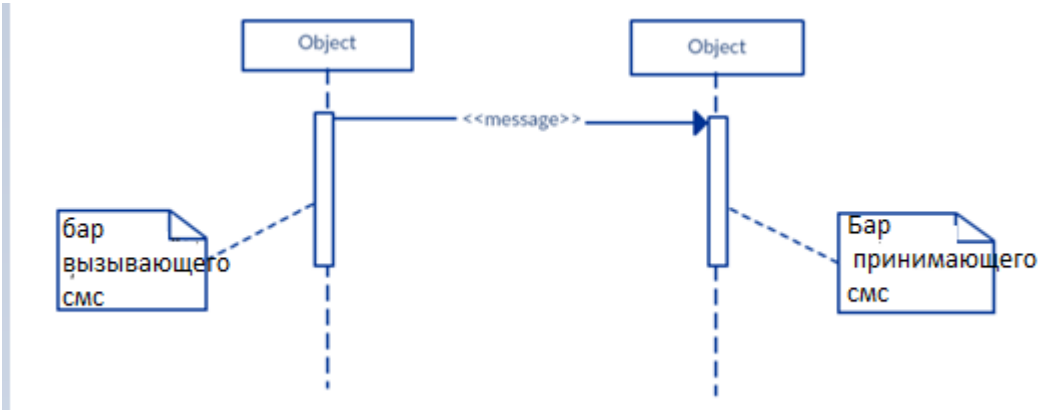
4. 

линия жизни Control – Класс-контроллер - активный элемент, который используются для выполнения некоторых операций над объектами (программный компонент, модуль, обработчик)

5. Активация – прямоугольник на линии жизни, который показывающий на период времени , в течении которого элемент выполняет операцию

взаимодействия между двумя объектами. Длина прямоугольника указывает на продолжительность пребывания объектов в активном состоянии.

На диаграмме последовательности взаимодействия между двумя объектами происходит, когда один объект посылает сообщение другому.



Использование строки активации на спасательных линиях вызывающего сообщения (объекта, отправляющего сообщение) и получателя сообщения (объекта, принимающего сообщение) указывает на то, что оба они активны/осуществляются во время обмена сообщением.

8.Стрелки -*используются для вызова процедур выполнения операций или обозначают отдельно вложенных потоков управления. Начало стрелки всегда соприкасается с линией жизни или блоком активации.*

Стрелка от одного к другому приемнику указывает сообщение на схеме последовательности. Сообщение может идти в любом направлении: слева направо, справа

налево или обратно к самому вызывающему сообщение абоненту. В то время как вы можете описать сообщение, отправляемое с одного объекта на другой, на стрелке, с разными заголовками стрелок, вы можете указать тип отправляемого или получаемого сообщения.

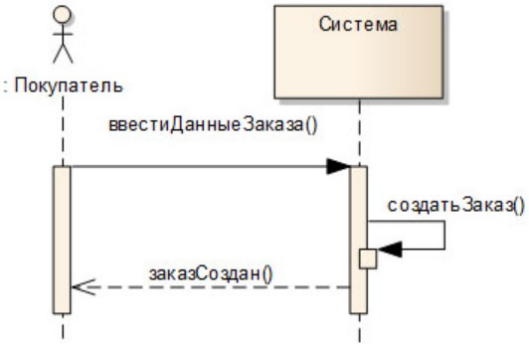
Стрелка сообщения содержит описание, известное как подпись сообщения.

Атрибут = имя_сообщения (аргументы): return_type

- **синхронное сообщение (synchCall)** - соответствует синхронному вызову операции и подразумевает ожидание ответа от объекта получателя. Пока ответ не поступит, никаких действий в Системе не производится. 

- **асинхронное сообщение (asynchCall)** - которое соответствует асинхронному вызову операции и подразумевает, что объект может продолжать работу, не ожидая ответа. 

- **ответное сообщение (reply)** – ответное сообщение от вызванного метода. Данный вид сообщения показывается на диаграмме по мере необходимости или, когда возвращаемые им данные несут смысловую нагрузку.
- **потерянное сообщение (lost)** – сообщение, не имеющее адресата сообщения, т.е. для него существует событие передачи и отсутствует событие приема
- **найденное сообщение (found)** – сообщение, не имеющее инициатора сообщения, т.е. для него существует событие приема и отсутствует событие передачи



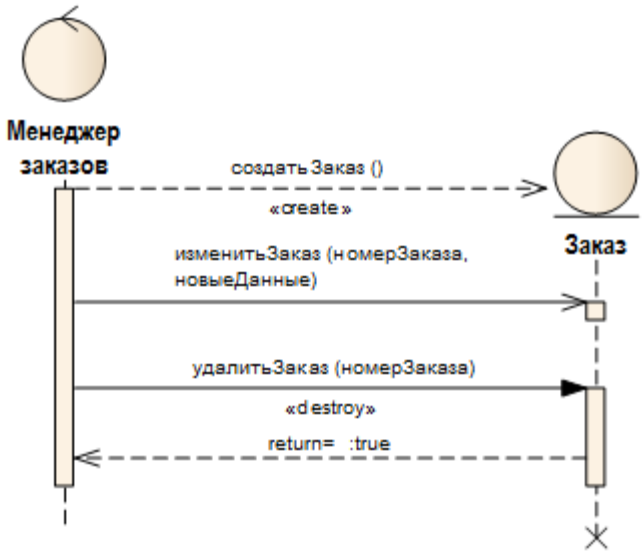
На диаграмме состояний частично показан обмен сообщениями в рамках сообщений инициирующих изменение состояния объекта., если есть действие стрелка выбирается с темным наконечником

Диаграмма последовательности объединяет диаграмму деятельности, диаграмму состояний и диаграмму классов.

диаграмме последовательности мы можем увидеть следующие аспекты:

- **Сообщения**, побуждающие объект к действию
- **Действия**, которые вызываются сообщениями (методы) – зачастую это передача сообщения следующему объекты или возвращение определенных данных объекта
- **Последовательность** обмена сообщениями между объектами

- **Рекурсивный вызов-** действие отправляющий запрос само себе, в данном случае она говорит себе создать заказ



Если нам нужно создать участника(в данном случае новый заказ), мы ведем стрелку к этому элементу, если он не был создан.

После этого всегда должен быть блок активации на этой линии жизни, в данном случае это активатор для изменения заказа, так как он был создан пустым.

Сообщение в большинстве случаев (за исключением диаграмм, описывающих концептуальный уровень Системы) это вызов методов отдельных объектов, поэтому для корректного исполнения метода в сообщении необходимо передать какие-то данные и определить, что мы хотим видеть в ответ. При именовании сообщения на уровне проектирования реализации системы в качестве имени сообщения следует использовать имя метода.

Отдельные фрагменты диаграммы взаимодействия можно выделить с помощью фрейма. Фрейм должен содержать метку оператора взаимодействия. UML содержит следующие операнды:

- **Alt -** Несколько альтернативных фрагментов (alternative); выполняется только тот фрагмент, условие которого истинно

НАШ ОПЕРАТОР if-then-else; case; switch

- **Opt -** Необязательный (optional) фрагмент; выполняется, только если условие истинно. Эквивалентно **alt** с одной веткой

if-then, без ветки иначе

- **Par -** Параллельный (parallel); все фрагменты выполняются параллельно

Это наши разграничители на схеме отделает наш if от else

- **loop -** Цикл (loop); фрагмент может выполняться несколько раз, а защита обозначает тело итерации

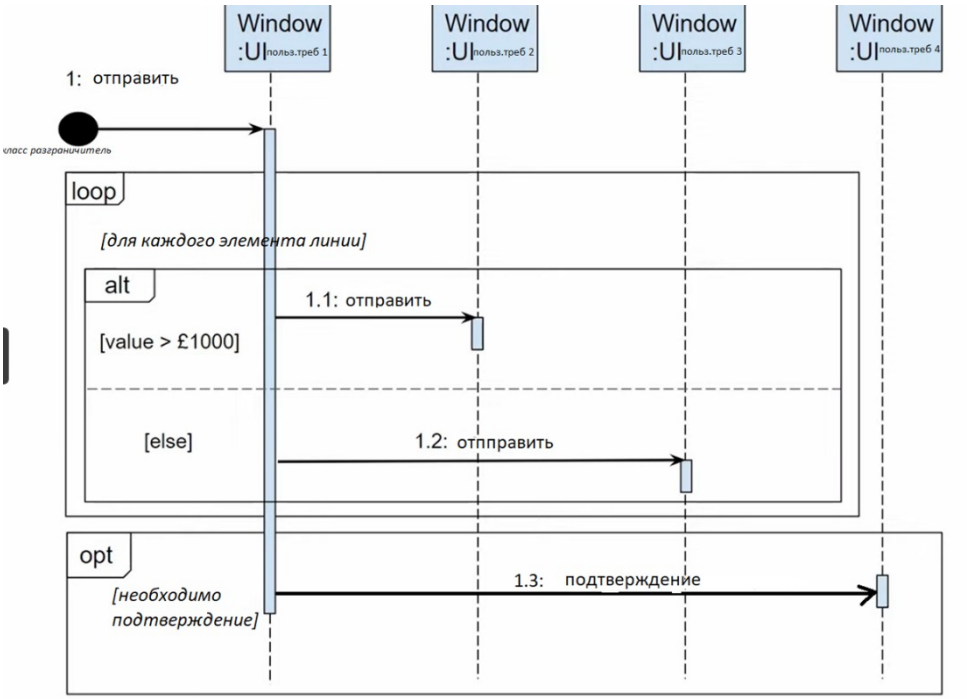
- **Neg -** Отрицательный (negative) фрагмент; обозначает неверное взаимодействие

К примеру, блок ждет пароля от пользователя, время ожидания вышло и ему выведется сообщение Время вышло

• **ref -** Ссылка (reference); ссылается на взаимодействие, определенное на другой диаграмме. Фрейм рисуется, чтобы охватить линии жизни, вовлеченные во взаимодействие. Можно определять параметры и возвращать значение

ссылка на другой блок или диаграмму

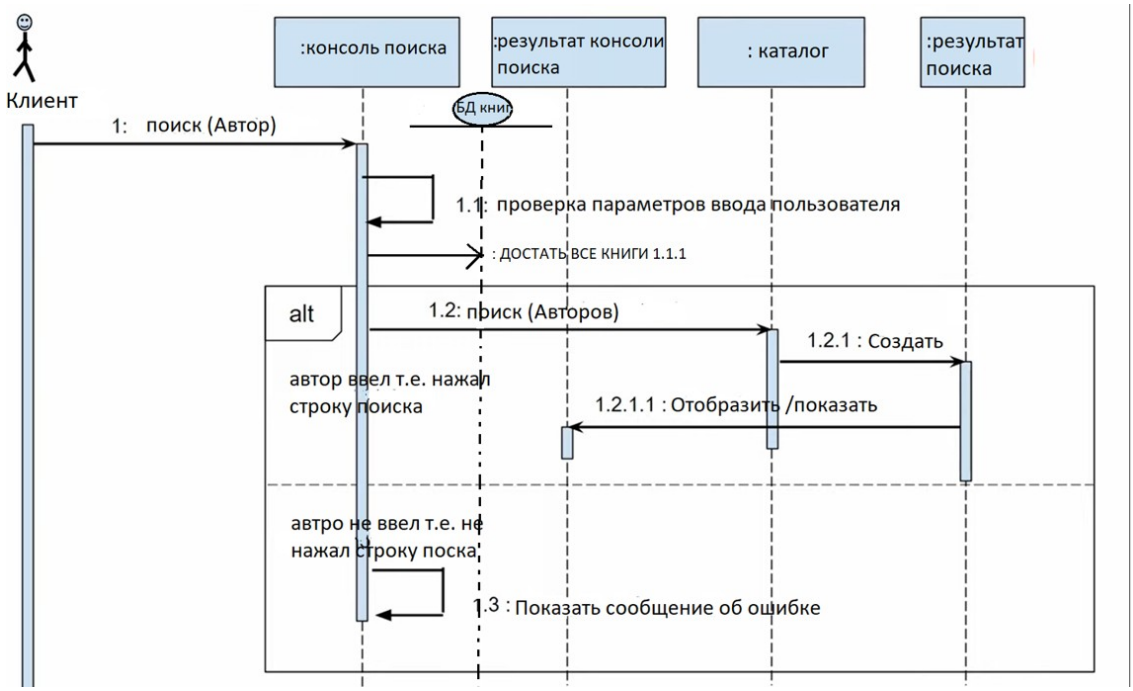
• **Sd -** (sequence diagram); используется для очерчивания всей диаграммы последовательности, если это необходимо.



Как показать цикличность

В данном случае используется класс разграничитель(черный кружек)и 4 линии жизни (пользовательские инерфейсы 4 шт). Прямоукольник loop выделил область которая указывет на цикл, а прямоуголник с alt указывает на условие(пока пользователь не достиг просмотра 1000минут -он видит сообщение 1.1, если время становится больше 1000-он увидит сообщение 1.2. Далее Блок блок ALT,он показывет истенное выражение- в данном случае пользователю нравится интерфейс и он нажимает на лайк-это и есть подтверждение.

Цикличность потока взаимодействия может быть представлена на диаграмме последовательности с помощью операнда loop. При использовании оператора цикла можно указать минимальное и максимальное число итераций. Также фрейм должен содержать условие, при наступлении которого взаимодействие повторяется.



есть два сценария:

1. Основной
2. Альтернативный

1.Основной: КЛИЕНТ вводит ФИО автора и нажимает кнопку поиска. После чего система проверяет параметры ввода пользователя, если валидация пройдена- система ищет Автора по каталогам, когда поиск окончен -система выдает результат

2. Если КЛИЕНТ не ввел имя автора, но нажал на кнопку поиска, то система выводит сообщение об ошибке.

ЛЕКЦИЯ 11-12

Тема: Диаграмма СОСТОЯНИЙ. Принцип работы. Список основных действий. Типы сообщений. Правила построения диаграмм.

Описать поведение отдельно взятого объекта помогает диаграмма состояний.

Также зачастую диаграмма состояний используется аналитиками для описания последовательности переходов объекта из одного состояния в другое.

Диаграмма состояний покажет нам все возможные состояния, в которых может находиться объект, а также процесс смены состояний в результате внешнего влияния.

Основными элементами диаграммы состояний являются «Состояние» и «Переход».

состояние может содержать только имя или имя и дополнительно список внутренних действий. Список внутренних действий содержит перечень действий или деятельностей, которые выполняются во время нахождения объекта в данном состоянии. Данный список фиксированный.

Список основных действий включает следующие значения:

• **entry** - действие, которое выполняется в момент входа в данное состояние (входное действие);

• **exit** - действие, которое выполняется в момент выхода из данного состояния (выходное действие);

• **do** - выполняющаяся деятельность ("do activity") в течение всего времени, пока объект находится в данном состоянии

• **defer** - событие, обработка которого предписывается в другом состоянии, но после того, как все операции в текущем будут завершены.

Переход — это элемент диаграммы состояний, который представляет собой переход объекта из одного состояния в другое. Он определяет событие или условие, которое вызывает изменение состояния, и указывает, какой переход должен быть выполнен при наступлении этого события или условия.

Переходы могут быть направленными или ненаправленными. *Направленный переход указывает однонаправленный поток выполнения от одного состояния к другому. Ненаправленный переход представляет переход, который может быть выполнен в обоих направлениях между состояниями*. ⟶

Правила построения диаграммы состояний

При построении диаграммы состояний UML рекомендуется следовать определенным правилам, чтобы обеспечить понятность и четкость модели. Вот некоторые из основных правил построения диаграммы состояний:

1. **Определите объект или систему.**

определите, для какого объекта или системы вы создаете диаграмму состояний. Это поможет установить контекст и ограничения модели.

2. **Выделите состояния.**

Определите все возможные состояния объекта или системы, которые имеют значимое поведение. Состояния должны быть достаточно четкими и понятными, чтобы отразить основные переходы и поведение объекта.

3. **Определите переходы**.

Опишите переходы между состояниями, указывая события или условия, которые инициируют переход. Переходы должны быть логичными и понятными, а защитные условия, если применимы, должны быть ясно определены.

4. **Укажите действия.**

Определите действия или операции, которые выполняются при переходе между состояниями. Это может быть связано с изменением переменных, вызовом методов или выполнением других действий.

5. **Уточните внутреннее поведение**.

Если состояние является сложным или имеет дополнительные подсостояния, уточните внутреннее поведение каждого состояния, определив его собственные переходы, действия и события.

6. **Используйте подходящие символы.**

Применяйте соответствующие символы UML для обозначения состояний, переходов, событий, действий и других элементов диаграммы состояний. Это поможет обеспечить единообразие и понятность модели.

7. **Поддерживайте четкость и простоту.**

Старайтесь поддерживать диаграмму состояний простой, понятной и легко читаемой. Избегайте перегруженности символами и излишней сложности. Выделите основные состояния и переходы, которые наиболее важны для понимания поведения объекта или системы.

8. **Документируйте и комментируйте**.

Предоставьте достаточное количество комментариев и документации для объяснения основных аспектов модели. Это поможет другим разработчикам и заинтересованным сторонам лучше понять модель и ее назначение.

Помните, что диаграмма состояний служит для визуализации поведения объекта, поэтому поддерживайте её ясность и понятность.

Как построить диаграмму состояний ШАГИ

Шаг 1: Определение состояний

На этом шаге необходимо определить все состояния в которых может находиться моделируемый объект. Определим состояния на примере устройства климат-контроля. Допустим, устройство может находиться в состояниях «Режим ожидания», «Охлаждение», «Обогрев» и «Вентиляция».

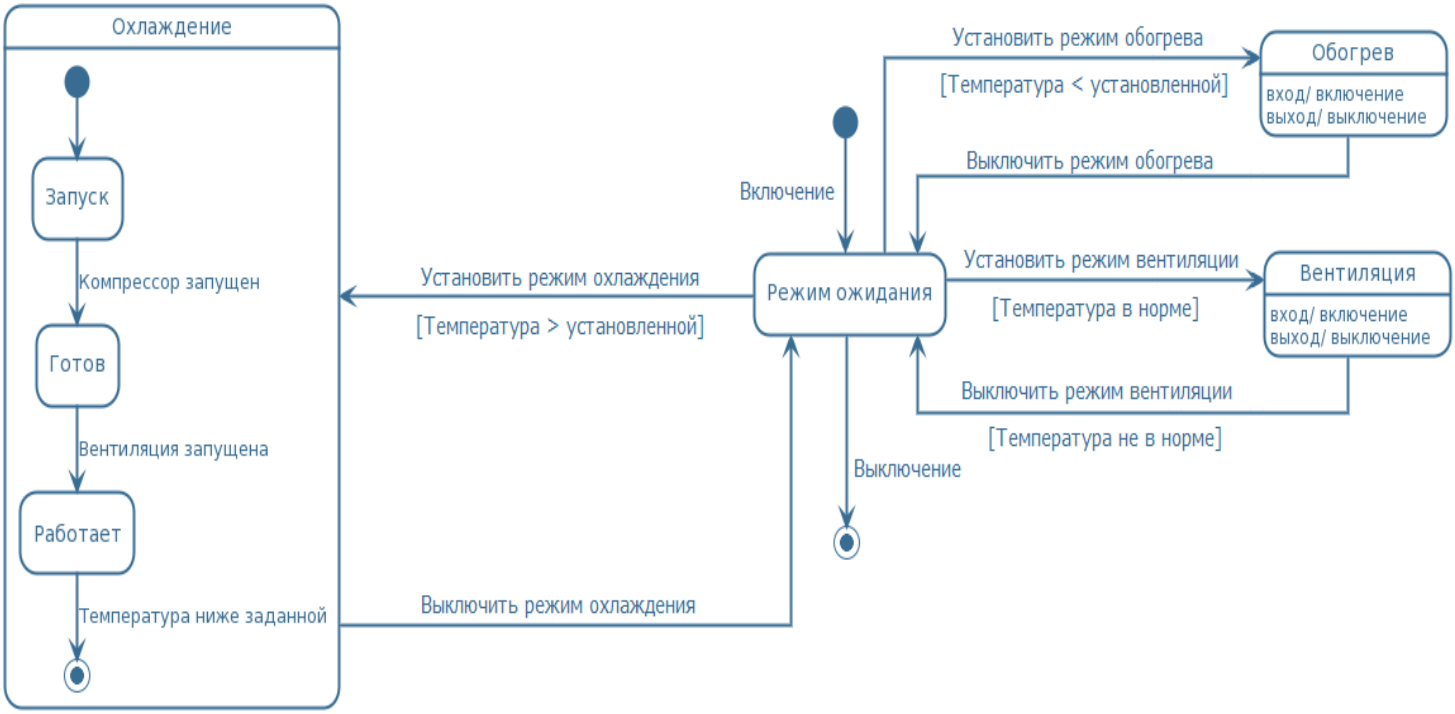
Шаг 2: Определение переходов

На этом шаге необходимо определить переходы между состояниями, события, вызывающие переходы, и защитные условия. Предположим, что у нас есть события «Включить», «Выключить», «Установить режим охлаждения», «Установить режим обогрева» и «Установить режим вентиляции».

Шаг 3: Добавление защитных условий

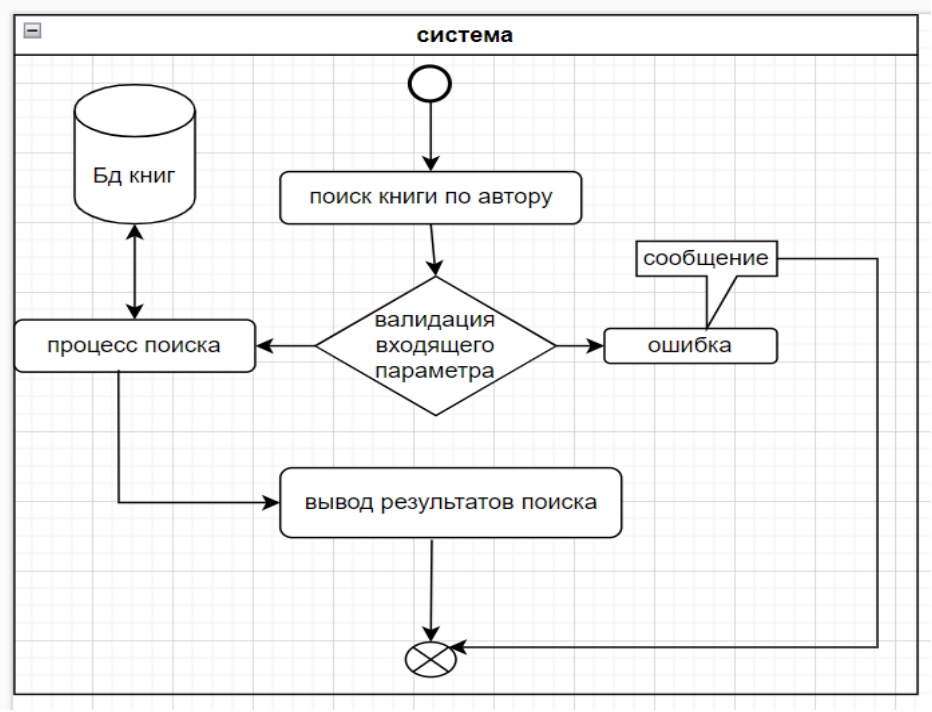
Добавим защитные условия для контроля переходов. Теперь определим защитные условия, для события «Установить режим охлаждения» защитным условием может быть «Температура выше заданной», для «Установить режим обогрева» — «Температура ниже заданной»

В результате, построенная диаграмма состояний выглядит следующим образом



Пример state diagram для кейса «Управление температурой»

https://app.diagrams.net/


система
Бд книг
поиск книги по автору
сообщение
валидация входящего параметра
процесс поиска
ошибка
вывод результатов поиска

ЛЕКЦИЯ 13-15

Тема: DFD методология моделирования. Нотация и принципы работы. Разбор примеров. Знакомство с правила распределения потоков. Определение правил оформления функций

Стандарт описания бизнес-процессов **DFD — Data Flow Diagram** переводится как диаграмма потоков данных и используется для описания процессов верхнего уровня и для описания реально существующих в организации потоков данных.

Диаграммы потоков данных показывают, как каждый процесс преобразует свои входные данные в выходные, и выявляют отношения между этими процессами. DFD представляет моделируемую систему как сеть связанных работ.

При построении DFD-схемы бизнес-процесса *нужно помнить, что данная схема показывает потоки материальных и информационных потоков и ни в коем случае не говорит о временной последовательности работ*, хотя в большинстве случаев временная последовательность работ и совпадает с направлением движения потоков в бизнес-процессе.

три уровня проектирования DFD

Проектирование DFD-диаграмм охватывает 3 уровня абстракции:

1. Концептуальная диаграмма показывает движения потоков данных на самом верхнем уровне абстракции. Концептуальная диаграмма детализируется

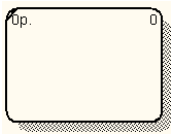
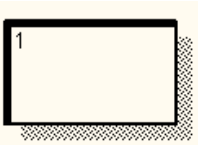
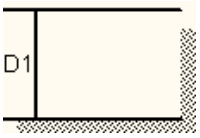
логическим и **физическим** потоками данных.

2. Диаграмма логических потоков данных отражает будущее или текущее состояние, описывая какие преобразования должны происходить независимо от физических ограничений.

3. Диаграмма физических потоков данных моделирует хранилища данных: принтеры, формы, устройства и другие проявления данных.

Основные компоненты нотации:

1. **Работы (Activities).** Отображают процессы обработки и изменения информации
2. **Стрелки (Arrows).** Отображают информационные потоки
3. **Хранилища данных (Data Store).** Отображают данные, к которым осуществляется доступ, эти данные используются, создаются или изменяются работами
4. **Внешние сущности (External References).** Отображают объекты, с которыми происходит взаимодействие

Описание	Графическое представление	
Работа (Activity)	Объект обозначает функции или процессы, которые обрабатывают и изменяют информацию.	
Информационный поток (Precedence)	Объект обозначает информационный поток от объекта-источника к объекту-приемнику.	
Внешняя ссылка (External reference)	Указывают на место, организацию или человека, которые участвуют в процессе обмена информацией с системой, но располагаются за рамками этой диаграммы.	
Хранилище данных (Data store)	Хранилища данных представляют собой собственно данные, к которым осуществляется доступ, эти данные также могут быть созданы или изменены работами. На одной диаграмме может присутствовать несколько копий одного и того же хранилища данных.	

Требования к оформлению функций:

1. Каждая функция должна иметь идентификатор;

2. Названия работы нужно формулировать согласно следующее формуле:

Название работы = Действие + Объект, над которым действие осуществляется

Например, если эта работа связана с действием по продаже продукции, то ее нужно назвать Продажа продукции

3. Название работы должно быть по возможности кратким (не более 50 символов) и состоять из 2-3 слов. В сложных случаях также рекомендуется для каждого краткого названия работы сделать ее подробное описание, которое поместить в глоссарий.

Требования к оформлению потока данных:

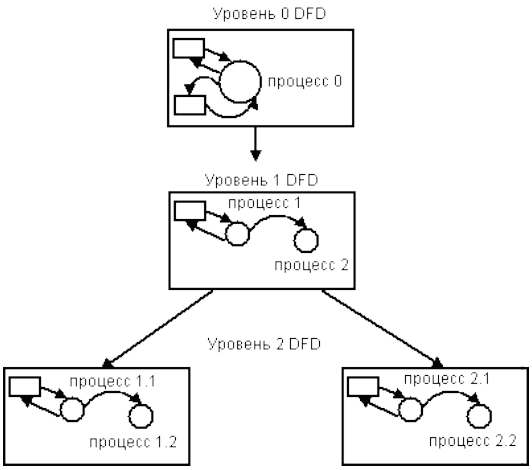
1. Название потока нужно формулировать согласно следующей формуле:

Название потока = Объект, представляющий поток + Статус объекта

Если речь идет о продукции, которую отгрузили клиенту, то поток можно назвать <Продукция, отгруженная> или <Продукция, отгруженная клиенту>. В данном случае <Продукция> это объект, представляющий поток, а <отгруженная клиенту> — статус объекта.

2. Название должно быть по возможности кратким и состоять из 2-3 слов.

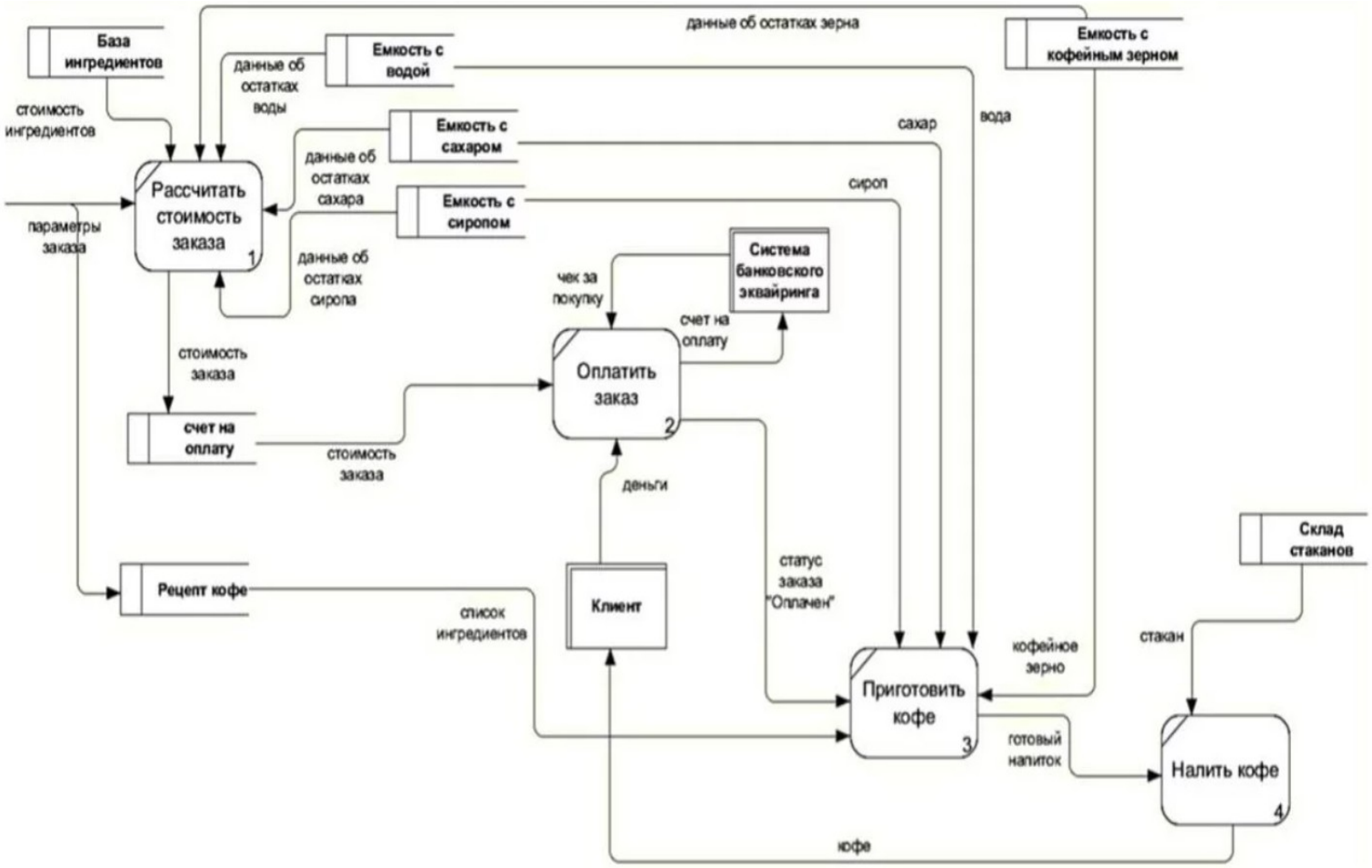
При детализации должны выполняться следующие правила:



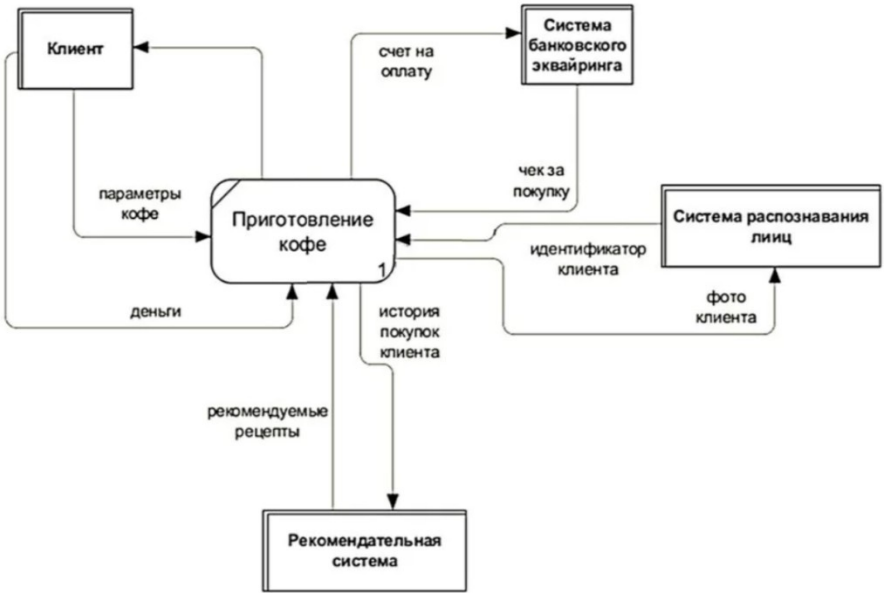
- **правило балансировки** — *при детализации процесса дочерняя диаграмма в качестве внешних источников/приемников данных может* иметь только те компоненты данные с которыми имеет информационную связь соответствующий процесс на родительской диаграмме;
- **правило нумерации** — при детализации процессов должна поддерживаться их иерархическая нумерация.
- **правило семи** — для того, чтобы диаграмма легко читалась, количество функций на диаграмме не должно быть больше семи.

На этапе декомпозиции процесса **Приготовление кофе** важно понять, что в первую очередь необходимо рассчитать стоимость заказа на основании параметров заказа: сколько требуется воды, сиропа, зерен и сахара.

Кроме того, нужно знать, достаточно ли этих ингредиентов в аппарате. Данные об ингредиентах мы обозначим как хранилища, которые передают данные об остатках в процесс **Рассчитать стоимость заказа**. Стоимость ингредиентов передаётся в процесс расчёта из соответствующей базы. Результатом процесса является рассчитанная стоимость заказа, которая попадает в хранилище **Счёт на оплату**. Дальше стоимость заказа поступает в процесс **Оплатить заказ**, клиент вносит деньги, а сам процесс взаимодействует с внешней сущностью **Система банковского эквайринга**. Процесс направляет счёт на оплату и получает чек за покупку.



<u>приготовление кофе в кофейном автомате с системой распознавания лиц.</u>



Когда заказ оплачен, ингредиенты и рецепт передаются в процесс **Приготовить кофе**. Готовый напиток наливается в стаканчик, переданный из **Склада стаканов**. Это происходит в процессе **Налить кофе**. В результате клиент получает готовый напиток.

Давайте добавим в этот пример немного киберпанка. Предположим, что у нас умный кофейный автомат, который распознает клиента и хранит историю его покупок. Аппарат помнит, что клиент по понедельникам в хорошем настроении пьет сладкий латте, а в плохом настроении предпочитает маленькую чашечку бодрящего эспрессо.

Контекстная диаграмма в этом случае дополняется потоками идентификации клиента и формирования рекомендаций. В диаграмме также появится несколько новых внешних сущностей: **Система распознавания лиц** и **Рекомендательная система**. В качестве тренировки вы можете самостоятельно раскрыть эту контекстную диаграмму. Она будет похожа на предыдущий пример, но будет содержать ещё больше процессов обмена данными.

Пример DFD: предоставление контента по интересам пользователя

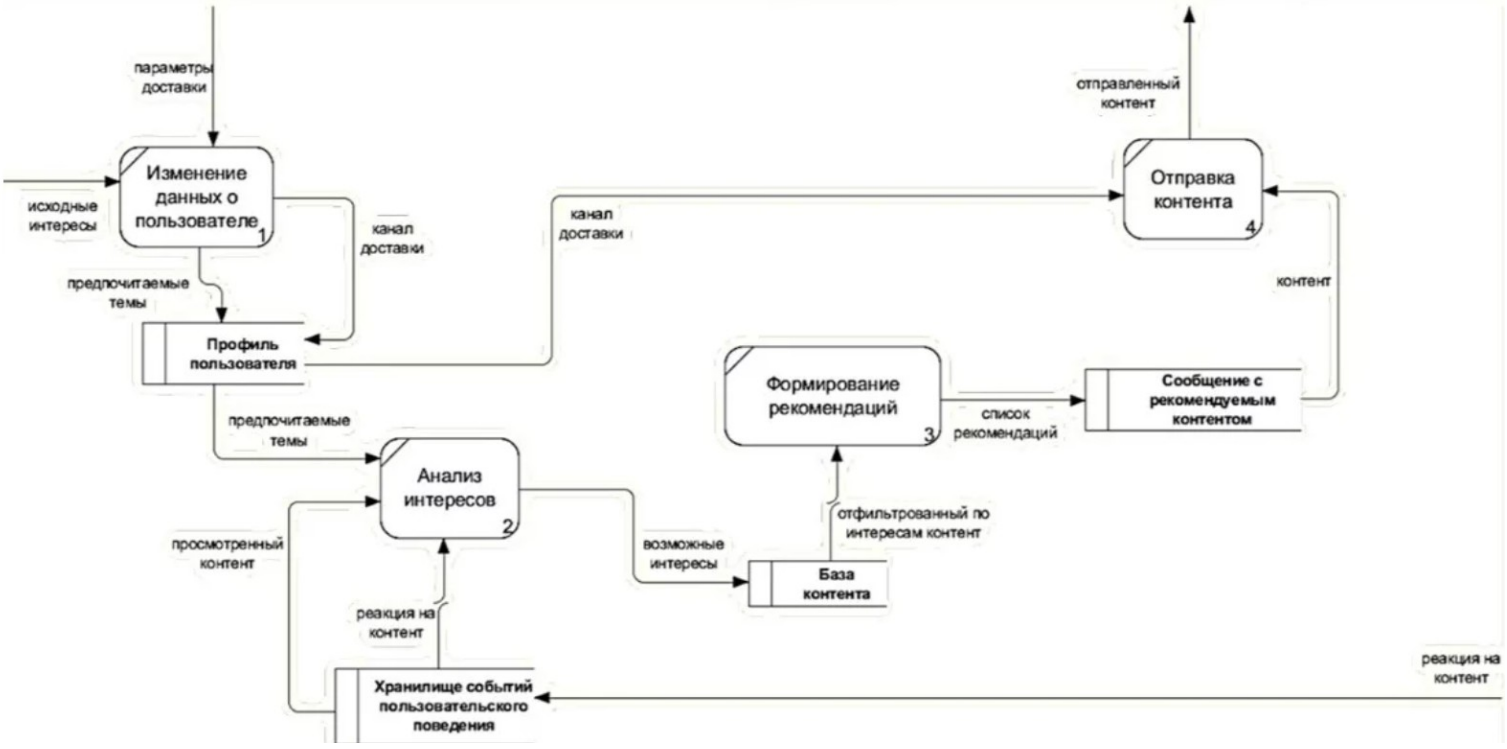
систему по предоставлению пользователю контента с учетом его интересов. Контент поступает по каналам, которые предпочел клиент: это может быть telegram, любой другой мессенджер или просто email.

Как выглядит процесс?

Клиент вводит первичные параметры доставки контента и свои исходные интересы, а система будет направлять ему контент. Поскольку система интеллектуальная, контент будет формироваться с учетом **реакций** и **предпочтений** пользователя (Content-based filtering).

Если пользователю нравится смотреть на котиков, будем показывать ему больше котиков. Если пользователь указал, что подборка котиков ему не очень интересна, будем исключать такой материал из контента.

При этом система сама будет определять, что нравится пользователю, а что не нравится.

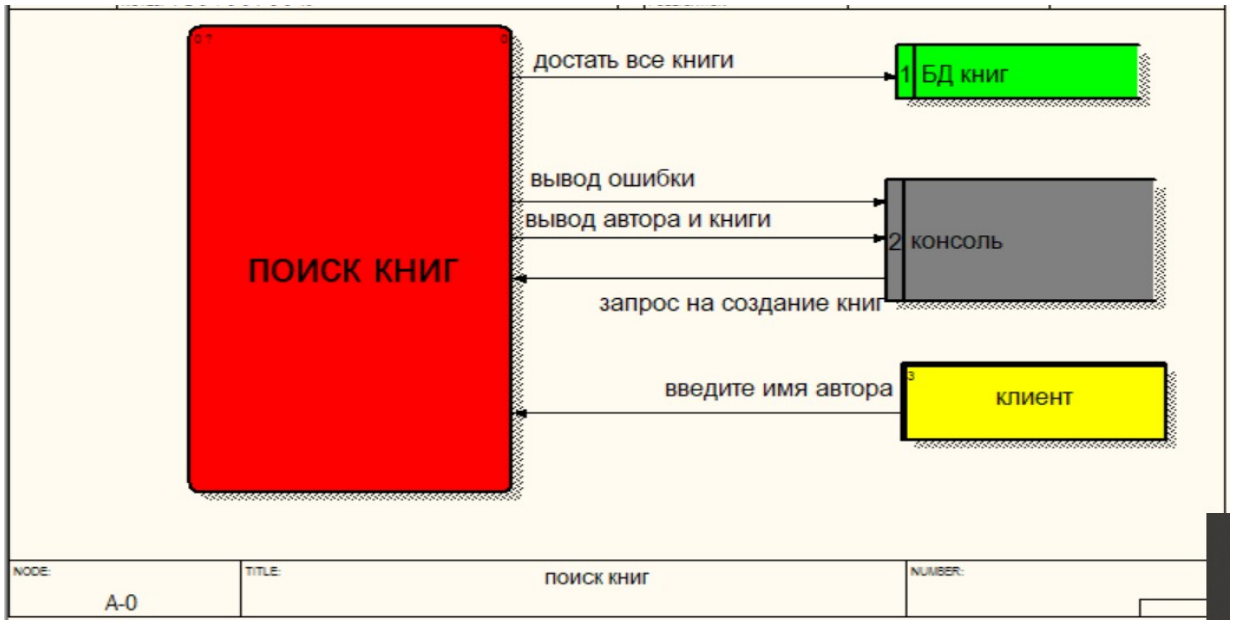


В процессе **Изменение данных пользователем**, куда приходят исходные интересы и параметры доставки, формируются предпочитаемые темы. Они передаются в промежуточное хранилище **Профиль пользователя**.

Просмотренный контент и — главное — реакция на него собираются в **Хранилище событий пользовательского поведения**. Как правило, это NoSQL-хранилища. Для сбора реакций на контент система анализирует поведение пользователя. Например, просмотрел ли пользователь данное видео полностью или быстро закрыл, прочитал статью и оставил комментарий либо поделился ссылкой и т.д.

В процессе **Анализа интересов**, куда попадают предпочитаемые пользователем темы и контент, который он просмотрел, формируется поток возможных интересов. Так система формирует **Базу контента**, откуда отфильтрованный контент попадает в процесс **Формирование рекомендаций**.

Сформированный список рекомендаций передаётся в промежуточное хранилище **сообщения с рекомендуемым контентом**. Процесс **Отправка** с учетом предпочитаемого канала доставляет контент пользователю.

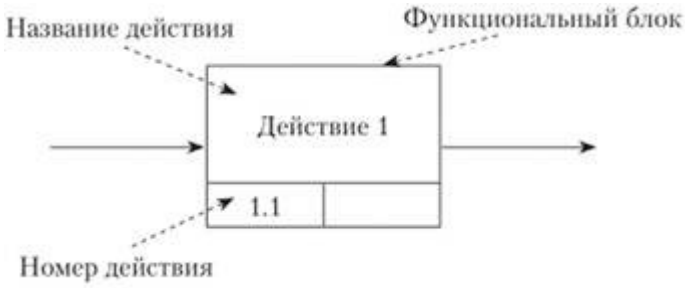


ЛЕКЦИЯ 16-17

Тема: бизнес-моделирование в нотации IDEF3.Обзор нотации. Виды соединений. Типы связей. Потоки данных

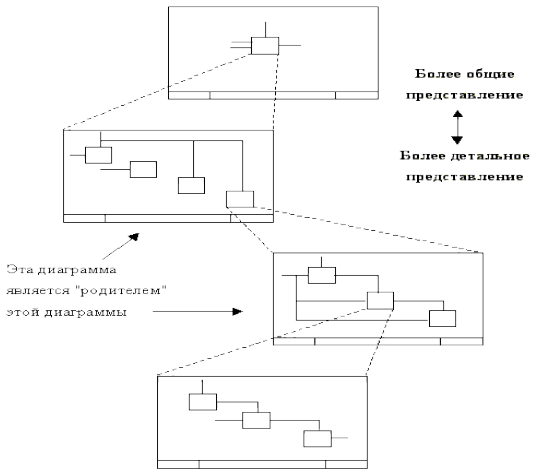
IDEF3 является технологией, хорошо приспособленной для сбора данных, требующихся для проведения структурного анализа системы.

В отличие от большинства технологий моделирования бизнес-процессов, **IDEF3** не имеет жестких синтаксических или семантических ограничений, делающих неудобным описание неполных или нецелостных систем.



Функциональный элемент (элемент поведения, единица работы) используется для обозначения действия, работы или события. Он отражается в виде прямоугольника, в центре которого указывается название действия (глагол или отглагольное существительное). Внизу блока указывается номер действия с учетом номера родительской диаграммы

*Если модель в нотации **IDEF0 позволяет получить общее представление о функциях**, выполняемых моделируемой системой, связях между функциями, механизмах исполнения, **то модель в нотации IDEF3 позволяет проследить логику взаимодействия функций, их последовательность и взаимозависимость связками AND, OR, XOR.***



Можно сначала построить функциональную модель в нотации IDEF0, проведя исследования предметной области. Затем, используя полученные знания о предметной области, построить отдельную модель в нотации IDEF3.

А можно создать смешанную модель, дополняя по мере необходимости функциональную модель в нотации IDEF0 диаграммами в нотации IDEF3. Также можно дополнять модель DFD диаграммами в нотации IDEF3.

В каждом конкретном случае моделирования системы принимается решение о необходимости построения каждого вида модели.

Основная цель нотации IDEF3 — дать бизнес-аналитикам возможность описать ситуацию, когда процессы (действия) выполняются в определенной последовательности и взаимной зависимости, а также описать объекты, участвующие совместно в одном процессе.

Имитационное тестирование часто используют для оценки эксплуатационных качеств разрабатываемой системы. Более подробно методы имитационного анализа будут рассмотрены ниже.

Методология IDEF3

Стандарт IDEF0, который был рассмотрен ранее является развитием классического DFD – подхода и предназначен для описания бизнес-процессов верхнего уровня. Для описания временной последовательности и алгоритмов выполнения работ стандарт IDEF0 не подходит. Для решения этой задачи стандарт IDEF0 получил дальнейшее развитие в результате чего был разработан стандарт IDEF3, который входит в семейство стандартов IDEF.

Стандарт IDEF3 предназначен для описания бизнес-процессов нижнего уровня и содержит объекты – логические операторы, с помощью которых показывают альтернативы и места принятия решений и в бизнес-процессе, а также объекты – стрелки с помощью которых показывают временную последовательность работ в бизнес-процессе (рис. 5).

виды соединений:

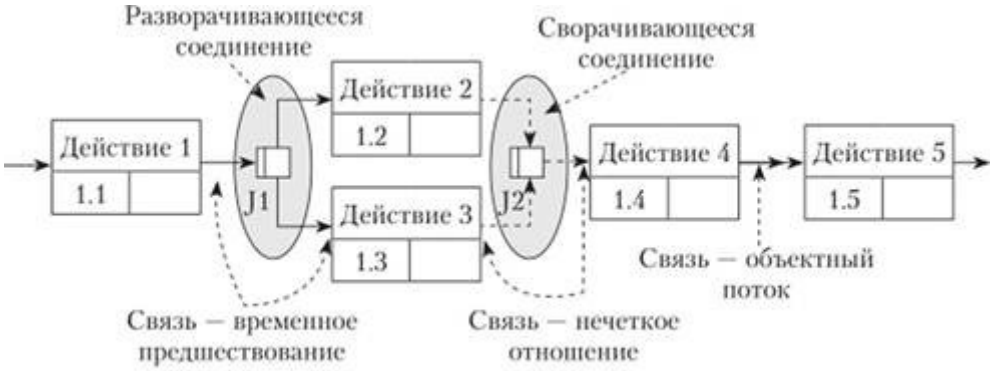
1) **разворачивающиеся соединения**, используемые для отражения связей, где завершение одного процесса инициирует запуск нескольких других процессов:

2) **сворачивающиеся соединения**, используемые для отражения связей, где завершение нескольких процессов приводит к запуску следующего одного процесса.

Разворачивающиеся и сворачивающиеся соединения могут быть также нескольких типов:

- • "и" (обозначается квадратом с символом "&");
- • "исключающее “или”" (обозначается квадратом с символом "X");
- • "или" (обозначается квадратом с символом "O").

На рис. 5.10 приведен образец построения ШЕЕЗ-диаграммы.



В отличие от классической методологии WFD в стандарте IDEF3 связи между работами делятся на три типа, обозначения, названия и смыл которых, приведены в таблице 3.

Таблица 3. Типы связей между работами в стандарте IDEF3.

Название связи	**Вид связи**	**Смысл связи**
Связь предшествования		Обозначает, что вторая работа начинает выполняться после завершения первой работы.
Связь отношения		Обозначает, что вторая работа может начаться и даже закончиться до того момента, когда закончится выполнение первой работы.

Связь потоков объектов		Одновременно обозначает временную последовательность работ и материальный либо информационный поток. В данном случае вторая работа начинает выполняться после завершения первой работы. При этом выходом первой работы объект название которого надписано над стрелкой (в данном случае документ). Эта связь также обозначает, что объект порождаемый первой работой, используется в последующих работах.

Помимо наличия нескольких типов связей между работами в стандарте IDEF3 логические операторы, которые в данном случае называются перекрестками также делятся на несколько типов: "Исключающий ИЛИ", "И" и "ИЛИ".

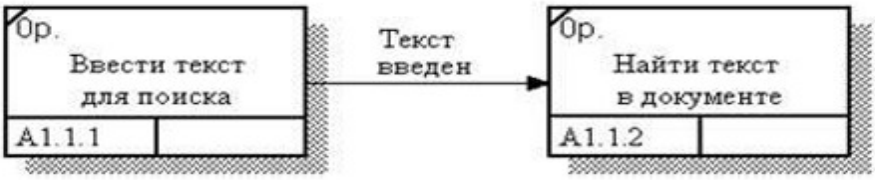


Рис. 2. Связь "временное предшествование" между действиями A1.1.1 и A1.1.2

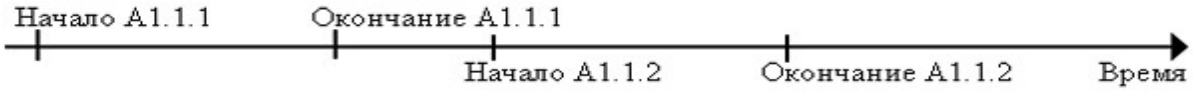


Рис. 3. Временная шкала выполнения действий для рис. 9.2

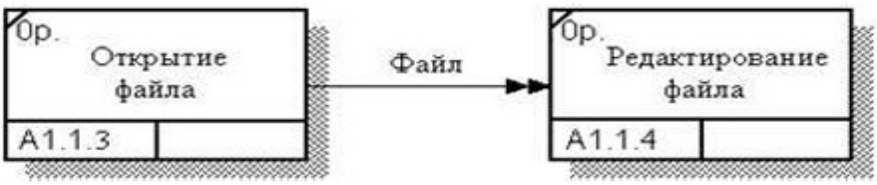


Рис. 4. Объектная связь между действиями A1.1.3 и A1.1.4

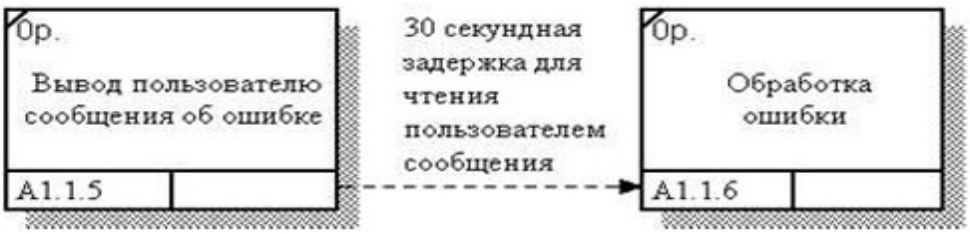


Рис. 5. Связь "нечеткое отношение"

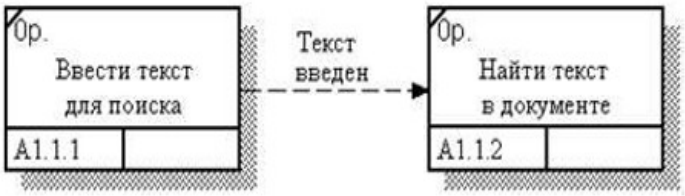
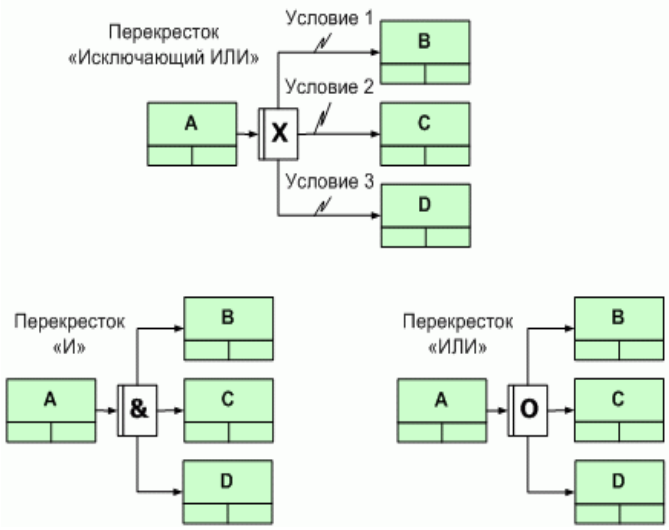


Рис. 6. Альтернативная связь предшествования

ЛЕКЦИЯ 18-19

Тема: Бизнес-моделирование в нотации IDEF3.Обзор нотации. Типы соединений. Операторы связей между процессами.



Перекресток "Исключающий ИЛИ" обозначает, что после завершения работы "А" (рис. 6), начинает выполняться только одна из трех расположенных параллельно работ B, C или D в зависимости от условий 1, 2 и 3.

Перекресток "И" обозначает, что после завершения работы "А", начинают выполняться одновременно три параллельно расположенные работы B, C и D.

Перекресток "ИЛИ" обозначает, что после завершения работы "А", может запуститься любая комбинация трех параллельно расположенных работ B, C и D.

Соединение типа **"исключающее “или”**:

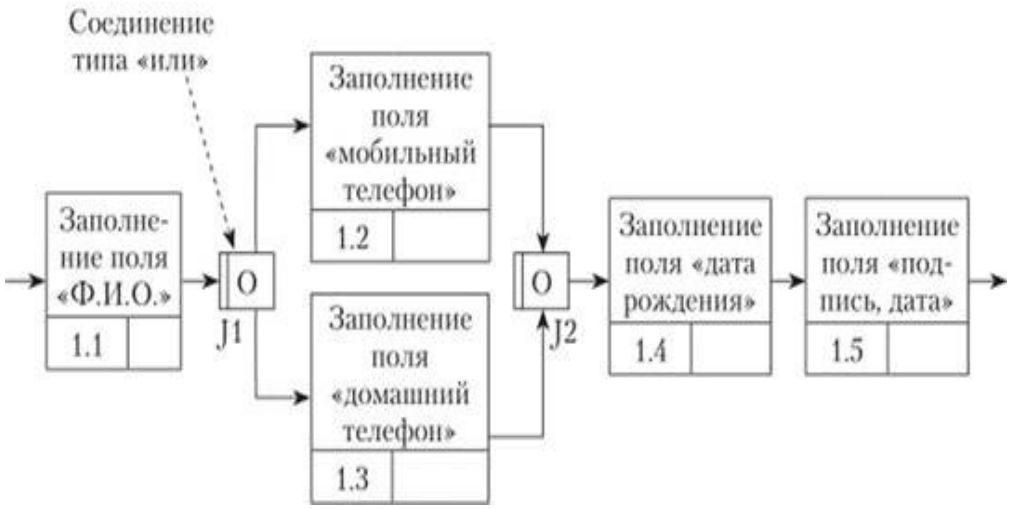
"исключающее “или”" используется для того, чтобы показать, что результатом согласования проекта договора может быть:

а) проект договора согласован;

б) по проекту договора есть замечания и он отправлен на доработку

. В первом случае, если он согласован, то осуществляется следующее действие — подписание договора. Во втором случае, когда по нему есть замечания, осуществляется его доработка.

Соединение типа **"или"**

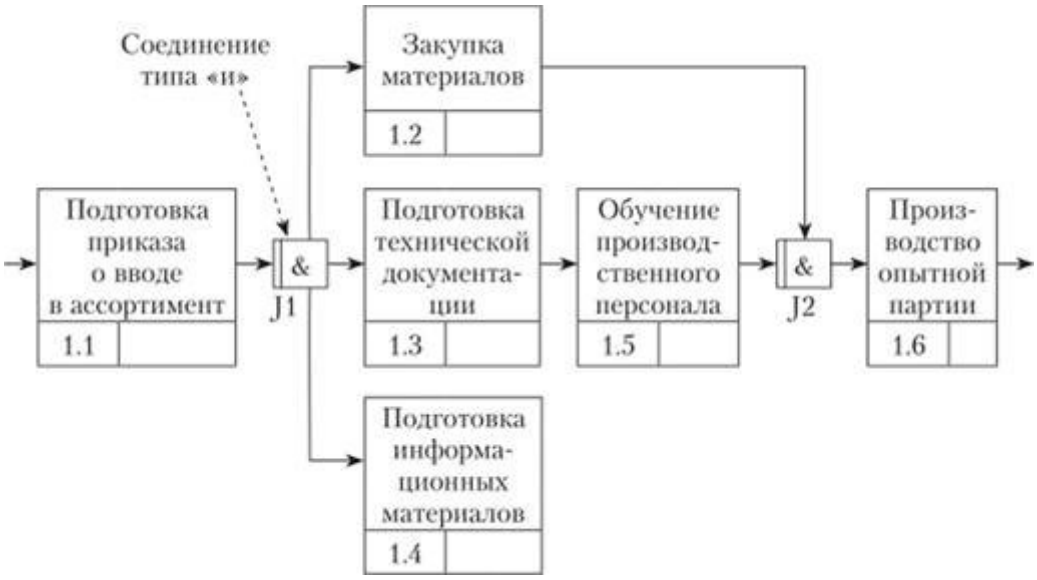


процесса "Заполнение анкеты"

пример использования соединения типа "или", где после действия "Заполнение поля “Ф.И.О.”" может быть выполнено действие "Заполнение поля “мобильный телефон”" или действие "Заполнение поля “домашний телефон”" либо оба эти действия. Одно из них точно должно быть выполнено.

Перекресток "Исключающий ИЛИ" является самым неопределенным, так как предполагает несколько возможных сценариев реализации бизнес-процесса и применяется для описания слабо формализованных ситуаций.

Соединение **типа «И»**



Следует учитывать, что если соединение "и" инициирует выполнение последнего действия, то все действия, которые присоединяются к сворачиваемому соединению типа "и" должны быть выполнены полностью.

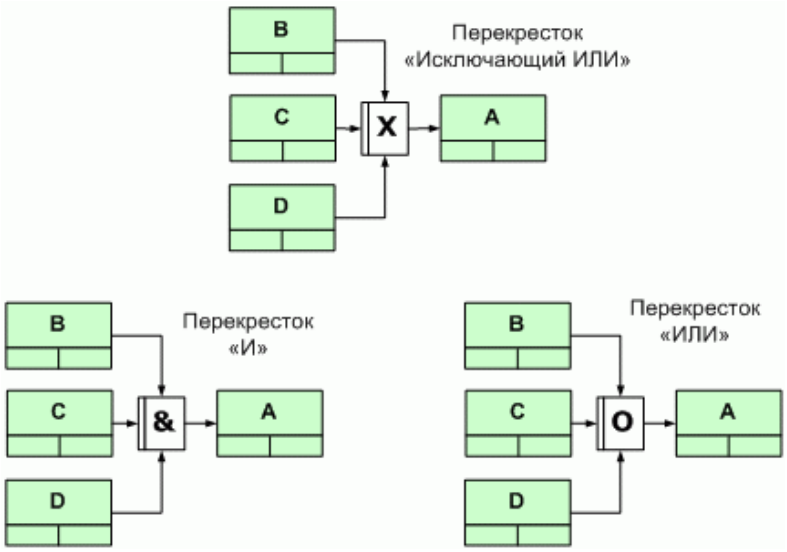
"Подготовка к продаже нового изделия" состоит из следующих подпроцессов:

- 1.1. Подготовка приказа о вводе в ассортимент нового продукта.
- 1.2. Закупка материалов для производства изделия.
- 1.3. Подготовка технической документации по изготовлению нового изделия.
- 1.4. Подготовка информационных материалов для продвижения и продажи.

- 1.5. Обучение производственного персонала изготовлению нового изделия.
- 1.6. Производство опытной партии нового изделия.

Процессы "Закупка материалов для производства изделия", "Подготовка информационных материалов для продвижения и продажи" и "Подготовка технической документации по изготовлению изделия" начинаются сразу после того, как выпущен приказ о вводе в ассортимент нового продукта. Процесс "Производство опытной партии нового изделия" может начаться только после того, как обучен производственный персонал и закуплен материал для производства.

Применение перекрестков "Исключающий ИЛИ", "И" и "ИЛИ" - схемы схождения.



Перекрестки "И" и "ИЛИ" подразделяются еще на два подтипа – *синхронные и асинхронные*.

Перекрестки синхронного типа обозначают, что работы B, C и D запускаются одновременно после завершения работы A.

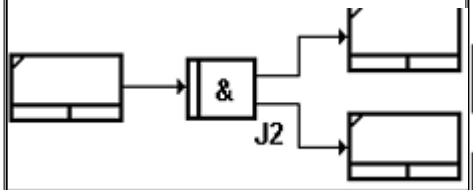
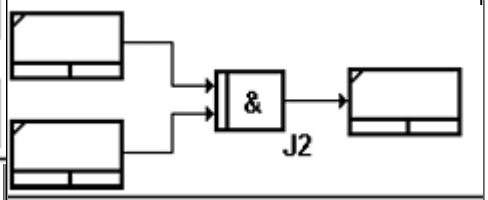
Перекрестки асинхронного типа требований к одновременности не предъявляют.

схемы взаимосвязи работ и перекрестков называются схемами расхождения, так как от перекрестков расходятся несколько работ. *Существует и другие схемы взаимосвязи перекрестков и работ* – это так называемые *схемы схождения*, когда к перекрестку подходит несколько работ

В таблице 4 приведены обозначения, названия и смысл всех типов перекрестков как в схемах схождения, так и в схемах расхождения.

Таблица 4. Обозначения, названия и смысл типов перекрестков в схемах схождения и расхождения.

Название перекрестков	**Обозначение перекрестков**	**Смысл перекрестков**	
		Схема расхождения	**Схема схождения**

"Исключающий ИЛИ"		X	Только одна последующая работа запускается	Только одна предшествующая работа должна быть завершена
"И"	**Асинхронный**	&	Все последующие работы запускаются 	Все предшествующие работы должны быть завершены 
	Синхронный	&	Все последующие работы запускаются одновременно	Все предшествующие работы должны быть завершены одновременно
"ИЛИ"	**Асинхронный**	O	Одна или несколько последующих работ запускаются	Одна или несколько предшествующих работ должны быть завершены
	Синхронный	O	Одна или несколько последующих работ запускаются одновременно	Одна или несколько предшествующих работ должны быть завершены одновременно

Последним отличием стандарта IDEF3 в отличие от классической методологии WFD является использование на схеме бизнес-процесса такого элемента как "объект ссылки", который связывается с работами и перекрестками. С помощью объектов ссылки показывается прочая важная информация, которую целесообразно зафиксировать при описании бизнес-процесса.

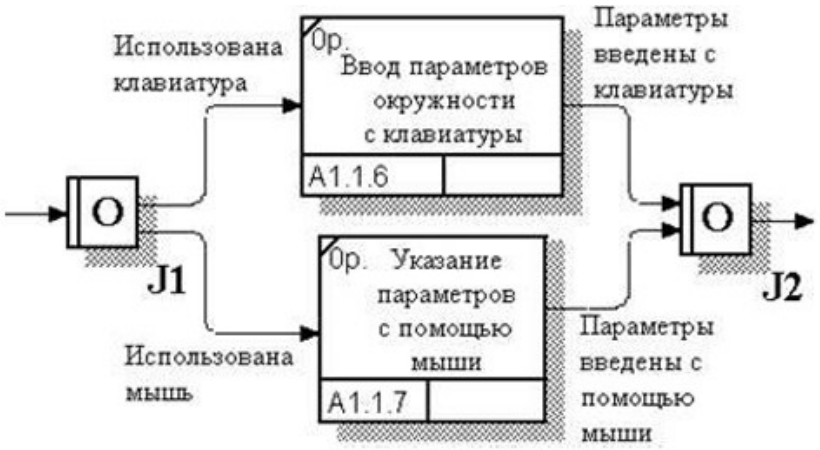
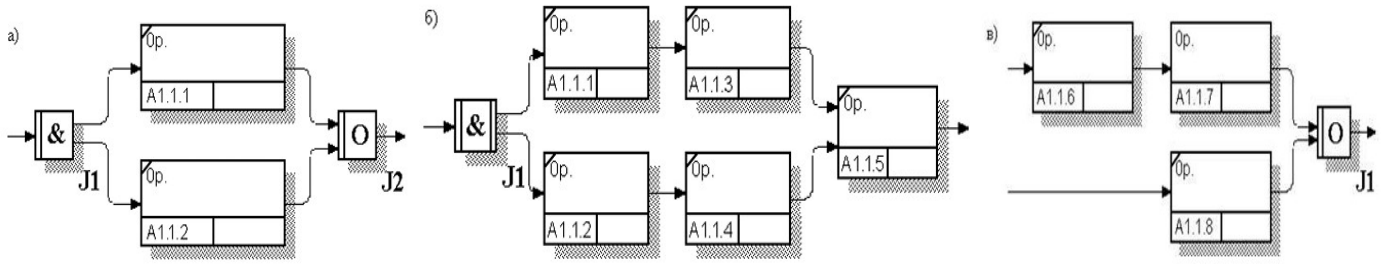


Рис. 9. Пример использования соединения "ИЛИ"

Все соединения на диаграммах должны быть парными, т.е. любое разворачивающее соединение

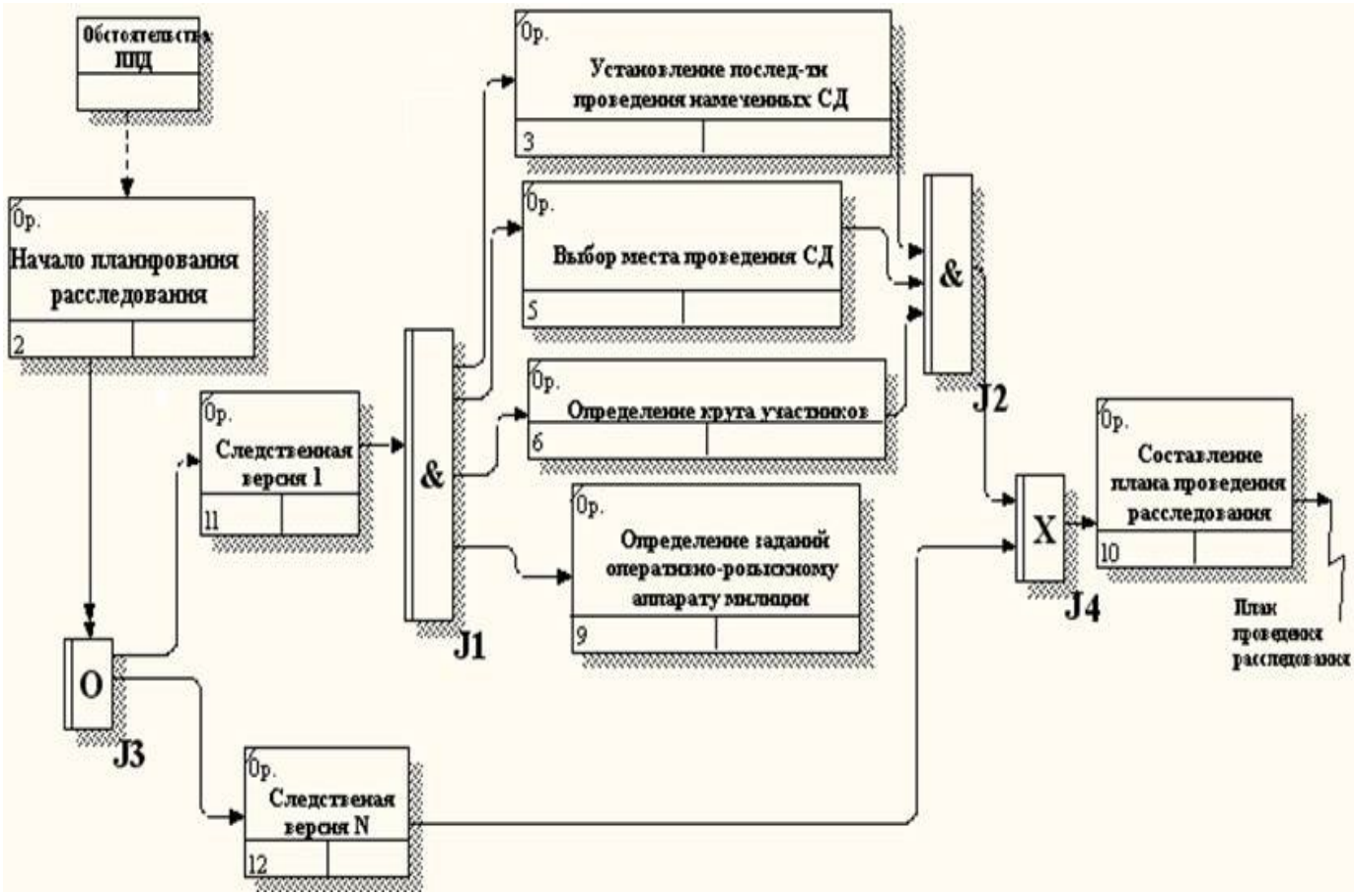
должно иметь парное себе сворачивающее соединение, хотя типы соединений не обязательно должны совпадать.

Соединения могут комбинироваться для создания более сложных ветвлений (рис. 11, 12). Комбинации соединений следует использовать с осторожностью, так как перегруженные ветвлением диаграммы сложны для восприятия.



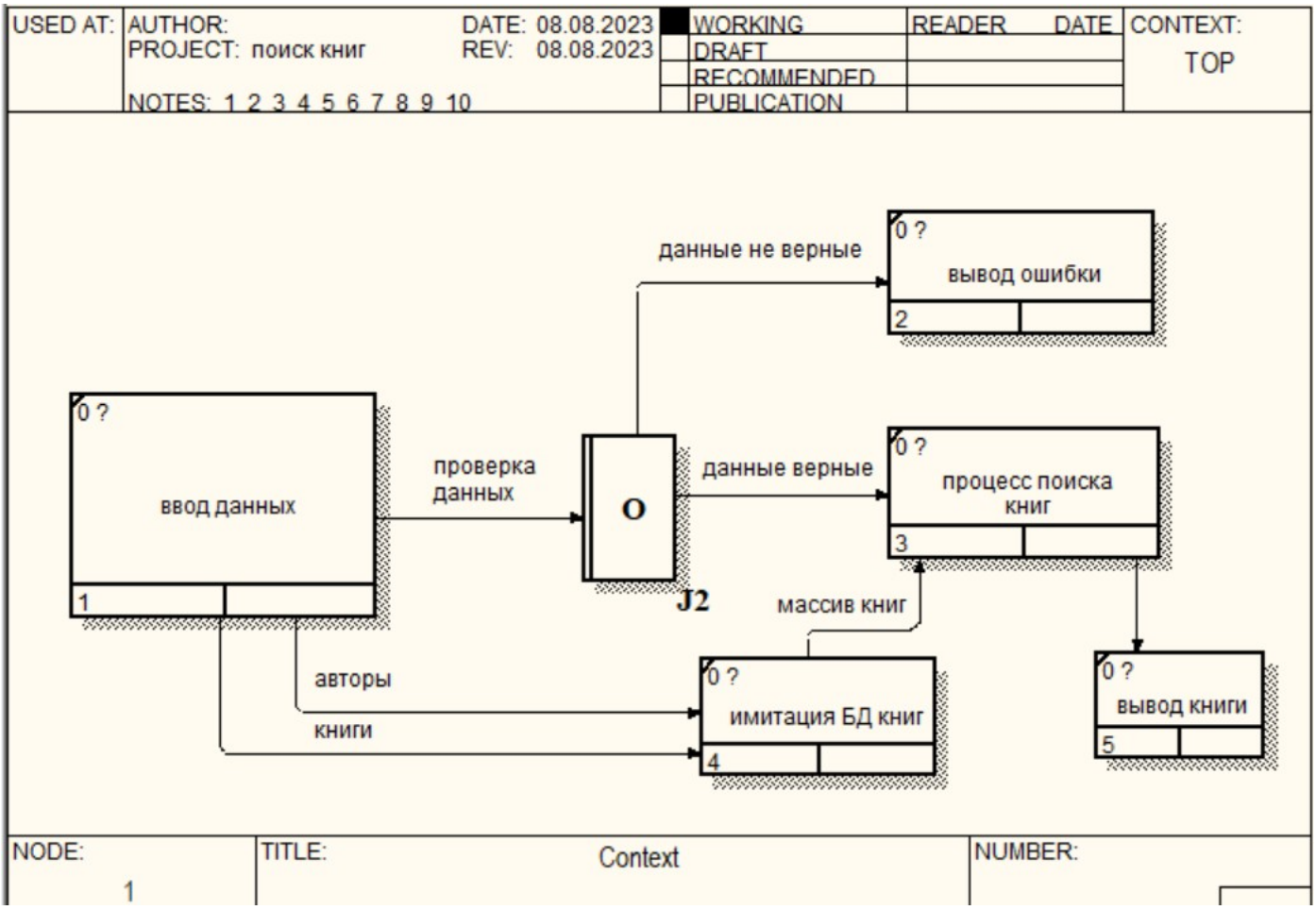
Логически выяснить взаимосвязь блоков и разделить их на под блоки модели.

Декомпозиция диаграмм



Блоки представляют функции системы (действия, процессы или операции), **а дуги - данные** (информацию или предметы).

Блок на диаграмме рассматриваемого уровня описывается более подробно блоками и дугами диаграммы более низкого уровня. Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются точно теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы.



ЛЕКЦИЯ 20

Тема: имитационное моделирование процессов. Принцип моделирования.

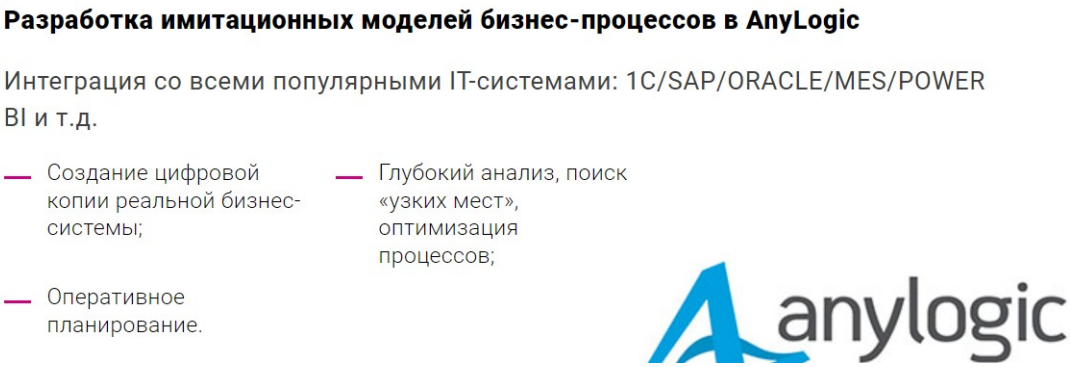
Использование имитационного моделирования для оптимизации бизнеса

Данный инструмент дает возможность определить, как изменения повлияют на компанию при эксперименте на модели, а не на «живой» фирме. Остановимся более подробно на инструкции.

В любой компании важно оптимизировать деятельность. Для одних руководителей понятие «оптимизация» сводится к приказу от вышестоящего начальства, для других является необходимостью, полноценным проектом, от итога которого будет зависеть дальнейшая перспектива компании.

Тем не менее, анализ и оптимизацию деятельности выполняют топ-менеджеры или консультанты, приглашенные с внешней стороны. С учетом того, что в качестве объекта оптимизации может выступать совершенно любой параметр (производимый продукт, операторы, принимающие звонки, общее количество обслуживаемых клиентов и пр),

идеи по улучшению также могут кардинально различаться — *от полной ликвидации отдела до внедрения нанотехнологий*. Понять, какая идея окажется выигрышной, довольно сложно, а проводить эксперименты на действующей реальной компании — непозволительно дорого.



Что такое имитационное моделирование

Имитационное моделирование — это способ исследования, который базируется на замене изучаемой системы на модель, имитирующую эту систему. Над установленной моделью проводят необходимы эксперименты и в итоге получают сведения о настоящей системе.

Имитационное моделирование активно применяется в проектах по реинжинирингу деятельности организаций, когда требуется заранее спрогнозировать результаты.

В первую очередь оптимизации подвергаются такие показатели, как:

- затраты процесса;

• длительность процесса;

• объем произведенного продукта или количество обслуженных клиентов.

Нерезультативные значения этих параметров снижают эффективность процессов, что в свою очередь приведет к потере финансового актива организации и недовольству руководителя.

Почему данные параметры выделяются как ключевые? Высокая цена бизнеса-процесса напрямую повышает затраты фирмы. Длительное выполнение процесса зачастую приводит к запоздалому получению результатов, когда они уже будут не актуальны. Недостаточное количество продукта организации — показатель даже не требует комментариев.

Метод имитационного моделирования дает возможность оценить как время выполнения процесса, так и временные промежутки, затрачиваемые на задержки в ходе его выполнения.

Например, оператор отлучился на перерыв или товарно-материальные ценности привезли с опозданием, Кроме этого, метод позволяет оценить непосредственно количество продукта, получаемое за конкретный интервал наблюдения.

Проведение имитационного моделирования подразумевает выполнение 4 ключевых этапов:

1. Построение модели процессов, осуществление которых нужно оптимизировать.
2. Старт имитации выполнения процессов утвержденной модели.
3. Анализ полученных данных.
4. Повторение трех вышеперечисленных пунктов для альтернативных сценариев осуществления процесса и выбор максимально подходящего.

Более подробней процесс выглядит так:

Имитационное моделирование дает возможность имитировать выполнение процесса таким образом, как он происходил бы в реальности, но в ускоренном режиме.

Чтобы смоделировать все разнообразия подобных ситуаций, следует учитывать некоторые факторы:

- в какой конкретно временной промежуток возникают события, приводящие к активации некоторой деятельности;
- актуальные графики работы персонала и оборудования (то есть, трудовых ресурсов компании);
- трудовой ресурс одномоментно может работать только над одной задачей;
- значения переменных или вероятности, на базе которых определяется следующий шаг;
- итоги одного вида деятельности могут повлиять на другой.

Прибавляя к данным факторам описание бизнес-процессов, мы получаем готовую модель, над которой свободно можно проводить эксперименты и получать ответы на важные вопросы.

Плюсы имитационного моделирования

Метод имитационного моделирования и ФСА также будут полезны, если нужно спроектировать новый бизнес. Как должна выглядеть организационная структура бизнеса? Как часто следует выполнять процессы? Какая длительность и стоимость процессов будет оптимальной?

- На все данные вопросы можно получить ответ ДО момента, когда фирма начнет свою деятельность и выяснит, что материальных ресурс все время не хватает, а трудовые — перегружены.
- на начальном этапе спроектировать эффективно работающий бизнес, то большинство проблем вообще не возникнут, что в итоге сэкономит много времени, которое можно направить на развитие молодой организации.

1.Имитационные модели позволяют анализировать системы и находить решения в тех случаях, когда такие методы, как аналитические вычисления и линейное программирование не справляются с задачей.

2. После того, как вы определитесь с уровнем абстракции, разрабатывать имитационную модель будет гораздо проще, чем аналитическую, поскольку процесс создания модели будет инкрементальным и модульным.

3. Структура имитационной модели естественным образом отображает структуру моделируемой системы.

4. Имитационная модель позволяет вам отслеживать все объекты системы, учтенные в выбранном уровне абстракции, добавлять метрики и проводить статистический анализ.

5. Одним из главных преимуществ имитационного моделирования является возможность проигрывать модель во времени и анимировать ее поведение. Анимация будет неоспоримым преимуществом при демонстрации модели и может оказаться полезной для верификации модели и нахождения ошибок.

6.Имитационные модели намного убедительнее электронных таблиц. Если вы используете имитационное моделирование, то при презентации проекта у вас будет явное преимущество перед теми, у кого на руках только цифры и решение, полученное из «черного ящика».

Недостатки имитационного моделирования

Однако наравне со многими преимуществами имитационного моделирования, существует и ряд недостатков.

- Стоит отметить момент, что для получения валидных результатов требуется работа по определению законов распределения случайных величин и по внесению всех сведений для проведения имитации.

- Также само по себе моделирование не отвечает на вопросы, результативно ли функционирует система, являются ли параметры показателей оптимальными и как провести перестройку бизнес-процесса. Для этого компании нужен бизнес-аналитик.

Но только при помощи механизмов имитационного моделирования и в некоторых случаях функционально-стоимостного анализа специалист может оперативно получить и обработать ценные сведения, которые абсолютно необходимы руководителю компании для принятия управленческих решений. При этом принять важные решения руководитель может не со слов консультанта, а на основании сравнения параметров ключевых показателей.