



Московский государственный технический
университет им. Н.Э. Баумана

Цикл лекций по дисциплине «Программная инженерия»

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук.

rtn@bmstu.ru
rtn.51@mail.ru

Москва – 2024



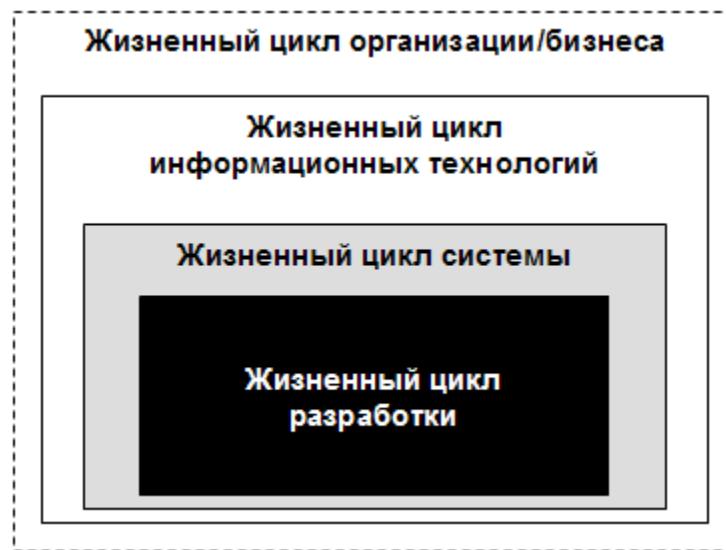
Лекция 2

Жизненный цикл и процессы разработки программного обеспечения

Жизненным циклом ПО называют весь период существования ПО, связанный с подготовкой к его разработке, разработкой, использованием и переработками, начиная с того момента, когда принимается решение о его разработке до того момента, когда полностью прекращается всякое ее использование.

Разработка программного обеспечения в контексте связанных дисциплин, практик, методов и специфики работы проектной команды.





Содержание четырех категорий жизненного цикла

При этом создаются и перерабатываются различного рода **артефакты — создаваемые человеком информационные сущности, документы в достаточно общем смысле, участвующие в качестве входных данных и получающиеся в качестве результата различных деятельности.**

Примерами артефактов являются:

- ❖ модель предметной области,
- ❖ описание требований,
- ❖ техническое задание,
- ❖ архитектура системы,
- ❖ проектная документация на систему в целом и на ее компоненты,
- ❖ прототипы системы и компонентов, исходные коды,
- ❖ пользовательская документация и прочее.

**ISO/IEC 12207 «Standard for Information Technology
Software Life Cycle Processes»(1995)**

**(российский аналог ГОСТ Р-1999. «Процессы
жизненного цикла программного обеспечения»)**

Определяет общую структуру
жизненного цикла ПО в виде 3-х
ступенчатой модели, состоящей из
процессов, видов деятельности и
задач.

Стандарт описывает вводимые элементы в терминах их целей и результатов.

При этом задаются неявно возможные взаимодействия между ними без определения четкой структуры взаимодействий.

Задаются метрики, по которым можно было бы отслеживать ход работ и их результативность.

Процессы жизненного цикла ПО

по ISO 12207

Основные процессы:

1. Приобретение ПО;
2. Поставка ПО (передача в использование);
3. Разработка ПО;
4. Эксплуатация ПО;
5. Сопровождение ПО

Поддерживающие процессы:

- Документирование;
- Управление конфигурациями;
- Обеспечение качества;
- Верификация;
- Валидация;
- Совместные экспертизы;
- Аудит;
- Разрешение проблем.

Организационные процессы:

- Управление проектом;
- Управление инфраструктурой;
- Усовершенствование процессов;
- Управление персоналом.

Адаптация:

Адаптация описываемых стандартом процессов под нужды конкретного проекта

Основные процессы жизненного цикла

1. Приобретение

Процесс приобретения (как его называют в ГОСТ – “заказа”) определяет работы и задачи заказчика, приобретающего программное обеспечение или услуги, связанные с ПО, на основе контрактных отношений.

Процесс приобретения состоит из следующих работ:

- **Initiation** – инициирование (подготовка);
- **Request-for-proposal preparation** – подготовка запроса на предложение (подготовка заявки на подряд);
- **Contract preparation and update** – подготовка и корректировка договора;
- **Supplier monitoring** – мониторинг поставщика (надзор за поставщиком);
- **Acceptance and completion** – приемка и завершение (приемка и закрытие договора).

Основные процессы.

2. Поставка ПО

Процесс поставки определяет работы и задачи поставщика. Процесс поставки включает следующие работы:

- 1. Initiation – инициирование (подготовка);**
- 2. Preparation of response – подготовка предложения (подготовка ответа);**
- 3. Contract – разработка контракта (подготовка договора);**
- 4. Planning – планирование;**

Основные процессы.

2. Поставка ПО

5. Execution and control - выполнение и контроль;
6. Review and evaluation – проверка и оценка;
7. Delivery and completion – поставка и завершение (поставка и закрытие договора).

Основные процессы.

3. Разработка ПО

Процесс разработки определяет работы и задачи разработчика. Процесс состоит из следующих работ:

- 1) Process implementation – определение процесса (подготовка процесса);
- 2) System requirements analysis – анализ системных требований (анализ требований к системе);

Основные процессы.

3. Разработка ПО

- 3) **System design** – проектирование системы (проектирование системной архитектуры);
- 4) **Software requirements analysis** – анализ программных требований (анализ требований к программным средствам);

Основные процессы.

3. Разработка ПО

5) Software architectural design –

проектирование программной архитектуры;

6) Software detailed design – детальное

проектирование программной системы (техническое проектирование программных средств);

Основные процессы.

3. Разработка ПО

- 7) **Software coding and testing** – кодирование и тестирование ПО;
- 8) **Software integration** – интеграция программной системы (сборка программных средств);
- 9) **Software qualification testing** – квалификационные испытания программных средств
- 10) **System integration** – интеграция системы в целом (сборка системы);

Основные процессы.

3. Разработка ПО

- 11) **System qualification testing** –
квалификационные испытания системы;
- 12) **Software installation** – установка
(ввод в действие);
- 13) **Software acceptance support** –
обеспечение приемки программных
средств.

Стандарт отмечает, что работы проводятся с использованием проектного подхода. Работы могут пересекаться по времени, то есть проводиться одновременно или с наложением, а также могут предполагать рекурсию и разбиение на итерации.

Основные процессы.

4. Эксплуатация ПО

Процесс эксплуатации определяет работы и задачи оператора службы поддержки.

Процесс включает следующие работы:

1. **Process implementation** – определение процесса (подготовка процесса);
2. **Operational testing** – операционное тестирование (эксплуатационные испытания);
3. **System operation** – эксплуатация системы;
4. **User support** – поддержка пользователя.

Основные процессы.
5. Сопровождение ПО

Процесс включает следующие работы:

- 1) **Process implementation** – определение процесса (подготовка процесса);
- 2) **Problem and modification analysis** – анализ проблем и изменений;
- 3) **Modification implementation** – внесение изменений;

Основные процессы.

5. Сопровождение ПО

4) Maintenance review/acceptance –

проверка и приемка при сопровождении;

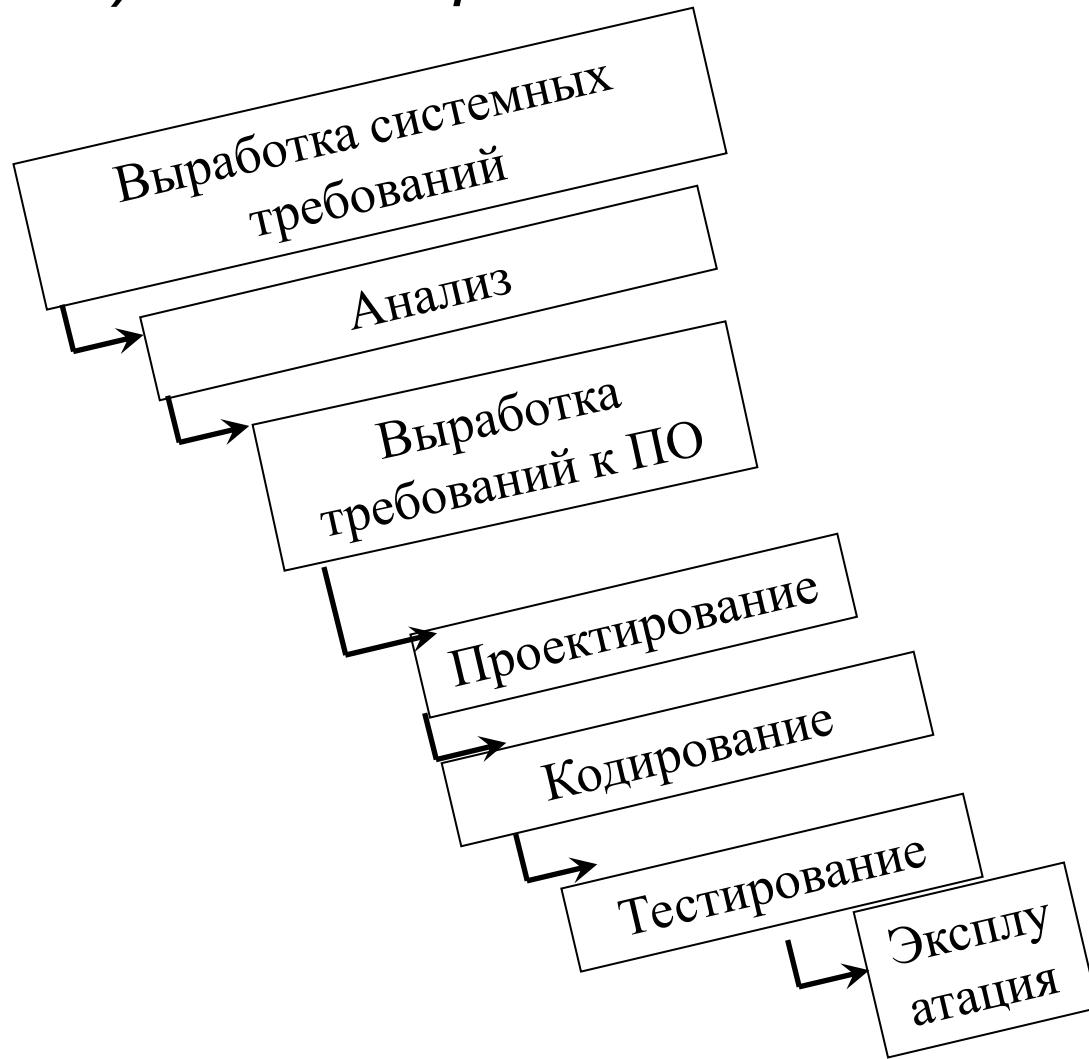
5) Migration – миграция (перенос);

6) Software retirement – вывод

программной системы из эксплуатации
(снятие с эксплуатации).

Модели жизненного цикла

Каскадная (водопадная) модель ЖЦ ПО

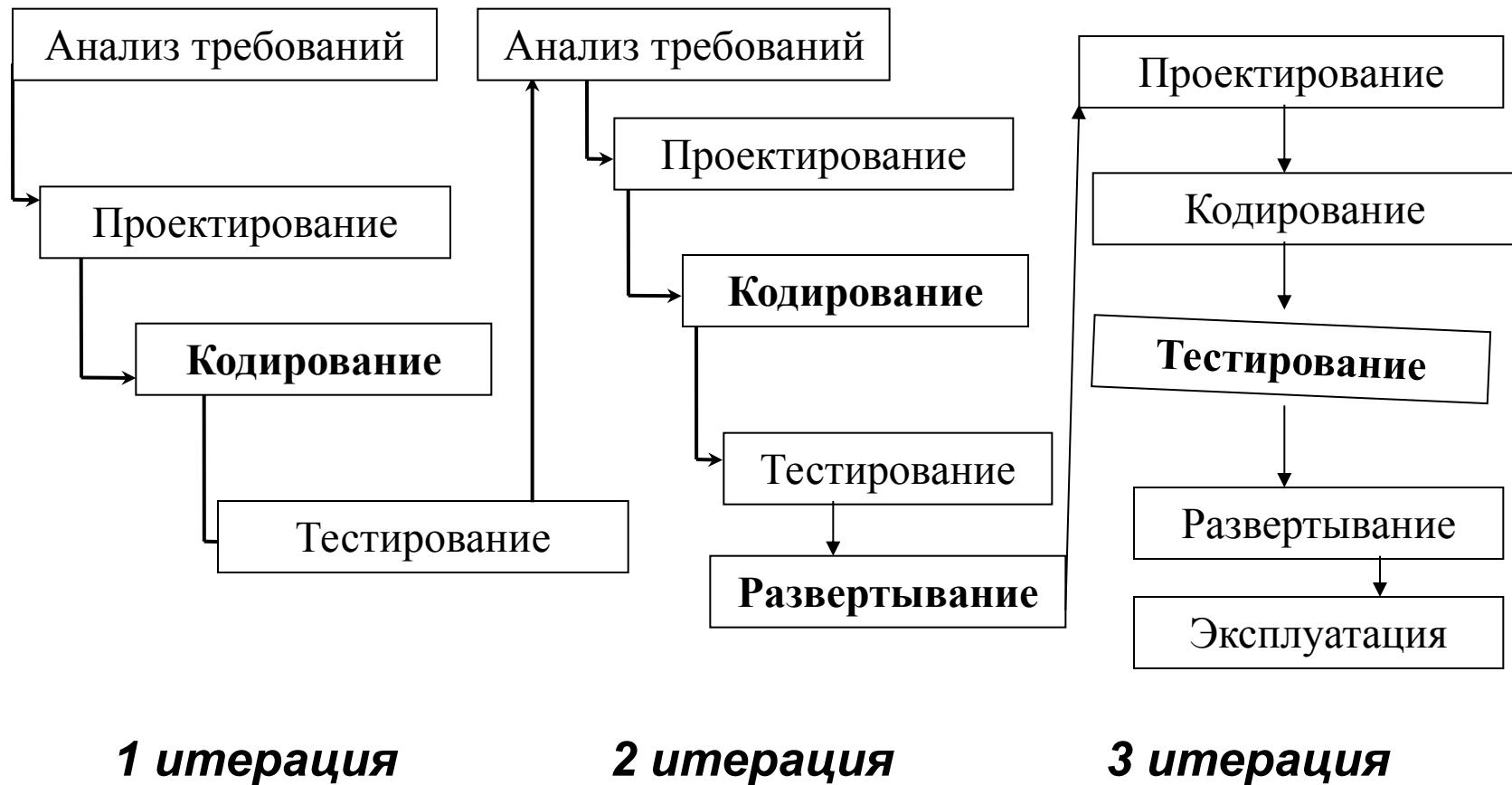


Водопадная схема включает несколько важных операций, применимых ко всем проектам:

- составление плана действий по разработке системы;
- планирование работ, связанных с каждым действием;
- применение операции отслеживания хода выполнения действий с контрольными этапами.

Итеративная и инкрементальная модель – эволюционный подход

Итеративная модель предполагает *разбиение
жизненного цикла проекта на
последовательность итераций, каждая
из которых напоминает «минипроект»*



Итеративная или инкрементальная модель

Сpirальная модель

Сpirальная модель была впервые сформулирована Барри Boehmом (Barry Boehm) в 1988 году.

Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла.

Боэм формулирует наиболее
распределенные (по приоритетам) риски:

- 1. Дефицит специалистов.**
- 2. Нереалистичные сроки и бюджет.**
- 3. Реализация несоответствующей
функциональности.**
- 4. Разработка неправильного
пользовательского интерфейса.**

- 5. “Золотая сервировка”, ненужная оптимизация и оттачивание деталей.**
- 6. Непрекращающийся поток изменений.**
- 7. Нехватка информации о внешних компонентах, определяющих окружение системы или вовлеченных в интеграцию.**

- 8. Недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами.**
- 9. Недостаточная производительность получаемой системы.**
- 10.“Разрыв” в квалификации специалистов разных областей знаний.**

Большая часть этих рисков связана с организационными и процессными аспектами взаимодействия специалистов в проектной команде.

Определение задач, альтернатив, ограничений

Совокупная стоимость

Ход работ

Анализ рисков

Оценка альтернатив, выделение рисков и способов борьбы с ними

Фиксация результатов

Экспертизы

Планирование следующих итераций



Разработка и верификация очередной части продукта

Сpirальная модель обладает рядом преимуществ:

1) Модель уделяет специальное внимание раннему анализу возможностей повторного использования. Это обеспечивается, в первую очередь, в процессе идентификации и оценки альтернатив.

2) Сpirальная модель предполагает возможность эволюции жизненного цикла, развитие и изменение программного продукта. Главные источники изменений заключены в целях, для достижения которых создается продукт.

Подход, предусматривающий скрытие информации о деталях на определенном уровне дизайна, позволяет рассматривать различные архитектурные альтернативы.

3) Спиральная модель предоставляет механизмы достижения необходимых параметров качества как составную часть процесса разработки программного продукта.

Эти механизмы строятся на основе идентификации всех типов целей (требований) и ограничений на всех “циклах” спирали разработки. Например, ограничения по безопасности могут рассматриваться как риски на этапе специфирования требований.

4) Сpirальная модель не проводит различий между разработкой нового продукта и расширением (или сопровождением) существующего.

Этот аспект позволяет избежать часто встречающегося отношения к поддержке и сопровождению как ко “второсортной” деятельности.

5) Спиралевидный подход, основанный на управлении рисками и возможности своевременного отбрасывания непривлекательных альтернатив (на ранних стадиях проекта) сокращает расходы и одинаково применим и к аппаратной части, и к программному обеспечению.



Лекция 2

Разработка технического задания

Стандарты на разработку ТЗ

1. ГОСТ 2.114-95. ЕСПД. Технические условия.
2. ГОСТ 19.201-78. ЕСПД. Техническое задание.
Требования к содержанию и оформлению.
3. ГОСТ 34.602-89. Информационная технология (ИТ).
Комплекс стандартов на автоматизированные системы.
Техническое задание на создание автоматизированной
системы.

«Техническое задание (ТЗ) – это исходный документ на проектирование технического объекта.

ТЗ устанавливает основное назначение разрабатываемого объекта, его технические и тактико-технические характеристики, показатели качества и технико-экономические требования, предписание по выполнению необходимых стадий создания документации (конструкторской, технологической, программной и т. д.) и её состав, а также специальные требования».

(Википедия).

Основные свойства требований в ТЗ:

1. Требование должно быть понятным.
2. Требование должно быть конкретным.
3. Требование должно быть тестируемым.

Если результат выполнения требования невозможно протестировать, значит, оно либо не понятное, либо не конкретное.

Именно во владении этими тремя свойствами требований и заключается мастерство и профессионализм!

- Должны ли быть описаны в нем спецификации различных функций, алгоритмы, типы данных и прочие технические штуки?
- А что такое техническое проектирование, о котором, кстати, сказано и в ГОСТах, и как оно связано с Техническим заданием?
- А где вообще граница между Техническим заданием и Техническим проектом?

В ответах на эти вопросы кроется очень коварная вещь. Именно поэтому часто возникают споры о достаточности или отсутствии необходимой детализации требований, о понятности документа Заказчиком и Исполнителями, об избыточности, формате представления и т.д.

- ❖ **Техническое задание** – это документ, в основе которого лежат требования, сформулированные на понятном (привычном) для Заказчика языке.
- ❖ При этом должна использоваться отраслевая терминология, понятная Заказчику.
- ❖ Никаких привязок к особенностям технической реализации быть не должно.
- ❖ На этапе ТЗ не важно, на какой платформе будут реализовываться эти требования.
- ❖ Однако, если речь идет о внедрении системы на основе уже существующего ПО, то такая привязка может иметь место, но только на уровне экранных форм, форм отчетов и прочего.
- ❖ Выяснением и формулированием требований, а также разработкой Технического задания должен заниматься бизнес-аналитик, а также возможно участие программиста.

Технический проект (ТП) – это документ, который предназначен для технической реализации требований, сформулированных в ТЗ.

В ТП (Рабочий проект) описываются структуры данных, триггеры и хранимые процедуры, алгоритмы и другое, что потребуются техническим специалистам. Заказчику в это вникать вовсе не обязательно!

ТП делает Архитектор системы (вот совмещение этой роли с программистом вполне нормально). Чем больше проект, тем больше людей работает над Техническим заданием.

По экспертным оценкам, стоимость затрат на разработку ТЗ может составлять 30-50%.

Такой разброс обусловлен различными платформами автоматизации, подходами и технологиями, применяемыми проектными командами при разработке.

Например, если речь идет о разработке на классическом языке типа С++, то без детального технического проектирования тут не обойтись. Если речь идет о внедрении системы на платформе 1С, то тут с проектированием ситуация несколько иная.

ГОСТ рекомендует следующие разделы:

- общие сведения;
- назначение и цели создания (развития) системы;
- характеристика объектов автоматизации;
- требования к системе;
- состав и содержание работ по созданию системы;
- порядок контроля и приемки системы;
- требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;
- требования к документированию;
- источники разработки.

Раздел 1. Общие сведения.

Рекомендации по ГОСТ: полное наименование системы и ее условное обозначение.

Пишем, как будет называться система, ее краткое наименование, шифр темы или шифр (номер) договора. (Можно вообще удалить этот раздел).

Полезная информация: перечень документов, на основании которых создается система, кем и когда утверждены эти документы.

Тут стоит указать ту нормативно-справочную документацию, которую Вам предоставили для ознакомления с определенной частью требований
плановые сроки начала и окончания работы по созданию системы.

Раздел 2. Назначение и цели создания (развития) системы.

Назначение системы следует формулировать четко.

Если написать что-то вроде *«качественно автоматизировать складской учет в компании X»*, то потом можно долго обсуждать результат при его завершении что такое качественно!!!

Лучше сразу написать примерно так:

«Система предназначена для ведения складского учета в компании X в соответствии с требованиями, зафиксированными в данном ТЗ».

Цели создания системы

Пример неудачной цели: «Обеспечить быстрое оформление документов менеджером». Что такое быстрое? Это можно потом доказывать бесконечно.

Лучше переформулировать данную цель так:
«Менеджер по продажам должен иметь возможность оформить документ «Реализация товаров» из 100 строк за 10 минут».

Если в настоящее время менеджер тратит на это около часа, что слишком много для этой компании и для них это важно.

Раздел 3. Характеристика объектов автоматизации.

Рекомендации по ГОСТ: краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию.

- Можно привести ссылки на документы, которые полезно изучить составу проектной команды для погружения в вопрос: отраслевые особенности,
- сведения об условиях эксплуатации объекта автоматизации,
- характеристиках окружающей среды.

Раздел 4. Требования к системе

Рекомендации по ГОСТ: требования к системе в целом:

- требования к структуре и функционированию системы;
- требования к численности и квалификации персонала системы
- режиму работы системы и персонала;
- характеристики качества системы;
- требования к надежности;
- требования безопасности;
- требования к эргономике и технической эстетике;
- требования к транспортабельности для подвижных АС;

- требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;
- требования к защите информации от несанкционированного доступа;
- требования по сохранности информации при авариях;
- требования к защите от влияния внешних воздействий;
- требования к патентной чистоте;
- требования по стандартизации и унификации.

Требования к квалификации.

Возможно, разрабатываемая система потребует переподготовки специалистов. Это могут быть как пользователи будущей системы, так и ИТ-специалисты, которые будут нужны для ее поддержки.

Недостаточное внимание к данному вопросу нередко перерастает в проблемы.

Если квалификация имеющегося персонала явно недостаточна, лучше прописать требования к организации обучения, программе обучения, срокам и т.п.

Требования к защите информации от несанкционированного доступа.

Это как раз и есть требования к разграничению доступа к данным.

Если такие требования планируются, то их нужно расписать отдельно, как можно более детально по тем же правилам, что и функциональные требования (понятность, конкретность, тестируемость).

Поэтому, можно эти требования включить и в раздел с функциональными требованиями.

Требования к стандартизации.

Если существуют какие-либо стандарты разработки, которые применимы к проекту, они могут быть включены в требования. Как правило, такие требования инициирует ИТ-служба Заказчика.

Например, у компании 1С есть требования к оформлению программного кода, проектированию интерфейса.

Требования к структуре и функционированию системы.

Тут могут быть описаны требования к интеграции систем между собой, представлено описание общей архитектуры. Чаще требования к интеграции выделяют вообще в отдельный раздел или даже отдельное Техническое задание, т.к. эти требования могут оказаться достаточно сложными.

Требования к видам обеспечения:

- ❖ Математическое
- ❖ Информационное
- ❖ Лингвистическое
- ❖ Программное
- ❖ Техническое
- ❖ Метрологическое
- ❖ Организационное
- ❖ Методическое

Когда стоит описывать данные требования:

- Решения о том, на каком языке (или какой платформе) будет вестись разработка не принято.
- К системе предъявляются требования мультиязычного интерфейса (например, русский/английский).
- Для функционирования системы должно быть создано отдельное подразделения или приняты на работу новые сотрудники.
- Для функционирования системы у Заказчика должны произойти изменения в методиках работы и эти изменения должны быть конкретизированы и запланированы.
- Предполагается интеграция с каким-либо оборудованием и к нему предъявляются требования (например, сертификации, совместимости и пр.)
- Возможны другие ситуации.

Раздел 5. Состав и содержание работ

по созданию системы

Перечень стадий и этапов работ по созданию системы в соответствии с ГОСТ 24.601:

- сроки их выполнения,
- перечень организаций — исполнителей работ,
- ссылки на документы, подтверждающие согласие этих организаций на участие в создании системы,
- запись, определяющую ответственного (заказчик или разработчик) за проведение этих работ.

Другими словами, это план разработки системы, описание этапов разработки, возможность привлечения подрядчиков и т.п.

Раздел 6. Порядок контроля и приемки системы

- Виды, состав, объем и методы испытаний системы и ее составных частей (виды испытаний в соответствии с действующими нормами ГОСТ).
- Общие требования к приемке работ по стадиям (перечень участвующих предприятий и организаций, место и сроки проведения).
- Порядок согласования и утверждения приемочной документации.
- Но даже наличие тестируемых требований может оказаться недостаточно при сдаче системы, если четко не прописан порядок приемки-передачи работ.

Раздел 7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

- Приведение поступающей в систему информации (в соответствии с требованиями к информационному и лингвистическому обеспечению) к виду, пригодному для обработки с помощью ЭВМ.

К примеру, для функционирования системы может потребоваться использование каких-либо отраслевых или общероссийских справочников и классификаторов. Эти справочники должны каким-то образом появляться в системе, обновляться и правильно использоваться.

- Создание условий функционирования объекта автоматизации, при которых гарантируется соответствие создаваемой системы требованиям, содержащимся в ТЗ (ГОСТ).

Здесь следует прописать все изменения, которые могут потребоваться в организационной структуре предприятия для успешного функционирования разработанной системы.

Например:

- ❖ *в компании отсутствует локальная сеть, устаревший парк ЭВМ, на которых система не заработает.*

- ❖ Возможно, какая-то необходимая информация обрабатывалась на бумаге, а теперь ее необходимо вводить в систему.
- ❖ Возможно, что-то упрощалось, а теперь требуется учитывать более детально, соответственно кто-то должен собирать информацию по определенным правилам.

Раздел 8. Требования к документированию

Описать согласованный разработчиком и Заказчиком системы перечень подлежащих разработке комплексов и видов документов.

Игнорирование требований к документации очень часто приводит к самым неожиданным последствиям на проектах. Например, все сделано и все работает. Пользователи тоже умеют работать. Про документацию вообще не договаривались и не разговаривали. И вдруг при сдаче работ кто-то из топ-менеджеров Заказчика, Вас спрашивает: «А где руководства пользователя?» И все, не хочет принимать у Вас работу. За чей счет будете разрабатывать руководства?

Раздел 9. Источники разработки

Должны быть перечислены документы и информационные материалы (технико-экономическое обоснование, отчеты о законченных научно-исследовательских работах, информационные материалы на отечественные, зарубежные системы-аналоги и др.), на основании которых разрабатывалось ТЗ и которые должны быть использованы при создании системы.



Московский государственный технический
университет им. Н.Э. Баумана

Цикл лекций по дисциплине «Программная инженерия»

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук.

rtn@bmstu.ru
rtn.51@mail.ru

Москва – 2024



Лекция 3

Анализ предметной области и проектирование программного обеспечения



Решение принято,
проекту быть!



Провели совещание с
руководителями, собрали
некоторую информацию об их
видении результата



Обучение проектной команды
методикам и инструментам работы,
согласование правил работы, видов
и состава документации



Формирование рабочей
группы от Заказчика и
Исполнителя и
распределение ролей



Анкетирование



Опросы



Выделение ключевых
бизнес-процессов или
областей
автоматизации



Что дальше?



Формулирование ключевых
требований к системе, целей,
критериев успешности проекта,
процессов для детального изучения

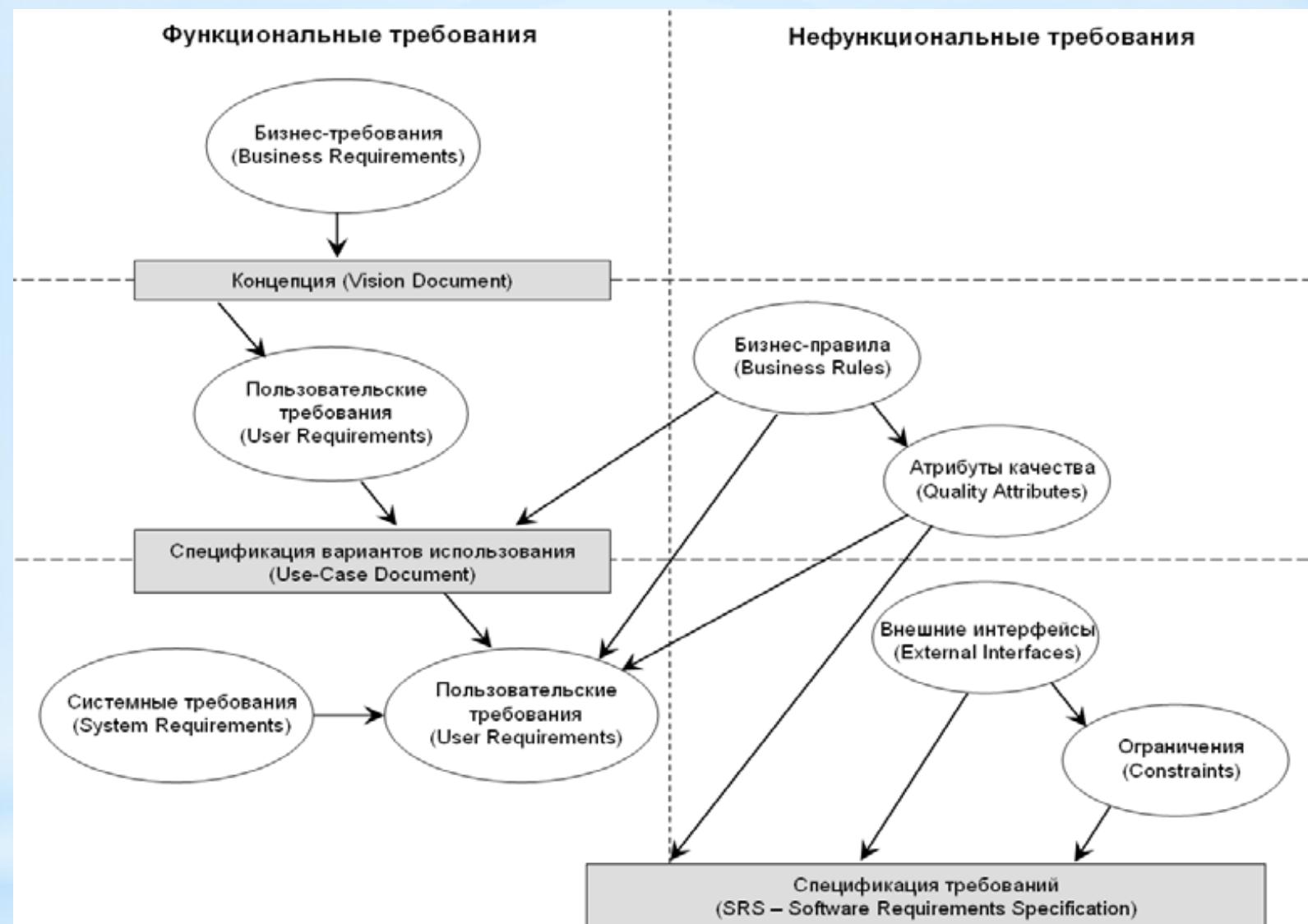
Сбор требований для автоматизации или описания бизнес-процессов

Анализ и проектирование

Программные требования (Software Requirements)

Программные требования – *Software Requirements* – свойства, которые должны быть надлежащим образом представлены для решения конкретных практических задач. Данная область знаний касается вопросов извлечения (сбора), анализа, специфирования и утверждения требований.

Классический пример высокоуровневого структурирования групп требований как требований к продукту описан в работах одного из классиков дисциплины управления требованиями – Карла Вигерса.



Уровни требований по Вигерсу

Функциональные требования должны указывать что информационная система должна делать в процессе функционирования. Функциональные требования представляют в виде вариантов использования, Use Case.

Нефункциональные требования должны указывать с соблюдением каких условий должна функционировать информационная система. Например, скорость отклика при выполнении заданной операции.

Группа функциональных требований

1. **Бизнес-требования (Business Requirements)** – определяют высокоуровневые цели организации или клиента (потребителя) – заказчика разрабатываемого программного обеспечения.
2. **Пользовательские требования (User Requirements)** – описывают цели/задачи пользователей системы, которые должны достигаться/выполняться пользователями при помощи создаваемой программной системы.
3. **Функциональные требования (Functional Requirements)** – определяют функциональность (поведение) программной системы, которая должна быть создана разработчиками для предоставления возможности выполнения пользователями своих обязанностей в рамках бизнес-требований и в контексте пользовательских требований.

Группа нефункциональных требований (Non-Functional Requirements)

Бизнес-правила (Business Rules) – связаны с политиками, корпоративными регламентами, стандартами, законодательными актами, внутрикорпоративными инициативами .

Бизнес-правила можно трактовать как положения, которые определяют или ограничивают некоторые аспекты бизнеса.

Они подразумевают организацию структуры бизнеса, контролируют или влияют на поведение бизнеса. Бизнес-правила часто отвечают за распределение ответственности сотрудников при работе с информационной системой.

Например, методология RUP выделяет отдельный артефакт Business Rule в рамках дисциплины Business Modeling.

*Все бизнес-правила, в рамках данной дисциплины, идентифицируются и описываются в документе *Business Rules Document*.*

*При разработке требований, в сценариях *Use Cases* обычно включается ссылка на уже описанное бизнес-правило.*

В настоящее время разработаны методы и

подходы формального представления

бизнес-правил, вплоть до формальных

языков описания:

Object Constraint Language (OCL);

Business Rules Markup Language (BRML).

**Бизнес-правила могут быть не только
использованы при определении требований к
разрабатываемому ПО, но и могут отдельно
оформляться в дизайне ПО. Существуют
специализированные инструментальные
средства и библиотеки, позволяющие
разрабатывать и поддерживать приложения,
интенсивно использующие бизнес-правила.**

Внешние интерфейсы

Конкретизация аспектов взаимодействия с другими системами, операционной средой (например, запись в журнал событий операционной системы), возможностями мониторинга при эксплуатации

*– все это **не функциональные требования. Это вопросы интерфейсов!!!***

Функциональные требования связаны непосредственно с функциональностью системы, направленной на решение бизнес- потребностей.

Атрибуты качества (*Quality Attributes*) –

Атрибуты качества ПО касаются вопросов портируемости, интероперабельности (прозрачности взаимодействия с другими системами), целостности, надежности, отказоустойчивости.

Ограничения (Constraints) – формулировки условий, модифицирующих требования или наборы требований, сужая выбор возможных решений по их реализации.

К ним могут относиться:

- параметры производительности, влияющие на выбор платформы реализации и/или развертывания (протоколы, серверы приложений, баз данных),
- выбор языка программирования по требованию заказчика и другое.

Системные требования (System Requirements)

Здесь описывают высокоуровневые требования к программному обеспечению, содержащему несколько взаимосвязанных подсистем и приложений. При этом, система может быть как целиком программной, так и состоять из программной и аппаратной частей. В общем случае, частью системы может быть персонал, выполняющий определенные авторизованные функции системы.

Независимые свойства (Emergent Properties) –

*требования, которые не могут быть
адресованы тому или иному компоненты
программной системы, но которые должны
быть соблюдены, например, в контексте
сетевой инфраструктуры или регламентов
работы пользователей.*

Требования с количественной оценкой – это требования, поддающиеся количественному определению (измерению).

Например, система должна обеспечить пропускную способность “столько-то запросов в секунду”.

Формулирование требования в форме “система должна обеспечить рост пропускной способности” без указания конкретных количественных характеристик является просто некорректно определенным требованием!

Большинство требований с количественной оценкой относится к атрибутам качества.

В качестве примера можно привести реальное требование, присущее в реальном проекте по электронному документообороту:

“Система должна производить поиск документов <определенного вида> за время, не превышающее 5 секунд”.

Подходы к выявлению требований к ПО:

1. **Интервьюирование** – традиционный подход извлечения требований. Составляется список вопросов к заказчику ПО. На основе этих данных системные аналитики и программисты формулируют требования к разрабатываемой системе.
2. **Прототипы** – отличный инструмент для уточнения и/или детализации требований.

3. Разъясняющие встречи - термин, пришедший из общей практики менеджмента и базирующийся на идеях сотрудничества заинтересованных лиц для совместного анализа путей решения проблем, определения и предупреждения рисков и т.п.

4. Наблюдение (observation) – подразумевает непосредственное присутствие аналитиков и инженеров рядом с пользователем в процессе выполнения последним его работ по обеспечению функционирования бизнес-процессов.

Анализ требований (Requirements Analysis)

- Обнаружение и разрешение конфликтов между требованиями;
- Определение границ задачи, решаемой создаваемым программным обеспечением;
- Детализация системных требований для установления программных требований.

Результатом анализа требований должны быть однозначно интерпретируемые требования, реализация которых проверяется, а стоимость и ресурсы – предсказуемы.

Архитектурное проектирование очень
близко к концептуальному моделированию.
В принципе, можно говорить о том, что
деятельность по моделированию в большей
степени касается того, "что" надо сделать, а
архитектурное проектирование - "как" это
будет реализовано.

Выбор архитектуры ПО

Архитектурой ПО называют совокупность базовых концепций (принципов) построения программного обеспечения.

Архитектура ПО определяется сложностью решаемых задач, степенью универсальности разрабатываемого ПО и числом пользователей, одновременно работающих с одной его копией.

Однопользовательские архитектуры:

- ❖ программы;
- ❖ пакеты программ ;
- ❖ программные комплексы ;
- ❖ программные системы .

Многопользовательские архитектуры

рассчитаны на работу в локальной или глобальной сети. Такие архитектуры реализуют системы, построенные по принципу «клиент-сервер».

Пакеты программ – совокупность программ, решающих задачи некоторой прикладной области.

- Например, пакет графических программ, пакет математических программ.
- По сути это библиотека программ, которые принадлежат к одной прикладной области.
- Каждая программа сама вводит необходимые данные и выводит результаты.

Программные комплексы – совокупность программ, совместно обеспечивающих решение небольшого класса сложных задач одной прикладной области.

Вызов программ в программном комплексе осуществляется специальной *программой-диспетчером*, которая обеспечивает несложный интерфейс с пользователем.

От пакета программ отличается еще и тем, что для решения одной задачи несколько программ могут вызываться последовательно или циклически.

Данные хранятся в ОП или файлах.

Программные системы – совокупность программ (подсистем), позволяющая решать широкий класс задач из некоторой прикладной области.

Программы, входящие в программную систему взаимодействуют через ***общие данные*** и имеют развитый пользовательский и внутренний интерфейсы.

Такие системы требуют тщательного проектирования.

Проектирование архитектуры системы – определение состава необходимого оборудования, программного обеспечения и операций, выполняемых обслуживающим персоналом.

Стадии разработки:

- *Постановка задачи* (формулируют назначение ПО и основные требования к нему. Стадия «Техническое задание»).
- *Анализ требований и разработка спецификаций* (стадия «Эскизный проект»).

- **Проектирование** (стадия «Технический проект»).
- **Реализация** (стадия «Рабочий проект»).
- **Этап сопровождения** сейчас рассматривается отдельно.

Эскизный проект призван помочь в выборе оптимального решения при наличие альтернатив. Этот этап может быть пропущен либо его осуществляет научная организация, если приходится проводить предварительное моделирование будущей сложной системы.

АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ ПО ПРИ СТРУКТУРНОМ ПОДХОДЕ

Собственно разработка любого программного обеспечения начинается с анализа требований к будущему программному продукту.

В результате анализа получают спецификации разрабатываемого ПО, выполняют декомпозицию и содержательную постановку решаемых задач, уточняют их взаимодействия и их эксплуатационные ограничения.

Строят общую модель предметной области.

Спецификации программного обеспечения

при структурном подходе

- ❖ **Спецификации** представляют собой полное и точное описание функций и ограничений разрабатываемого программного обеспечения.
- ❖ Спецификации функциональные описывают функции разрабатываемого программного обеспечения,
- ❖ Спецификации эксплуатационные определяет требования к техническим средствам, к надежности, информационной безопасности.

Основные требования к функциональным спецификациям:

1. Требование *полноты* означает, что спецификации должны содержать всю существенную информацию, где ничего важного не было бы упущено и отсутствует несущественная информация, например детали реализации, чтобы не препятствовать разработчику в выборе наиболее эффективных решений;

2. Требование *точности* означает, что спецификации должны однозначно восприниматься как заказчиком так и разработчиком.

Требование 2 выполнить достаточно сложно в силу того, что естественный язык для описания спецификаций не подходит: даже подробные спецификации на естественном языке не обеспечивают необходимой точности. Точные спецификации можно определить, только разработав некоторую *формальную модель* разрабатываемого программного обеспечения.

Формальные модели, используемые на этапе определения спецификаций можно разделить на две группы:

- ❖ модели, зависящие от подхода к разработке (структурного или объектно-ориентированного),
- ❖ модели, не зависящие от него.

При этом *диаграммы переходов состояний*, которые демонстрируют особенности поведения разрабатываемого ПО при получении тех или иных сигналов извне, и математические модели предметной области используют при любом подходе к разработке.

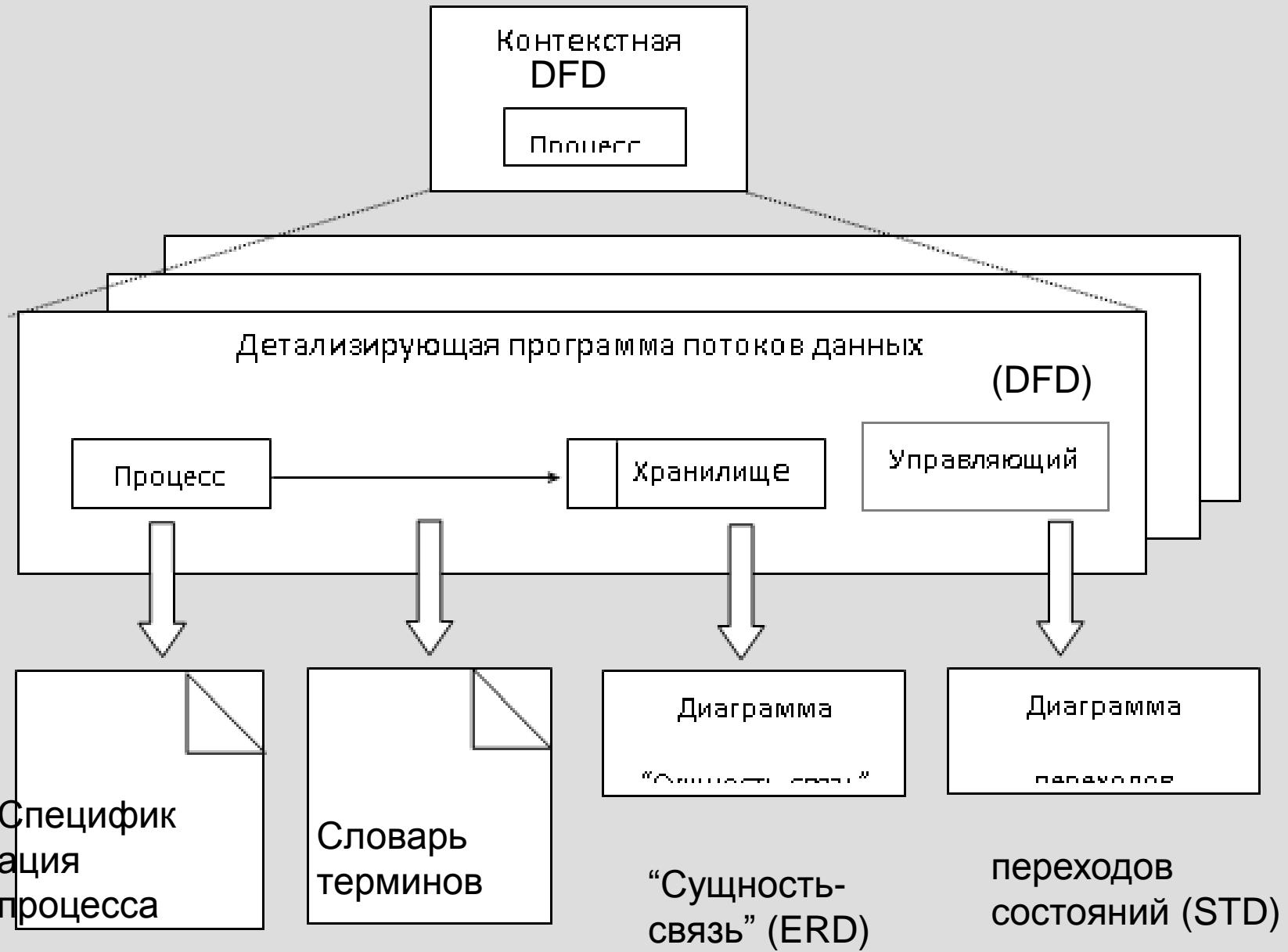


Классификация моделей разрабатываемого ПО ,
используемых на этапе определения спецификаций

Методологии *структурного анализа и проектирования*, основанные на моделировании потоков данных, обычно используют комплексное представление программного обеспечения в виде совокупности моделей:

- **диаграмм потоков данных (DFD - Data Flow Diagrams),** описывающих взаимодействие источников и потребителей информации через процессы, которые должны быть реализованы в системе;

- диаграмм “сущность-связь” (**ERD - Entity Relationship Diagrams**), описывающих базы данных разрабатываемой системы;
- диаграмм переходов состояний (**STD - State Transition Diagrams**), характеризующих поведение системы во времени;
- спецификаций процессов;
- словаря терминов.



Спецификации процессов обычно представляют в виде краткого текстового описания, схем алгоритмов, псевдокодов, Flow-форм или диаграмм Насси-Шнейдермана. Поскольку описание процесса должно быть кратким и понятным, как разработчику, так и заказчику, для их спецификации чаще всего используют псевдокоды.

Выбор парадигмы программирования

Парадигма (*парáдигуma* -пример, модель, образец)-
**совокупность фундаментальных научных
установок, представлений и терминов,
принимаемая и разделяемая научным
сообществом и объединяющая большинство
его членов. Обеспечивает преемственность
развития науки и научного творчества.**

Парадигма программирования - это совокупность
идей и понятий, определяющих стиль написания
компьютерных программ.

Важно отметить, что парадигма программирования
не определяется однозначно языком
программирования. Практически все современные
языки программирования в той или иной мере
допускают использование различных парадигм.

ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ

1. Императивная

- Процедурная
- Структурная
- Модульная
- Аспектно-ориентированная
- Объектно-ориентированная
- Субъектно-ориентированная
- Агентно-ориентированная
- Компонентно-ориентированная
- Прототипно-ориентированная
- Обобщённое программирование

2. Конкатенативная

3. Декларативная

- Функциональная
- Аппликативная
- Комбинаторная
- Основанная на продолжениях
- В терминах Рефал-машины
- Логическая
- Ограничениями
- Потоком данных

4. Метапрограммирование

- Языково-ориентированная
- Пользовательская
- Автоматизация процесса программирования
- Рефлексивное

5. Параллельная

6. Событийно-ориентированная

- Реактивная

7. Сервис-ориентированная

- Автоматная

Событийно-ориентированное программирование

(англ. *event-driven programming*)

Это парадигма программирования, в которой выполнение программы определяется событиями:

- ❖ действиями пользователя (клавиатура, мышь),
- ❖ сообщениями других программ и потоков,
- ❖ событиями операционной системы (например, поступлением сетевого пакета).

Объектно-ориентированное программирование
(ООП)-парадигма программирования, в которой
основными концепциями являются понятия
классов и объектов.

Субъектно-ориентированное
программирование (*subject - oriented programming*)
— метод построения объектно-ориентированных
систем, как композиции субъектов.

Субъектом называется приложение, способное в рамках системы самостоятельно реализовывать задачу, имеющую несколько путей решения.

В отличие от "неразумного" Объекта, Субъект наделен возможностью выбирать этот путь, то есть он способен сам корректировать последовательности своих действий для достижения поставленной цели.

СОП дополняет ООП, решая проблемы, возникающие при разработке больших систем, при решении задач интеграции и переносимости.

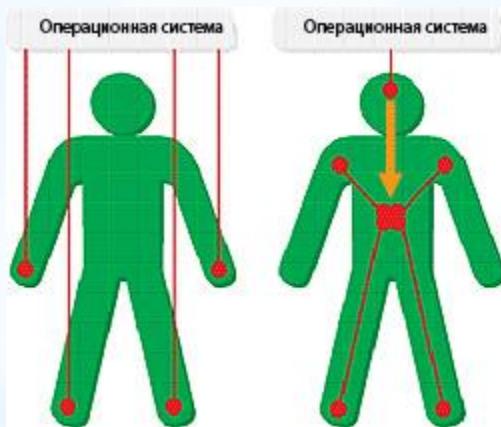
Использование субъектного подхода требует:

- ❖ разбиения системы на субъекты;
- ❖ написания правил для их правильной композиции.

Стратегия управления такими приложениями должна основываться не на четких и конкретных командах операционной системы, а на «инструкциях» (заданных правилах).

Управляя *Объектом*, мы используем отдельные его методы, а *Субъекту* достаточно указать "номер" инструкции, на основании которой он функционирует. И он уже сам будет управлять своими методами, чтобы достичь результата.

Управление Объектом и Субъектом



Для эффективной разработки сложных интеллектуальных систем необходимо опираться на элементы, обладающие достаточной самостоятельностью.

На самом деле платформа для этого подхода в архитектуре систем подготавливалась на протяжении последних десяти-пятнадцати лет.

Как же должен быть устроен Субъект, чтобы реализовать все эти возможности?

В первую очередь нужно формализовать на уровне системы существенные особенности такого элемента:

1. Субъект живет самостоятельной жизнью.

Его сердцем является таймер, который как бы "питает"

Субъект машинным временем операционной системы.

2. Субъект имеет органы осязания в виде сенсоров.

Сенсоры следят за состоянием внешней и внутренней среды **Субъекта**. На основании анализа этих состояний

Субъект выбирает ту или иную линию поведения.

3. Субъект содержит набор линий поведения.

Каждая из них представляет собой роль, которая может быть выражена набором подпрограмм, преследующим определенную цель.

4. С роли на роль Субъект переключается самостоятельно,

исходя из анализа состояния системы.

Совершенствование самоорганизующихся приложений в мире информационных систем повторяет эволюцию живой природы - от простейших одноклеточных к более сложным организмам.

Не случайно первыми объектами, приспосабливающимися к среде обитания, были компьютерные вирусы, чье поведение очень похоже на поведение их "соДратъев" из реальной жизни:

- внедриться в тело программы и использовать для "питания" ее машинное время;**
- скрытно размножаться;**
- выполнять деструктивные действия в зараженных системах.**



Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана»
(МГТУ им. Н. Э. Баумана)

Цикл лекций по дисциплине

«Методология программной инженерии»

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук.

rtn@bmstu.ru

Москва - 2024

Лекция3

Характеристики качества программных средств

*Основные факторы, определяющие качество сложных
программных средств*

***Стандарты регламентирующие качество ПО**

- 1. ISO 9126:1991.** (ГОСТ РФ – 1993). ИТ. Оценка программного продукта. Характеристики качества и руководство по их применению.

- 2. ISO 9126-1-4: 2002.** ИТ. ТО. Качество программных средств: Ч.1. Модель качества.
Ч.2. Внешние метрики.
Ч. 3. Внутренние метрики.
Ч. 4. Метрики качества в использовании

Общее представление о качестве ПС международным стандартом **ISO 9126:1-4:2002** рекомендуется описывать тремя взаимодействующими и взаимозависимыми *метриками характеристик качества*, отражающими:

- **внутреннее качество**, проявляющееся в процессе разработки и других промежуточных этапов жизненного цикла ПС;
- **внешнее качество**, заданное требованиями заказчика в спецификациях и отражающееся характеристиками конечного продукта;
- **качество при использовании** в процессе нормальной эксплуатации и результативностью достижения потребностей пользователей с учетом затрат ресурсов.



Рис. 1.1. Основные характеристики качества ПС

* Модель характеристик качества ПС и компонентов состоит из шести групп базовых показателей, каждая из которых детализирована несколькими нормативными субхарактеристиками :

* **1. Функциональные возможности** детализируются:

- a) Функциональной пригодностью* для применения по назначению;
- b) корректностью (правильностью, точностью)* реализации требований;
- c) способностью к взаимодействию с компонентами и средой;*
- d) защищенностью – безопасностью функционирования.*

***2. Надежность** характеризуется:

- a) уровнем завершенности* – отсутствием дефектов и ошибок;
- b) устойчивостью* при наличии дефектов и ошибок;
- c) восстанавливаемостью* после проявления дефектов;
- d) доступностью* – готовностью реализации требуемых функций.

***3. Эффективность** рекомендуется отражать:

- a) временной эффективностью* реализации комплекса программ;
- b) используемостью вычислительных ресурсов.*

***4. Применимость (практичность)** предлагается описывать:

- a) понятностью функций и документации;*
- b) простотой использования комплекса программ;*
- c) изучаемостью процессов функционирования и применения.*

***5. Сопровождаемость** представляется:

- a)анализируемостью – удобством для анализа предложений модификаций;*
- b)изменяемостью компонентов и комплекса программ;*
- c)тестируемостью изменений при сопровождении.*

***6. Мобильность (переносимость)** предлагается отражать:

- a)адаптируемостью к изменениям среды;*
- b)простотой установки – инсталляции после переноса;*
- c)замещаемостью компонентов при корректировках комплекса программ.*

1.а. **Функциональная пригодность** - это набор и описания атрибутов, определяющих *назначение, основные, необходимые и достаточные функции ПС*, заданные техническим заданием и спецификациями требований заказчика или потенциального пользователя.

В процессе проектирования комплекса программ атрибуты функциональной пригодности должны конкретизироваться в спецификациях на ПС в целом и на компоненты.

***При формулировании и выборе функциональных требований следует четко формулировать в документах контракта:**

- экономические, организационные, технические и/или социальные стратегические цели всего жизненного цикла ПС и его компонентов;
- назначение, внешнюю среду и условия эффективного применения ПС;
- системную эффективность и, в том числе, требуемые технико-экономические показатели применения ПС в составе системы;
- функциональные задачи основных компонентов и ПС в целом, а также системную эффективность каждого;
- необходимое и достаточное качество и временной регламент решения каждой функциональной задачи;
- соответствие ПС и его компонентов стандартам и нормативным документам на проектирование и применение;
- ограничения параметров внешней среды и условий для применения ПС, гарантирующие требуемые характеристики функциональной пригодности.

Функциональная пригодность в течение жизненного цикла ПС зависит от ***структурных (архитектурных) характеристик***, которые должны отражаться в требованиях ТЗ на компоненты и ПС в целом:

- соответствие функций и структуры программного средства аппаратной и операционной среде и их ограниченным ресурсам;
- состав, структура и способы организации данных, а также требования к обмену данными между компонентами ПС, должны быть адекватны организации информационного обеспечения и функциям системы;
- должны быть заданы временной регламент и характеристики процесса динамической реализации автоматизированных функций;
- должно быть определено допустимое время, задержки выдачи результатов решения задач;
- должны быть предусмотрены и реализованы требования к контролю, хранению, обновлению и восстановлению программ и данных.

При этом должны быть определены:

- назначение БД и состав информации в ней;
- совместимость ПС с другими системами по источникам и потребителям информации БД;
- организация сбора, передачи, контроля и корректировки информации БД;
- интенсивность и объемы потоков информации.

1.b. Правильность - корректность: это способность ПС обеспечивать правильные (или приемлемые) результаты для пользователей.

Эталонами для выбора требований к корректности при проектировании могут быть:

- верифицированные и взаимоувязанные требования к функциям комплекса, компонентов и модулей программ,
- правила их структурного построения, организация взаимодействия и интерфейсов.

Субхарактеристики, атрибуты качества и свойства для выбора функциональных возможностей программных средств

Субхарактеристики	Атрибуты качества и свойства
Корректность	<ul style="list-style-type: none">— соответствие требований к функциям ПС требованиям к информационной системе;— соответствие требований к функциональным компонентам требованиям к функциям ПС;— соответствие текстов программ требованиям к функциональным компонентам ПС;— соответствие объектного кода исходному тексту программ функциональных компонентов ПС;— степень покрытия тестами функций и возможных маршрутов исполнения программ
Способность к взаимодействию	<ul style="list-style-type: none">— с операционной системой и аппаратной средой;— с внешней средой системы и с пользователями;— между программными компонентами;— между компонентами распределенных информационных систем
Защищенность	<ul style="list-style-type: none">— соответствие критериям и требованиям защиты от предумышленных угроз безопасности ПС;— соответствие методам и средствам защиты от проявления случайных дефектов программ и данных;— обеспечение эффективности оперативных методов защиты и восстановления при проявлениях и реализации угроз безопасности;— соответствие стандартам и нормативным документам на защиту от различных типов угроз безопасности;— обеспечение равнопрочной защиты в соответствии с опасностью угроз и доступностью ресурсов для защиты

2. Надежность - свойства комплекса программ обеспечивать достаточно низкую *вероятность потери работоспособности – отказа*, в процессе функционирования ПС в реальном времени.

Основные атрибуты надежности могут быть объективно измерены и сопоставлены с требованиями.

Основные количественные характеристики программных средств и их атрибуты

Характеристики качества	Мера	Шкала
Надежность <i>Завершенность:</i> наработка на отказ при отсутствии рестарта; степень покрытия тестами функций и структуры программ	Часы %	10—1000 50—100
Устойчивость: наработка на отказ при наличии автоматического рестарта; относительные ресурсы на обеспечение надежности и ре- старта	Часы %	10—1000 10—90
Восстанавливаемость: длительность восстановления	Минуты	10^{-2} —10
Доступность-готовность: относительное время работоспособного функционирования	Вероятность	0,9—0,999
Эффективность <i>Временная эффективность:</i> время отклика — получения результатов на типовое задание; пропускная способность — число типовых заданий, исполн- яемых в единицу времени	Секунды Число в ми- нуту	0,1—100 1—1000
Используемость ресурсов: относительная величина использования ресурсов ЭВМ при нормальном функционировании программного средства	Вероятность	0,7—0,95

Качественные характеристики

1. Практичность (применимость).

Это свойство ПС, отражающее сложность его понимания, изучения и применения. Эти свойства зависят от назначения системы и её функций.

2. Понятность.

Это свойства ПС, обеспечивающие пользователю понимание, является ли программа пригодной для его целей, и как ее можно использовать для конкретных задач и условий применения.

Ее можно описать качественно четкостью концепции, широтой демонстрационных возможностей, полнотой, комплектностью и наглядностью представления в эксплуатационной документации возможных функций и особенностей их реализации.

Основные качественные характеристики программных средств и их атрибуты

Характеристики качества	Мера	Шкала
<p>Практичность</p> <p>Понятность:</p> <p>четкость концепции ПС;</p> <p>демонстрационные возможности;</p> <p>наглядность и полнота документации</p>	Порядковая	Отличая; хорошая; удовлет.; неудовлет.

Простота использования: возможность пользователю удобно и комфортно эксплуатировать и управлять ПС. Аспекты изменяемости, адаптируемости и легкости инсталляции могут быть предпосылками для простоты использования и выбора конкретного ПС. Она соответствует управляемости, устойчивости к ошибкам и согласованности с ожиданиями и навыками пользователей. Эта субхарактеристика должна учитывать физические и психологические особенности пользователей и отражать уровень контролируемости и комфорtnости условий эксплуатации ПС, возможность предотвращения ошибок пользователей. Должны обеспечиваться простота управления функциями ПС и достаточный объем параметров управления, реализуемых по умолчанию, информативность сообщений пользователю, наглядность и унифицированность управления экраном, а

Характеристики качества	Мера	Шкала
Простота использования: простота управления функциями; комфортность эксплуатации; Среднее время ввода заданий; Среднее время отклика на задание.	Порядковая Секунды Секунды	Отличая; хорошая; удовлет.; неудовлет. 1—1000 1—1000
Изучаемость трудоемкость изучения применения ПС; продолжительность изучения; объем эксплуатационной документации; объем электронных учебников	Чел.-часы Часы Страницы Кбайты	1—100 1—1000 10—1000 100—10000
Сопровождаемость Анализируемость: стройность архитектуры программ; унифицированность интерфейсов; полнота и корректность документации	Порядковая	Отличая; хорошая; удовлет.; неудовлет.
Изменяемость: трудоемкость подготовки изменений; длительность подготовки изменений.	Чел.-часы Часы	1—1000 1—1000
Тестируемость: трудоемкость тестирования изменений; длительность тестирования изменений	Чел.-часы Часы	1—1000 1—100
Мобильность Адаптируемость: трудоемкость адаптации; длительность адаптации.	Чел.-часы Часы	1 – 100 1 – 100
Простота установки: трудоемкость инсталляции; длительность инсталляции.	Чел.-часы Часы	1 – 100 1 – 100
Замещаемость: трудоемкость замены компонентов; длительность замены компонентов	Чел.-часы Часы	1 – 100 1 – 100

Изменяемость: приспособленность ПС к простой реализации специфицированных изменений и к управлению конфигурацией. Реализация модификаций включает проектирование, кодирование и документирование изменений. Для этого требуются определенная трудоемкость и время, связанные с исправлением дефектов и/или модернизацией функций, а также с изменением процессов эксплуатации. При выборе атрибутов этой субхарактеристики следует учитывать влияние структуры, интерфейсов и технических особенностей ПС. Изменяемость зависит не только от внутренних свойств ПС, но также от организации и инструментальной оснащенности процессов сопровождения и конфигурационного управления, на которые ориентированы архитектура, внешние и внутренние интерфейсы программ.

- дополнительные затраты на обеспечение высокой *надежности* ПС могут достигать 2—3-кратного увеличения затрат относительно функциональной пригодности при требованиях наработки на отказ в десятки тысяч часов, а для минимального обеспечения автоматического рестарта в ординарных системах составляют порядка 10—20%;
- для повышения *эффективности использования ресурсов* ЭВМ затраты могут быть относительно невелики (несколько процентов) и их трудно выделить из затрат на решение основных, функциональных задач;
- затраты на обеспечение *практичности* зависят в основном от сложности применения ПС, от качества и количества эксплуатационной документации и электронных учебников и могут составлять до 20% затрат на решение основных, функциональных задач.

2.3 Основные этапы процедуры оценки качества ПС

Оценка качеств ПС проводится на фазах ЖЦ (см. таб. 2.1) и включает выбор номенклатуры показателей, их оценку и сопоставление значений показателей, полученных в результате сравнения с базовыми значениями.

Таблица 2.1 Фазы жизненного цикла ПС

Процесс	Фаза	Подфаза	Результат
Разработка	Анализ	-	<ul style="list-style-type: none">• Определение требований.• Спецификация требований.• Техническое задание.
	Проектирование	Логическое проектирование	Логический проект (функциональный проект). Программно-технический проект: <ul style="list-style-type: none">• системы;• программ;• модулей;• документации;
	Реализация	-	<ul style="list-style-type: none">• Модули.• Программы.• Система.• Средства тестирования.• Дополняющая документация.

	Тестирование	-	<ul style="list-style-type: none"> • Тестирование модуля, программы, системы, дополняющая документация. • Сдача в фонд (при необходимости).
	Изготовление	Выпуск	<ul style="list-style-type: none"> • Программное средство в форме, готовой для поставки. • Документация. • Правила внесения изменений.
		Испытания	<ul style="list-style-type: none"> • Установленное ПС. • Организация применения. • Отчет об испытаниях. • Отзыв пользователя.
Применение	Внедрение	-	<ul style="list-style-type: none"> • Подтверждающее стабильной эксплуатации. • Предоставление набора услуг по внедрению.
	Эксплуатация	-	<ul style="list-style-type: none"> • Предложения об усовершенствовании. • Сообщение о функциональных отклонениях.
	Обслуживание (сопровождение)	-	<ul style="list-style-type: none"> • Информация о сопровождении программ. • Измененное.



*Федеральное государственное бюджетное образовательное учреждение высшего
образования «Московский государственный технический университет
* имени Н. Э. Баумана (национальный исследовательский университет)»
*(МГТУ им. Н. Э. Баумана)

Цикл лекций по дисциплине «Методология программной инженерии»

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук.

rtn@bmstu.ru

Москва - 2024



Лекция 4

Дефекты, ошибки и риски в жизненном цикле программных средств.

Общие особенности дефектов, ошибок и рисков в сложных ПС.

Риски при формировании требований к характеристикам сложных ПС

- * *Статистика ошибок и дефектов в комплексах программ и их характеристики* в конкретных типах проектов ПС, могут служить *ориентирами* для разработчиков при распределении ресурсов в жизненном цикле ПС и предохранять их от излишнего оптимизма при оценке достигнутого качества программных продуктов.
- * Источниками ошибок в ПС являются специалисты – конкретные люди с их индивидуальными особенностями, квалификацией, талантом и опытом.

* Вследствие этого, плотность потоков и размеры необходимых корректировок в модулях и компонентах при разработке и сопровождении ПС может различаться в десятки раз.

* В крупных комплексах программ статистика и распределение типов выполняемых изменений для коллективов разных специалистов нивелируются и проявляются достаточно общие закономерности, которые могут использоваться как ориентиры при их выявлении и систематизации.

* Этому могут помочь оценки типовых дефектов, модификаций и корректировок, путем их накопления и обобщения по опыту создания определенных классов ПС в конкретных предприятиях.

* К *понятию риски* относятся негативные события и их величины, отражающие потери, убытки или ущерб от процессов или продуктов, *вызванные дефектами* при проектировании требований, недостатками обоснования проектов ПС, а также при последующих этапах разработки, реализации и всего жизненного цикла комплексов программ.

*Риски проявляются, как *негативные последствия дефектов функционирования и применения ПС*, которые способны вызвать ущерб системе, внешней среде или пользователю, в результате отклонения характеристик объектов или процессов, от заданных требованиями заказчика, согласованными с разработчиками.

Оценки качества программных средств

могут проводиться с двух позиций:

- ❖ *с позиции положительной* эффективности и непосредственной адекватности их характеристик назначению, целям создания и применения;
- ❖ *с негативной позиции* – возможность ущерба от использования ПС или системы.

Показатели качества преимущественно отражают положительный эффект от применения системы. Основная задача разработчиков проекта состоит в обеспечении высоких значений качества.

- * Риски характеризуют возможные *негативные последствия дефектов* или ущерб пользователей при применении и функционировании ПС.
- * Задача разработчиков сводится к сокращению дефектов и ликвидации рисков.
- * Поэтому методы и системы управления качеством в жизненном цикле ПС близки к методам анализа и управления рисками проектов комплексов программ, они должны их дополнять и совместно способствовать совершенствованию программных продуктов и систем на их основе.

Характеристики дефектов и рисков связаны с достигаемой корректностью, безопасностью и надежностью функционирования программ и помогают:

- 1) оценивать реальное состояние проекта и планировать необходимую трудоемкость и длительность для его положительного завершения;
- 2) выбирать методы и средства автоматизации тестирования наиболее эффективные для устранения определенных видов дефектов и рисков;

- 3) рассчитывать необходимую эффективность контрмер и дополнительных средств оперативной защиты от потенциальных дефектов и не выявленных ошибок;
- 4) оценивать требующиеся ресурсы ЭВМ по расширению памяти и производительности, с учетом затрат на реализацию контрмер при модификации и устраниении ошибок и рисков.

Понятие ошибки в программе

- * в общем случае под ошибкой подразумевается ***неправильность, погрешность или неумышленное искажение объекта или процесса***, что может быть **причиной ущерба – риска** при функционировании и применении программы.
- * При этом предполагается, что ***известно правильное, эталонное состояние объекта или процесса***, по отношению к которому может быть определено наличие отклонения – ошибки или дефекта.
- * Исходным эталоном для любого ПС являются спецификация требований заказчика или потенциального пользователя, предъявляемых к программам.

* Любое отклонение результатов функционирования программы от предъявляемых к ней требований и сформированных по ним эталонов-тестов, следует квалифицировать как *ошибку – дефект в программе*, наносящий некоторый ущерб.

* Различие между ожидаемыми и полученными результатами функционирования программ могут быть следствием ошибок не только в созданных программах, но и ошибок в первичных требованиях спецификаций, явившихся базой при создании эталонов-тестов.

*Важной особенностью процесса выявления ошибок в программах является *отсутствие полностью определенной программы-эталона*, которой должны соответствовать текст и результаты функционирования разрабатываемой программы.

При отладке и тестировании обычно сначала обнаруживаются вторичные ошибки и риски, т.е. последствия и результаты проявления некоторых внутренних дефектов или некорректностей программ.

* Эти внутренние *дефекты* следует квалифицировать как *первичные* ошибки или причины обнаруженных аномалий результатов.

* Последующая локализация и корректировка таких первичных ошибок должна приводить к устранению ошибок, первоначально обнаруживаемых в результатах функционирования программ.

Разработчики проекта программного средства



Требования спецификаций и ограничения ресурсов проекта программного средства

Первичные ошибки в программах и данных — потенциальные угрозы проекту программного средства

Вторичные ошибки в результатах применения программного средства

Риски — ущерб системе и внешней среде от вторичных ошибок в программном средстве

Диагностика вторичных ошибок и рисков в программном средстве для обнаружения и исключения первичных ошибок

Сокращение или ликвидация вторичных ошибок и рисков в результате исключения первичных ошибок

Вторичные ошибки являются определяющими для эффективности функционирования программ.

Однако не каждая первичная ошибка вносит заметный вклад в выходные результаты, поскольку ряд первичных ошибок может оставаться не обнаруженным.

Классификация ошибок

- * *Небольшими ошибками* называют такие, на которые средний пользователь может и не обратить внимания при применении ПС.
- * Небольшие ошибки могут включать орфографические ошибки на экране, пропущенные разделы в справочнике и другие мелкие проблемы.
- * *По десятибалльной шкале рисков небольшие ошибки находятся в пределах от 1 до 3 приоритета.*

Умеренными ошибками называют те, которые влияют на конечного пользователя, но имеются слабые последствия или обходные пути, позволяющие сохранить достаточную функциональность ПС.

К умеренным ошибкам относятся дефекты:

- неверные ссылки на страницах,
- ошибочный текст на экране,
- сбои, если их трудно воспроизвести, и они не оказывают влияния на существенное число пользователей.
- Ошибки, которые можно исправить на этом уровне, следует исправлять, если на это есть время и возможность.

По десятибалльной шкале умеренные ошибки находятся в диапазоне от 4 до 7 приоритета.

Критические ошибки останавливают выпуск версии программного продукта!

Это могут быть *ошибки с высоким влиянием*, которые вызывают сбой в системе или потерю данных, отражаются на надежности и безопасности применения ПС, с которыми никогда не передается комплекс программ пользователю.

По десятибалльной шкале – от 8 до 10 приоритета.

Совокупность ошибок, дефектов и последствий модификаций проектов крупномасштабных комплексов программ можно упорядочить и условно представить в виде перевернутой пирамиды в зависимости от потенциальной опасности и возможной величины корректировок их последствий.

Типы дефектов, ошибок и модификаций при сопровождении программных средств

Дефекты при модификации и расширении функций программного средства

Ошибки оценки изменений характеристик системы и внешней среды

Ошибки вследствие оценки сложности модификаций программного средства

Ошибки вследствие большого масштаба корректировок программного средства

Ошибки корректности требований изменения программного средства

Ошибки проектирования и структуры программного средства

Системные ошибки программного средства

Алгоритмические ошибки программного средства

Ошибки реализации спецификаций программных компонентов

Программные ошибки компонентов

Ошибки в документации программного средства

Технологические ошибки ввода данных

Размер корректировок последствий дефектов и ошибок

Специалисты — источники дефектов и ошибок	Типы первичных дефектов и ошибок программного средства и документации
Заказчики проекта	Дефекты организации проекта и исходных требований заказчика
Менеджер проекта	Дефекты, обусловленные реальной сложностью проекта
Менеджер-архитектор комплекса программ	Ошибки планирования и системного проектирования программного средства
Проблемно-ориентированные аналитики и системные архитекторы	Системные и алгоритмические дефекты и ошибки проекта
Спецификаторы компонентов проекта	Алгоритмические ошибки компонентов и документов программного средства
Разработчики программных компонентов — программисты	Программные дефекты и ошибки компонентов и документов программного средства
Системные интеграторы	Системные ошибки и дефекты реализации версий программного средства и документации
Тестировщики	Программные и алгоритмические ошибки программного средства и документации
Управляющие сопровождением и конфигураций, инструкторы интерфейсов	Ошибки проектирования и реализации версий программного продукта
Документаторы	Дефекты и ошибки обобщающих документов

Принципы и свойства дефектов, ошибок и модификаций в сложных программных средствах

- * *Изменения характеристик системы и внешней среды*, принятые в процессе разработки ПС за исходные, могут быть результатом аналитических расчетов, моделирования или исследования аналогичных систем.
- * Может отсутствовать полная адекватность предполагаемых и реальных характеристик, что является *причиной сложных и трудно обнаруживаемых системных ошибок и дефектов развития проекта*.

* Эксперименты по проверке взаимодействия ПС с реальной внешней средой во всей области изменения параметров зачастую сложны и дороги, а при создании опасных ситуаций недопустимы.

* В этих случаях приходится использовать моделирование и имитацию внешней среды с заведомым упрощением её отдельных элементов и характеристик.

* *Однако полной адекватности моделей внешней среды и реальной системы добиться трудно, что может являться причиной крупных дефектов.*

* В исследованиях 20 крупных поставляемых программных продуктов, созданных в 13 различных организациях коллективы специалистов добились среднего уровня 0,06 дефекта на тысячу строк нового и измененного программного кода.

* При использовании структурного метода в пяти проектах достигнуто 0,04 – 0,075 ошибок на тысячу строк.

* Таким образом, *уровень ошибок около 0,05 на тысячу строк кода* в разных публикациях считается близким к предельному, для высококачественных программных продуктов.

* Другим примером оценок уровня ошибок критического ПС особенно высокого качества может служить программный продукт бортовых систем Шатла, созданный NASA.

* По оценке авторов в нем содержится менее одной ошибки на 10 000 строк кода.

Стоимость такого ПИ достигает 1000 \$ за строку кода.

* Это в среднем в 100 раз больше, чем для административных систем и в 10 раз больше, чем – для ряда ординарных критических управляющих систем реально времени.

Риски в жизненном цикле сложных ПС

Причины возникновения и проявления рисков:

1. Злоумышленные, активные воздействия заинтересованных лиц.

В этом случае **риски** могут быть обусловлены искажениями программ и информационных ресурсов и их уязвимостью от предумышленных, внешних воздействий (атак) с целью незаконного использования или искажения информации и программ.

Для решения этой проблемы созданы и активно развиваются методы, средства и стандарты обеспечения защиты программ и данных от **предумышленных негативных внешних воздействий**.

2. Риски при случайных, дестабилизирующих воздействиях дефектов ПС и отсутствии предумышленного негативного влияния на системы, ПС или информацию баз данных.

- Эти риски объектов и систем зависят от ситуаций отказа системы, отрицательно отражающихся на работоспособности программного комплекса и реализации их основных функций.
- Причинами которых могут быть дефекты и аномалии в аппаратуре, программах, данных или вычислительных процессах.
- При этом катастрофически, критически или существенно искажается процесс функционирования систем, что может наносить значительный ущерб при их применении.

- ❖ Процессы анализа и сокращения рисков должны сопутствовать основным этапам ЖЦ в соответствии с международными стандартами, а также методам систем обеспечения качества ПС.
- ❖ Эти процессы могут быть отражены *пятью этапами работ и процедур*, которые рекомендуется выполнять при поддержке базовых работ ЖЦ ПС, и могут служить основой для разработки соответствующих планов работ при управлении и сокращении рисков (Рис.2).

Подготовка исходных данных:

- определение целей, назначения и функций проекта программного средства и системы;
- разработка предварительных требований к функциональной пригодности и конструктивным характеристикам ПС;
- формирование группы экспертов для анализа угроз и управления рисками



Выделение, идентификация, анализ угроз и рисков программного средства:

- выделение источников и угроз нарушения требований и ограничений ресурсов, определение критерии работоспособности проекта ПС;
- анализ причин, выделение категорий угроз и возможных последствий рисков функциональной пригодности ПС;
- анализ причин, угроз и возможных рисков конструктивных характеристик и ограничения ресурсов проекта ПС



Оценивание опасности угроз и рисков и выбор контрмер для их сокращения:

- оценивание возможных последствий, уровней потенциальных угроз и приоритетов категорий рисков проекта ПС;
- планирование методов и ресурсов реализации контрмер для сокращения опасных, приоритетных рисков проекта ПС;
- распределение ресурсов на контрмеры для сбалансированного сокращения интегрального риска проекта ПС



Сокращение или ликвидация опасных рисков ПС:

- реализация контрмер для сокращения интегрального риска и составление отчетов о состоянии проекта;
- корректировка требований к функциональной пригодности, конструктивным характеристикам и ресурсов проекта ПС;
- регистрация результатов сокращения интегрального риска на очередном этапе проекта ПС



Контроль, регистрация, мониторинг и утверждение допустимого интегрального риска ПС:

- контроль, отслеживание и мониторинг реализации сокращения интегрального риска по этапам проекта ПС;
- документирование и утверждение допустимого интегрального риска по этапам жизненного цикла проекта ПС;
- оформление итогового отчета по результатам сокращения рисков ПС

* Если при оценке масштаба проекта ПС его величина в договоре определена *недостаточной – заниженной, то основной ущерб – риски достаются разработчикам.*

* Они вынуждены принимать жесткие меры для выполнения плановых сроков, выделенного бюджета работ при относительно малом числе специалистов, что угрожает рисками срыва сроков и нарушения стоимости проекта.

*** Недооценка размера проекта комплекса программ и ресурсов для его реализации, может резко увеличивать число дефектов в программах.**

*Кроме того, *ограниченные ресурсы* для реализации в полном объеме функциональной пригодности, могут отражаться на ее качестве и удобстве применения, а также на конструктивных характеристиках качества ПС:

- * на безопасности,
- * на надежности,
- * качестве взаимодействия с внешней средой и с пользователями,
- * качестве документации и других эксплуатационных факторах.

Риски при формировании требований к характеристикам сложных ПС

Для устранения или снижения рисков до допустимых пределов может быть необходимо изменение требований к функциональной пригодности, к конструктивным характеристикам и доступным ресурсам.

Поэтому на этапах проектирования последовательно
должны определяться:

1. При проектировании концепции и первичной оценке масштаба проекта:

* предварительные требования к назначению, функциональной пригодности и к конструктивным характеристикам качества ПС.

2. При предварительном проектировании :

- уточненная оценка масштаба проекта,
- требования к функциональным и конструктивным характеристикам качества с учетом общих ограничений ресурсов,
- перечень источников угроз и величины возможных рисков.

3. При детальном проектировании :

- подробные спецификации требований к функциональным и конструктивным характеристикам качества с детальным учетом и распределением реальных ограниченных ресурсов,
- интегральные риски при их оптимизации по критерию качество/затраты.

Для крупномасштабных проектов комплексов программ при уточнении требований к качеству при детальном проектировании целесообразно использовать *обобщенный уровень приоритета – риска*.

Эти данные могут использоваться, прежде всего, *как ориентиры* для селекции и планирования проектов.

Три группы обобщенных уровней приоритетов :

- 1.Доминирующие характеристики и риски, оказывающие наибольшее влияние на функциональную пригодность при допустимых затратах (**обобщенный приоритет > 8**).
- 2.Показатели конструктивных характеристик и рисков, имеющие достаточное влияние на функциональную пригодность и значительные затраты на реализацию (**4<обобщенный приоритет < 7**).
- 3.Характеристики качества, значения требований к которым не соответствуют их влиянию на функциональную пригодность и/или затратам на реализацию и могут не учитываться (**обобщенный приоритет < 3**).

Рекомендации:

*Следует исключить из требований,
конструктивные характеристики качества с
особенно низкими обобщенными
приоритетами рисков поскольку они в
наименьшей степени влияют на
функциональную пригодность ПС и не
оправдывают больших затрат на
реализацию!*



Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

Цикл лекций по дисциплине

«Методология программной инженерии»

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук.
rtn@bmstu.ru

Россия, Москва - 2024

Лекция 5

Управление конфигурацией в жизненном цикле программных средств

Процессы управления конфигурацией ПС.

Документирование конфигурационного управления

Цель управления конфигурацией при разработке и сопровождении сложных программных средств и систем, состоящих из многих компонентов, каждый из которых может иметь разновидности или версии, состоит в том, чтобы обеспечить управляемое и контролируемое развитие их структуры, состава компонентов и функций, а также сокращение дефектов в течение всего жизненного цикла ПС.

Процесс управления конфигурацией является процессом применения административных и технических процедур на всем протяжении ЖЦ ПС для:

- обозначения, определения и установления состояния базовой версии программных продуктов в системе;
- управления изменениями и выпуском объектов;
- описания и сообщения о состояниях объектов и заявок на внесение изменений в них;
- обеспечения полноты, совместимости и правильности объектов;
- управления хранением, обращением и поставкой объектов.

ISO 12207:1995. (ГОСТ Р – 1999). ИТ. Процессы жизненного цикла программных средств.

Процесс управления конфигурацией включает:

- 1) подготовку процесса;
- 2) определение конфигурации;
- 3) контроль конфигурации;
- 4) учет состояний конфигурации;
- 5) оценку конфигурации;
- 6) управление выпуском и поставку программного продукта.
- 7) Все основные и вспомогательные процессы подлежат адаптации и конкретизации применительно к характеристикам определенного проекта.

Стандарт ISO 15846 обобщает, детализирует и развивает основные концептуальные положения, представленные в стандарте ISO 12207.

В процессе проектирования ПС должна быть формализована и документально зафиксирована под названием

Концепция организации конфигурационного управления проектом по созданию ПС, содержащая в своей основе:

- ожидаемую длительность поддержки развития и модификации конкретного проекта ПС;
- масштаб и уровень предполагаемых изменений и модификаций;
- возможное число и периодичность выпуска базовых версий ПС;
- организационные основы процессов сопровождения и конфигурационного управления ПС;
- требования к документированию изменений и базовых версий;
- кто будет осуществлять управление конфигурацией:
покупатель, разработчик или специальный персонал поддержки ЖЦ ПС.



Объекты изменения	Типы изменений объектов	Право на выполнение изменения имеют:	Право на утверждение изменения имеют:
Версии модулей и компонентов программ, данных и документов	Устранение дефектов в программных модулях и компонентах	Программисты — разработчики модулей и компонентов	Руководители разработки функциональных групп программ и данных
Версии функциональных групп программ и документов	Корректировка функций и взаимодействия программных компонентов	Руководители разработки функциональных компонентов	Менеджер-архитектор программного средства
Базовые версии программного продукта и комплекса документации	Модификация и улучшение функций и качества базовых версий программного продукта	Менеджер-архитектор программного продукта	Менеджер и заказчик проекта программного продукта
Версии программного продукта пользователей	Адаптация к характеристикам внешней среды пользователей	Заказчик или сопровождающий версию программного продукта пользователя	Менеджер, сопровождающий версию программного продукта пользователя

Процессы управления конфигурацией ПС направлены на достижения следующих основных целей:

- ***обеспечить возможность оценки*** соответствия требованиям заказчика результатов жизненного цикла программного средства;
- ***обеспечить управляемую конфигурацию ПС*** на протяжении всего ЖЦ ;
- ***обеспечить управление входными и выходными данными*** процесса в течение всего ЖЦ, что гарантирует непротиворечивость и повторяемость работ в процессах;
- ***обеспечить контрольные точки для проверки, оценки состояния и контроля изменений*** посредством управления элементами конфигурации и определения **базовой версии** программного продукта;
- ***обеспечить контроль*** над тем, чтобы фиксировались дефекты и ошибки, а изменения регистрировались, утверждались и реализовались;
- ***гарантировать надежное физическое архивирование***, восстановление и сопровождение единиц конфигурации ПС.

План изменений конфигурации должен
содержать следующие разделы:

- ❖ **почему** и с какой целью производится корректировка программ или данных;
- ❖ **кто** выполняет, и кто санкционирует проведение изменений комплекса программ или компонентов;
- ❖ **какие** действия и процедуры должны быть выполнены для реализации изменений единиц конфигурации;
- ❖ **когда** по срокам и в координации, с какими другими процедурами следует реализовать определенную модификацию компонентов и конфигурацию ПС;
- ❖ **как** и с использованием, каких средств и ресурсов должны быть выполнены запланированные изменения ПС и компонентов.

- ❖ Четкая *организация и автоматизация этого процесса* позволяют избегать множества вторичных ошибок, обусловленных недостаточной координацией проводимых корректировок и формирования новых версий ПС коллективом специалистов.
- ❖ Утверждается *иерархия принятия решений на изменения компонентов и ПС в целом на разных уровнях проекта* должностными лицами проекта и специалистами различной квалификации.

- *Архивирование и тиражирование базовых версий программных продуктов и документов* должны гарантировать использование исключительно санкционированных версий программного продукта и компонентов, так что официальные версии могут быть получены только из архива.
- Каждая единица конфигурации должна быть идентифицирована, документирована и выпущена ее официальная версия до того, как осуществляется производство ПС для пользователей.
- Цель работ по архивированию и применению документов обеспечить получение документов жизненного цикла ПС, для копирования, повторной генерации, повторного тестирования и модификации программного продукта.

Финансирование УК ПС должно определяться специальным договором между разработчиком и заказчиком.

★ В ТЗ и в контракте следует четко определить порядок квалификации видов и причин изменений в программах и данных, а также распределение ответственности за их инициализацию, реализацию и финансирование.

★ Выявленные ошибки в программах и данных, которые искажают реализацию функций, согласованных с заказчиком в контракте и требованиях спецификаций, а также отраженные в документации на версию ПС, должны устраняться за счет разработчика.

★ Модификацию и расширение функций компонентов или создание новых версий комплекса программ, ранее не отраженных в требованиях ТЗ и контракте с заказчиком,

★ следует квалифицировать как дополнительную работу с соответствующим финансированием заказчиком.

- ❖ Для установления достоверности сообщений пользователей о выявленных дефектах и ошибках необходима детальная регистрация сценариев и условий, при которых проявляются аномалии.
- ❖ Установление разработчиками достоверности ошибок начинается с тестирования *эталонной базовой версии ПС при исходных данных пользователя, обнаружившего дефект.*
- ❖ Если проявляется ошибка, то она регистрируется как подтвержденная при зафиксированных тестовых данных. При отсутствии проявления ошибок на эталонной версии, при тех же исходных данных целесообразно проводить последующее тестирование копии версии, адаптированной к условиям применения у этого пользователя.
- ❖ Если и при этом ошибка не проявляется, то регистрируется её не подтверждение и информация о дефекте снимается с учета.

- Многие предприятия для взаимодействия с пользователями организуют, “*горячие линии*” для консультаций.
- Эти консультации позволяют оперативно проводить первичную селекцию претензий пользователей к приобретенным и эксплуатируемым версиям ПС, обусловленных их некомпетентностью или недостатками эксплуатационной документации.
- Некоторые претензии могут иметь причиной попытки применения ПС за пределами условий, допускаемых документацией.



Предложения по корректировке ПС анализируются Советом Конфигурационного Управления по их влиянию на качество функционирования ПС и по затратам на осуществление изменений.

Приоритетность каждого предлагаемого изменения программ целесообразно оценивать по следующим критериям:

- насколько данное изменение может улучшить эксплуатационные характеристики ПС в целом;
- каковы затраты на выполнение корректировок при создании новой базовой версии и их распространение пользователям;
- возможно ли, и насколько сильно влияние изменений на функциональные характеристики остальных компонентов версии ПС;

- какова срочность извещения пользователя о разработанной корректировке и целесообразно ли ее распространять до подготовки очередной базовой версии;
- для какого числа пользователей может быть полезно данное изменение;
- как данное изменение отразится на эксплуатации пользователями предыдущих версий;
- насколько подготовка и оперативное внедрение данного изменения может отразиться на сроках создания очередной базовой версии ПС.

**Руководство по документированию сопровождения и управления
конфигурацией проекта программного средства**

**Методика оформления отчетов об отказах, дефектах и предложениях
по корректировке версий программного средства**

**Описания выявленных отказов, дефектов и предлагаемых изменений
программного средства**

**Руководство по анализу и выполнению корректировок модулей
и функциональных компонентов комплексов программ**

**Описания подготовленных и утвержденных корректировок компонентов
и версий программного продукта**

**Комплекты документов откорректированных и утвержденных
базовых версий программного продукта**

**Описания параметров адаптации и характеристик
пользовательских версий программного продукта**

**Описания комплектов тестов для проверки состояния и изменений
базовых версий программного продукта**

**Руководство по подготовке и распространению извещений
на частные изменения версий программного продукта**

**Организаторы системы взаимодействия специалистов
для сопровождения и управления конфигурацией проекта
программного продукта**

**Авторы предложений об изменениях или информации о дефектах
и ошибках**

**Аналитики предварительной селекции предполагаемых изменений
программного средства**

**Совет управления конфигурацией и разрешения изменений
программного продукта**

**Разработчики корректировок и тестировщики
программных компонентов**

**Специалисты управления конфигурацией, контроля и утверждения
корректировок компонентов программного средства**

**Интеграторы для сборки и квалификационного тестирования
модификаций версии программного продукта**

**Совет управления конфигурацией, формирования и утверждения
новой базовой версии программного продукта**

**Специалисты, обеспечивающие архивирование, хранение
и тиражирование пользовательских копий базовых версий
программного продукта**

**Администраторы управления базой данных модификаций
и версий программного продукта**

Защита электронного документооборота в современных компьютерных системах



МГТУ имени Н.Э. Баумана, кафедра ИУ8

Москва, 2021

Значимость информационных технологий (ИТ) в современном мире

Компьютеры проникли во все сферы деятельности человека, начиная с дошкольного образования и заканчивая изучением новейших технологий, новых видов материи, неизвестных пока человечеству.

Благодаря разнообразию программного и аппаратного обеспечения сегодня возможно использование всех потенциальных возможностей информационных технологий (ИТ). Это позволяет хранить огромное количество информации, занимая при этом минимальное место.

Информационные технологии позволяют быстро обрабатывать информацию и держать ее в защищенном виде, обеспечивая информационную безопасность (ИБ).

Электронный документооборот (ЭДО) — совокупность автоматизированных процессов по работе с документами, представленными в электронном виде

Почему так бывает — используешь средства защиты информации (СЗИ),
а информация все равно
попадает в чужие руки ?

Понятие информационной безопасности

Информационная безопасность (ИБ) - это состояние защищенности личности, общества и государства от внутренних и внешних информационных угроз *.

Защита информации (ЗИ) представляет собой принятие правовых, организационных и технических мер, направленных на:

- 1) обеспечение состояния защищенности информации от неправомерного доступа, уничтожения, модификации, блокирования, копирования, предоставления, распространения, а также от иных неправомерных действий в отношении информации;
- 2) соблюдение конфиденциальности информации ограниченного доступа;
- 3) реализацию права на доступ к информации.

* **Доктрина информационной безопасности** - система официальных взглядов на обеспечение национальной безопасности РФ в информационной сфере. Указ Президента РФ от 5 декабря 2016 года, № 646

Актуальность проблемы обеспечения ИБ

Зарубежные страны наращивают возможности по воздействию на ИТ инфраструктуру в военных целях.

Усиливается деятельность организаций, осуществляющих техническую разведку в отношении российских организаций.

Внедрение ИТ без увязки с ИБ повышает вероятность проявления угроз.

Специальные службы используют методы информационно-психологическое воздействия на граждан.

Все больше зарубежных СМИ доносят информацию предвзято.

Российские СМИ за рубежом подвергаются дискриминации.

Внешнее информационное воздействие размывает традиционные российские духовно-нравственные ценности (особенно у молодежи).

Террористические и экстремистские организации широко используют механизмы информационного воздействия.

Возрастают масштабы компьютерной преступности, прежде всего в кредитно-финансовой сфере

Методы, способы и средства совершения компьютерных преступлений становятся все изощреннее.

Повышается сложность и количество скоординированных компьютерных атак на объекты критической информационной инфраструктуры (КИИ).

Остается высоким уровень зависимости отечественной промышленности от зарубежных ИТ.

Российские научные исследования в сфере ИТ являются недостаточно эффективными, ощущается недостаток кадров.

У российских граждан низкая осведомленность в вопросах обеспечения личной ИБ. Отдельные государства стремятся использовать технологическое превосходство для доминирования в информационном пространстве. В том числе и в сети Интернет.

Многомерность проблемы обеспечения ИБ

Обеспечение информационной безопасности предполагает осуществление многомерной деятельности, носящей систематический и комплексный характер. При ее осуществлении важно иметь в виду те задачи, которые выдвигаются перед сторонами, заинтересованными в информационной безопасности.

Выделяют четыре категории субъектов информационной безопасности, отличающиеся друг от друга правовым, техническим, финансовым, организационным и иным ресурсным обеспечением своей информационной безопасности, а именно: **государство в целом; государственные организации; коммерческие структуры; отдельные граждане.**

В обеспечении информационной безопасности выделяют несколько уровней:

- Концептуально-политический
- Законодательный
- Нормативно-технический
- Административный
- Программно-технический

Современная постановка задачи обеспечения ИБ

Основой безопасной ИТ-инфраструктуры является триада сервисов – Конфиденциальность, Целостность, Доступность – Confidentiality, Integrity, Availability (CIA).

Целью информационной безопасности является обеспечение трех наиболее важных сервисов безопасности: конфиденциальность, целостность и доступность.

Конфиденциальность – это гарантия, что информация может быть прочитана и проинтерпретирована только теми людьми и процессами, которые авторизованы это делать. Обеспечение конфиденциальности включает процедуры и меры, предотвращающие раскрытие информации неавторизованными пользователями. Информация, которая может считаться конфиденциальной, также называется чувствительной. Примером может являться почтовое сообщение, которое защищено от прочтения кем бы то ни было, кроме адресата.

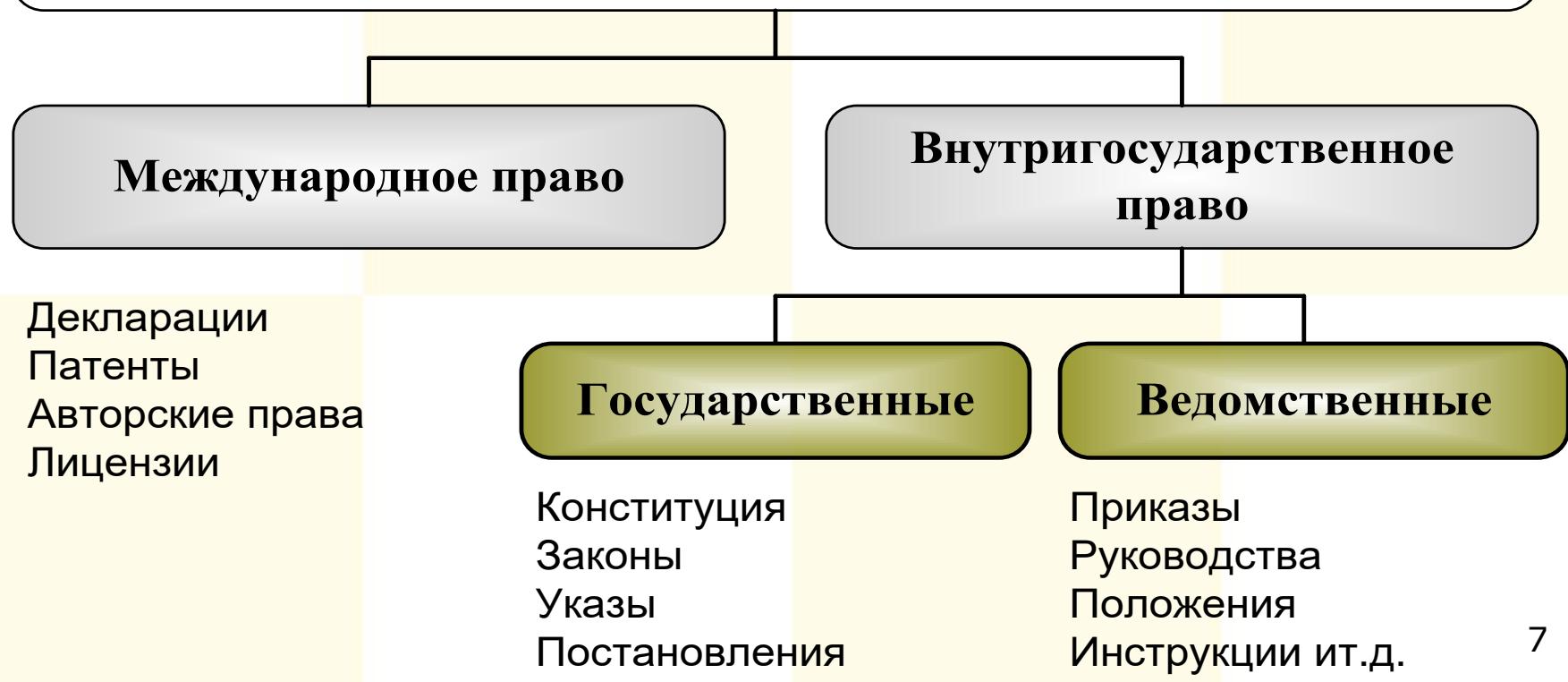
Целостность – это гарантирование того, что информация остается неизменной, корректной и аутентичной. Обеспечение целостности предполагает предотвращение и определение неавторизованного создания, модификации или удаления информации. Примером могут являться меры, гарантирующие, что почтовое сообщение не было изменено при пересылке.

Доступность – это гарантирование того, что авторизованные пользователи могут иметь доступ и работать с информационными активами, ресурсами и системами, которые им необходимы, при этом обеспечивается требуемая производительность. Обеспечение доступности включает меры для поддержания доступности информации, несмотря на возможность создания помех, включая отказ системы и преднамеренные попытки нарушения доступности. Примером может являться защита доступа и обеспечение пропускной способности почтового сервиса.

Структура правовой защиты информации

Правовая защита информации

Специальные правовые акты, правила, процедуры и мероприятия, обеспечивающие защиту информации на правовой основе



Законодательство РФ в области обеспечения ИБ

«О безопасности»	Доктрина ИБ РФ от 5 декабря 2016 г. , Указ Президента РФ от 5 декабря 2016 г. № 646. Закон РФ от 28 декабря 2010 года № 390-ФЗ
«О государственной тайне»	Закон РФ от 21 июля 1993 года № 5485-И(с изменениями и дополнениями от 6 октября 1997 г. № 131-ФЗ; от 2003 г. № 86-ФЗ; от 2003 г. № 153-ФЗ; от 2004 г. № 58-ФЗ; от 2004 г. № 122-ФЗ)
«О лицензировании отдельных видов деятельности»	Федеральный закон от 4 мая 2011 года № 99-ФЗ
Кодекс Российской Федерации об административных правонарушениях	От 30 декабря 2001 года № 195-ФЗ (выписка в части вопросов защиты информации) (с изменениями и дополнениями от 30 июня 2003 г. №86-ФЗ)
Уголовный кодекс Российской Федерации	От 13 июня 1996 года № 63-ФЗ (выписка в части вопросов защиты информации) (с изменениями и дополнениями от 8 декабря 2003 г. №162-ФЗ)
«Об электронной цифровой подписи»	Федеральный закон от 6 апреля 2011 года № 63-ФЗ
«О техническом регулировании»	Федеральный закон от 27 декабря 2002 года № 184-ФЗ (с изменениями и дополнениями от 9 мая 2005 г. № 45-ФЗ)
«Об информации, информационных технологиях и о защите информации»	Федеральный закон от 27 июля 2006 года № 149-ФЗ
«О персональных данных»	Федеральный закон от 27 июля 2006 года № 152-ФЗ
«О коммерческой тайне»	Федеральный закон от 29 июля 2004 года № 98-ФЗ

Система документов в области ЗИ

Правовые документы по технической защите информации

Конституция Российской Федерации

Федеральные законы (Законы Российской Федерации)

Указы и распоряжения Президента Российской Федерации

Постановления Правительства Российской Федерации

Организационно-распорядительные документы по технической защите информации

Концепции

Положения

Специальные нормативные документы по технической защите информации

Государственные стандарты

Специальные нормативные документы

Структура основных органов государственной власти РФ, решающих задачи в области ЗИ



Государственные ОВ, муниципальные ОВ, ЮЛ, ФЛ (граждане), ИП

Объекты и субъекты обеспечения ИБ

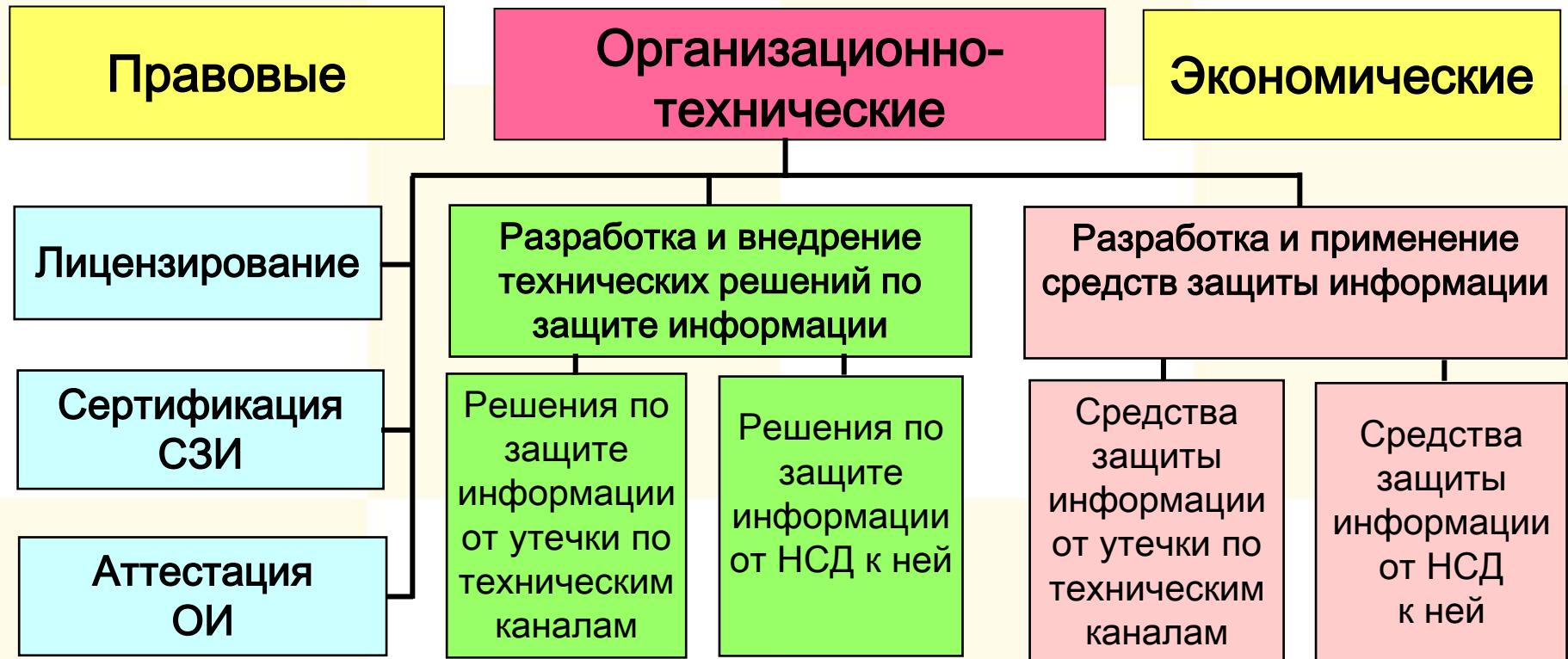
Объекты обеспечения ИБ в системах электронного документооборота

определяются тем, на что могут быть направлены действия злоумышленника:

- ✓ средства вычислительной техники (СВТ);
- ✓ данные, хранящиеся и обрабатывающиеся СВТ;
- ✓ технологии обработки данных;
- ✓ каналы передачи данных;
- ✓ знания и навыки администраторов и компетентных пользователей документооборота.

Субъект информационной безопасности — это активный участник процессов в деятельности обеспечения информационной безопасности, действующий на объект информационной безопасности независимо от характера этого действия: наносящего ущерб, разрушение или противодействующего этому.

Общая классификация методов обеспечения ИБ

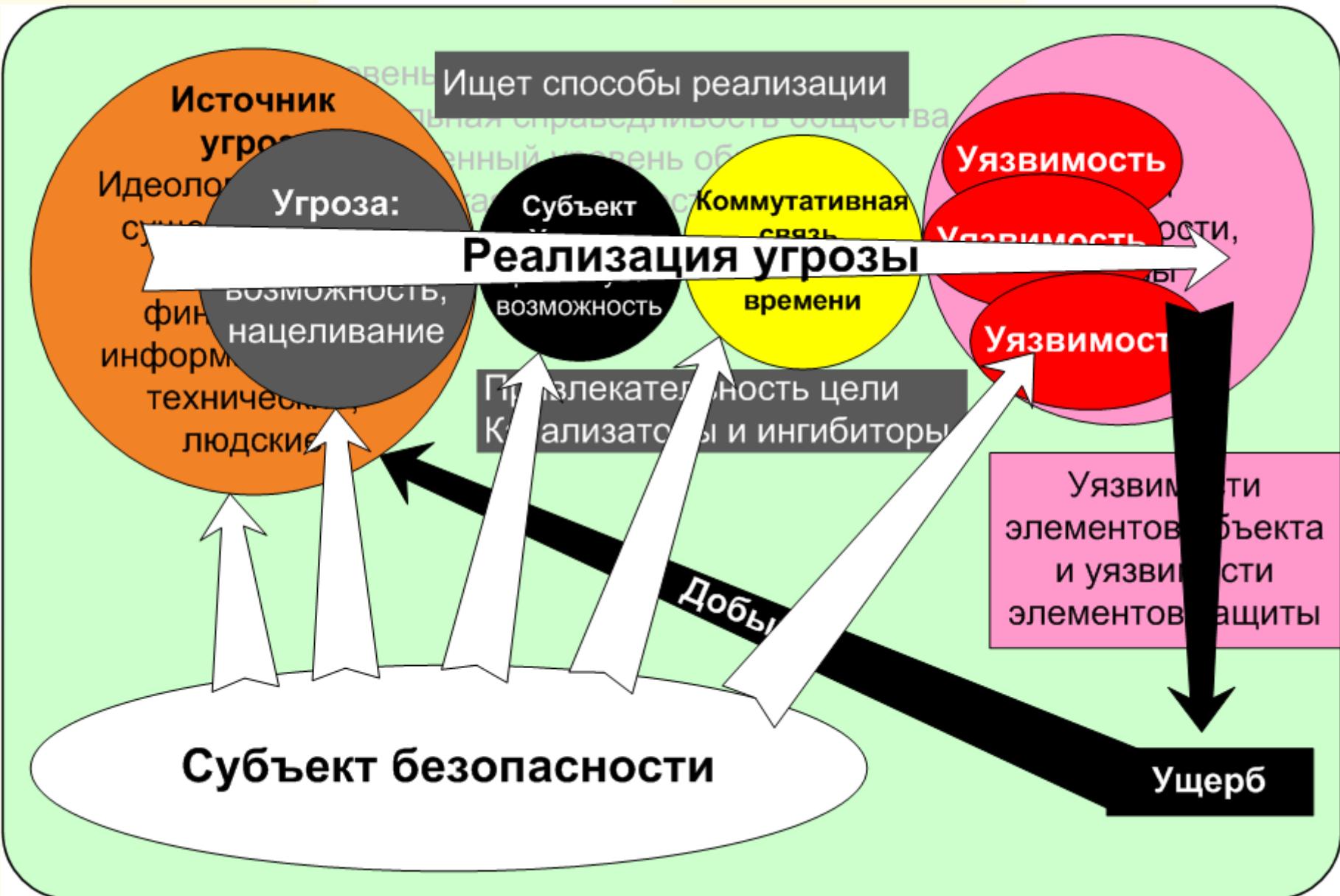


В соответствии с определением безопасности все методы противодействия угрозам сводятся к снижению вероятности неблагоприятного события и/или снижению ущерба от такого события.

Для реализации угрозы и нанесения ущерба активу должна сложиться последовательность событий:

1. Источник угрозы, исходя из своей идеологии, генерирует угрозы.
2. Угроза, как проект, получив компонент намерения, наращивает свои ресурсы, ищет уязвимости актива, оценивает привлекательность актива и готовит субъекта угрозы.
3. Субъект угрозы конкретизирует способ реализации угрозы, устанавливает коммутативные связи с уязвимостями и планирует необходимый интервал времени.

Для противодействия нужно разрушить систему, исключив любой элемент или фактор. Можно указать восемь методов противодействия угрозам, что может быть проиллюстрировано следующей структурной схемой



Теоретико-концептуальный и эмпирический подходы к обеспечению ИБ

Отличительной особенностью теоретико-концептуального подхода к защите информации является то, что на основе достижений концептуально-эмпирического подхода разрабатываются основы целостной теории защиты.

Т.о. подводится научно-методологическая база под теорию защиты информации.

При этом теория защиты, как и любая другая теория, определяется как совокупность основных идей в данной области знания.

Эмпирический подход к защите информации заключается в том, что

на основе анализа ранее проявлявшихся угроз информации и накапливаемого опыта защиты

разрабатываются и внедряются необходимые механизмы защиты.

Понятие угроз в сфере ИБ

Угроза информационной безопасности — совокупность условий и факторов, создающих опасность нарушения информационной безопасности.

Под угрозой понимается потенциально возможное событие, действие (воздействие), процесс или явление, которые могут привести к нанесению ущерба интересам владельцев информационных ресурсов и пользователей информационных технологий. Банк данных угроз ФСТЭК России содержит 221 актуальную угрозу, от **УБИ.001** (выявлена в 2000 году) до **УБИ.221** (выявлена в декабре 2020 года).

УБИ.221: Угроза модификации модели машинного обучения путем искажения («отравления») обучающих данных

Описание Угроза заключается в возможности модификации (искажения) **угрозы** модели машинного обучения, используемой в информационной (автоматизированной) системе, реализующей технологии искусственного интеллекта.

Данная угроза обусловлена:

- недостатками реализации процесса машинного обучения;
- недостатками устройства алгоритмов машинного обучения.

Реализация данной угрозы возможна при наличии у нарушителя возможности воздействовать на процесс машинного обучения

Источники • Внутренний нарушитель со средним потенциалом
угрозы • Внешний нарушитель с высоким потенциалом

Объект Программное обеспечение (программы), использующее
воздействия машинное обучение; модели машинного обучения; обучающие данные машинного обучения

Последствия • Нарушение целостности
реализации
угрозы

Метод снижения возможности угрозы

Снижение возможности реализации угрозы.
Точнее затруднение реализации выполнения угрозы и поэтому снижение привлекательности объекта угрозы для субъекта угрозы.

$$R = P_A \cdot (1 - P_E) \cdot C$$

Где P_A - вероятность наличия угрозы,

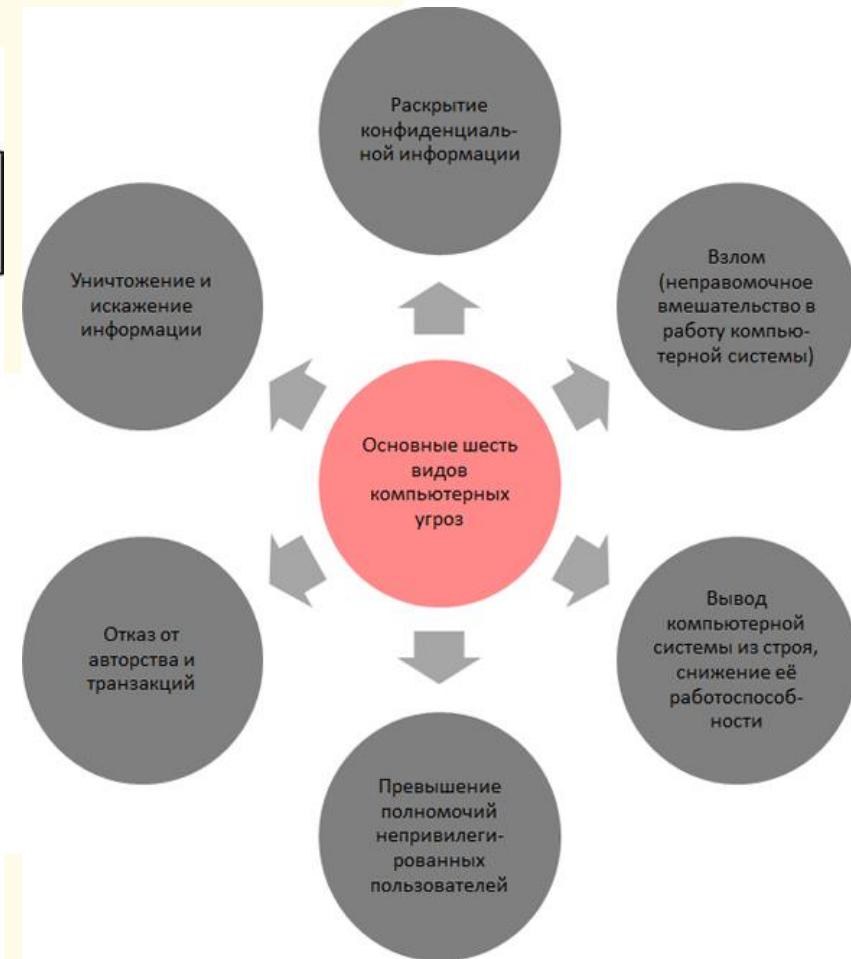
P_E - эффективность системы безопасности,

C - ущерб от атаки.

Это может быть достигнуто следующими способами:

- защита расстоянием,
- защита временем.

Классификация угроз



Угрозы ИБ являются целями деятельности нарушителей информационной безопасности.

При выполнении проектов по построению/усовершенствованию систем информационной безопасности специалисты определяют профиль всех актуальных угроз ИБ и на основе полученных данных разрабатывают необходимые технические и организационные меры по минимизации существующих угроз.

Базовая модель угроз ИБ

Модель угроз информационной безопасности – это описание существующих угроз ИБ, их актуальности, возможности реализации и последствий.

Стандарт СТО БР ИББС – 1.0-2010 определяет модель угроз информационной безопасности следующим образом: это «описание источников угроз ИБ; методов реализации угроз ИБ; объектов, пригодных для реализации угроз ИБ; уязвимостей, используемых источниками угроз ИБ; типов возможных потерь (например, нарушение доступности, целостности или конфиденциальности информационных активов); масштабов потенциального ущерба».

Адекватные модели угроз информационной безопасности позволяют выявить существующие угрозы, разработать эффективные контрмеры, повысив тем самым уровень ИБ, и оптимизировать затраты на защиту (сфокусировав её на актуальных угрозах).



В модели должны учитываться все актуальные угрозы на всех стадиях их жизненного цикла
У различных информационных систем, а также объектов одной информационной системы может быть разный спектр угроз, определяемый особенностями конкретной информационной системы и её объектов и характером возможных действий источника угрозы.

При построении таких моделей специалисты используют каталоги и перечни угроз, содержащиеся в официальных стандартах информационной безопасности и методических документах ФСТЭК и ФСБ России, а также списки угроз, выявленных при проведении аудита информационной системы заказчика.

Метод защиты расстоянием

Защита расстоянием разрушает коммутативную связь.

Она заключается в пространственном разделении сферы существования объектов безопасности (охраняемых ценностей) и сферы существования угроз, рассчитана на выявление внешнего нарушителя и подразумевает зональный принцип построения объекта.

В зоны, где размещаются объекты безопасности, допускается только тот персонал, который имеет отношение к производимым работам и учёту материальных ценностей. Если работы не производятся, то не допускается никто.

Этот метод защиты обеспечивается объёмно-планировочными решениями, физическими барьерами, мероприятиями контроля доступа и контролируется системой охранно-тревожной сигнализации с верификацией оханным телевидением. Силовые подразделения выполняют задачу парирования и пресечения попытки проникновения.

Защита расстоянием снижает и вероятность и величину возможного ущерба.

Метод защиты временем

Задача временем лишает субъект угрозы необходимого времени для реализации угрозы.

Это есть наиболее раннее обнаружение противоправного действия и немедленное реагирование силовых структур (субъекта безопасности). При этом может применяться блокирование зоны расположения предметов защиты.

Этот способ защиты реализуется в самом простом случае присутствием наблюдающего охранника, который сам же и производит силовые действия или тревожно-вызывной сигнализацией, либо объектовой охранной сигнализацией и/или современными средствами видеоаналитики.

Силовые подразделения выполняют задачу пресечения противоправного действия.

Задача временем снижает вероятность неблагоприятного события

Угрозы НСД к информации

Несанкционированный доступ к информации - **доступ к информации, нарушающий правила разграничения доступа штатными средствами вычислительной техники или автоматизированных систем.**

Под штатными средствами понимается совокупность программного, микропрограммного и технического обеспечения средств вычислительной техники или автоматизированных систем. К основным угрозам НСД можно отнести следующее:

- угрозы проникновения в *операционную среду* компьютера с использованием штатного программного обеспечения (средств операционной системы или прикладных программ общего применения);
- угрозы создания нештатных режимов работы программных (программно-аппаратных) средств за счет преднамеренных изменений служебных данных, игнорирования предусмотренных в штатных условиях ограничений на состав и характеристики обрабатываемой информации, искажения (модификации) самих данных и т.п.;
- угрозы внедрения *вредоносных программ* (программно-математического воздействия).

Кроме этого, возможны комбинированные угрозы, представляющие собой сочетание указанных угроз. Например, за счет внедрения *вредоносных программ* могут создаваться условия для НСД в *операционную среду* компьютера.

В общем случае, комплекс программно - технических средств и организационных мер по защите информации от НСД реализуется в рамках системы защиты информации от НСД (СЗИ НСД), условно состоящей из следующих четырех подсистем:

- управления доступом;
- регистрации и учета;
- криптографической;
- обеспечения целостности.

Источником угрозы НСД может быть **нарушитель, носитель с вредоносной программой или аппаратная закладка.**

Защита информации от НСД в АС

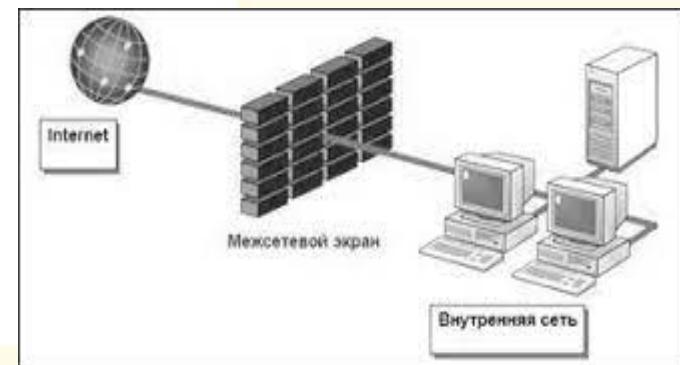
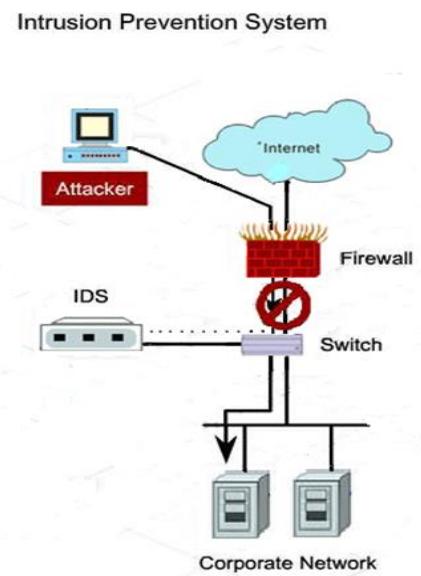
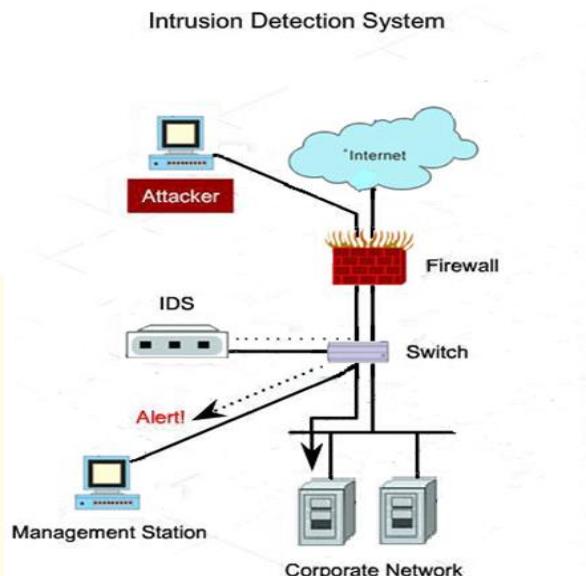
Устанавливается *девять классов защищенности АС от НСД к информации*. Каждый класс характеризуется определенной минимальной совокупностью требований по защите. Классы подразделяются *на три группы, отличающиеся особенностями обработки информации в АС*. В пределах каждой группы соблюдается иерархия требований по защите в зависимости от ценности (конфиденциальности) информации и, следовательно, иерархия классов защищенности АС.

Третья группа включает АС, в которых работает **один пользователь, допущенный ко всей информации АС, размещенной на носителях одного уровня конфиденциальности**. Группа содержит два класса - 3Б и 3А. Вторая группа включает АС, в которых **пользователи имеют одинаковые права доступа (полномочия) ко всей информации АС, обрабатываемой и (или) хранимой на носителях различного уровня конфиденциальности**. Группа содержит два класса - 2Б и 2А. Первая группа включает **многопользовательские АС, в которых одновременно обрабатывается и (или) хранится информация разных уровней конфиденциальности. Не все пользователи имеют право доступа ко всей информации АС**. Группа содержит пять классов - 1Д, 1Г, 1В, 1Б и 1А.

В общем случае, комплекс программно-технических средств и организационных (процедурных) решений по защите информации от НСД реализуется в рамках системы защиты информации от НСД (СЗИ НСД), условно состоящей из следующих *четырех подсистем*: - **управления доступом**; - **регистрации и учета**; - **криптографической**; - **обеспечения целостности**.

Система предотвращения вторжений (англ. Intrusion Prevention System, IPS) – программная или аппаратная система сетевой и компьютерной безопасности, обнаруживающая вторжения или нарушения безопасности и автоматически защищающая от них.

Системы IPS можно рассматривать как расширение Систем обнаружения вторжений (IDS), так как задача отслеживания атак остается одинаковой. Однако, они отличаются в том, что IPS должна отслеживать активность в реальном времени и быстро реализовывать действия по предотвращению атак. **Межсетевой экран, сетевой экран — программный или программно-аппаратный элемент компьютерной сети, осуществляющий контроль и фильтрацию проходящего через него сетевого трафика в соответствии с заданными правилами.**



Понятие о политике ИБ

Политика информационной безопасности (ПИБ) – совокупность руководящих принципов, правил, процедур и практических приёмов в области безопасности, которые направлены на защиту ценной информации.

Назначение ПИБ :

- формулирование целей и задач информационной безопасности организации с точки зрения бизнеса (позволяет определить это для руководства и продемонстрировать поддержку руководством значимости ИБ),
- определение правил организации работы в компании для минимизации рисков информационной безопасности и повышения эффективности бизнеса.

Требования к ПИБ:

- Политика должна быть утверждена высшим административным органом компании (генеральный директор, совет директоров и т.п.) для демонстрации поддержки руководства.
- Политика ИБ должна быть написана понятным языком для конечных пользователей и руководства компании и быть по возможности краткой.
- Политика ИБ должна определять цели ИБ, способы их достижения и ответственность. Технические подробности реализации способов содержатся в инструкциях и регламентах, на которые даются ссылки в Политике.
- Минимизация влияния политики безопасности на производственный процесс.
- Непрерывность обучения сотрудников и руководства организации в вопросах обеспечения ИБ
- Непрерывный контроль выполнения правил политики безопасности на этапе внедрения и в дальнейшем.
- Постоянное совершенствование политик безопасности.
- Согласованность во взглядах и создание корпоративной культуры безопасности.

Задача защиты персональных данных

Персональные данные - любая информация, относящаяся к определенному или определяемому на основании такой информации физическому лицу (субъекту персональных данных), в том числе его *фамилия, имя, отчество, год, месяц, дата и место рождения, адрес, семейное, социальное, имущественное положение, образование, профессия, доходы*, другая информация.

Защита персональных данных – это комплекс мероприятий, позволяющий выполнить требования законодательства РФ, касающиеся обработки, хранения и передачи персональных данных граждан.

Оператор персональных данных - государственный орган, муниципальный орган, юридическое или физическое лицо, организующие и (или) осуществляющие обработку персональных данных, а также определяющие цели и содержание обработки персональных данных, в том числе *представленных в электронном виде*.

Электронный документ (ЭД)

Электронный документ – это документированная информация, представленная в электронной форме, то есть в виде, пригодном для восприятия человеком с использованием электронных вычислительных машин, а также для передачи по информационно-телекоммуникационным сетям или обработки в информационных системах ([п. 11.1 ст. 2 Федерального закона](#) от 27.07.2006 № 149-ФЗ "Об информации, информационных технологиях и о защите информации"). При этом для электронного документа характерны:

- аутентичность - свойство электронного документа, гарантирующее, что электронный документ идентичен заявленному;
- достоверность - свойство электронного документа, при котором содержание электронного документа является полным и точным представлением подтверждаемых операций, деятельности или фактов и которому можно доверять в последующих операциях или в последующей деятельности;
- целостность - состояние электронного документа, в который после его создания не вносились никакие изменения;
- пригодность для использования - свойство электронного документа, позволяющее его локализовать и воспроизвести в любой момент времени.

Исходя из этого, можно сделать вывод, что электронный документ - это любой документ, который представлен в электронном виде, в том числе это может быть скан-образ документа, файл, набранный в текстовом редакторе, и т.п.

При этом электронные документы могут быть формализованными, т.е. составленными в таком виде, который позволяет с помощью программных средств распознавать их содержимое, и неформализованными (например, скан-копия).

Обеспечение защиты персональных данных

Согласно требованию Федерального закона №152 от 27 июля 2006 г. "О персональных данных", а так же Федерального Закона №261 от 25 июля 2011 года "О внесении изменений в федеральный закон "О персональных данных" оператор персональных данных обязан выполнить ряд организационных и технических мер касающихся процессов обработки персональных данных.

- а) проведение мероприятий, направленных на предотвращение несанкционированного доступа к персональным данным и (или) передачи их лицам, не имеющим права доступа к такой информации;
- б) своевременное обнаружение фактов несанкционированного доступа к персональным данным;
- в) недопущение воздействия на технические средства автоматизированной обработки персональных данных, в результате которого может быть нарушено их функционирование;
- г) возможность незамедлительного восстановления персональных данных, модифицированных или уничтоженных вследствие несанкционированного доступа к ним;
- д) постоянный контроль за обеспечением уровня защищенности персональных данных. (Назначение ответственного лица)

Классификация информационных систем персональных данных

В соответствии с Постановлением Правительства № 1119 от 1 ноября 2012г. «Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных» для ИСПДн установлено 3 уровня информационных угроз.

Для определения уровня защищенности необходимо определить тип информационной системы и актуальные угрозы.

По типам информационные системы делятся на:

- Обрабатывающие специальные категории персональных данных;
- Обрабатывающие биометрические персональные данные;
- Обрабатывающие общедоступные персональные данные;
- Обрабатывающие иные категории персональных данных;
- Обрабатывающие персональные данные сотрудников оператора.

Актуальные угрозы делятся на следующие типы:

- Угрозы 1-го типа актуальны для информационной системы, если для нее в том числе актуальны угрозы, связанные с наличием недокументированных (недекларированных) возможностей **в системном программном обеспечении, используемом в информационной системе**;
- Угрозы 2-го типа актуальны для информационной системы, если для нее в том числе актуальны угрозы, связанные с наличием недокументированных (недекларированных) возможностей **в прикладном программном обеспечении, используемом в информационной системе**;
- Угрозы 3-го типа актуальны для информационной системы, если для нее актуальны угрозы, не связанные с наличием недокументированных (недекларированных) возможностей **в системном и прикладном программном обеспечении, используемом в информационной системе**;

Электронная подпись (ЭП)

Электронная подпись – это реквизит документа, полученный с помощью криптографических преобразований информации и позволяющий подтвердить принадлежность ЭП ее владельцу, а также зафиксировать состояние информации/данных (наличие, либо отсутствие изменений) в электронном документе с момента его подписания. Подпись связана как с автором, так и с самим документом с помощью криптографических методов, и не может быть подделана с помощью обычного копирования.

Сокращенное название (согласно Федеральному Закону № 63) – ЭП, но чаще используют устаревшую аббревиатуру ЭЦП (электронная цифровая подпись). Это, например, облегчает взаимодействие с поисковиками в интернете, так как ЭП может также означать электрическую плиту, электровоз пассажирский и т.д.

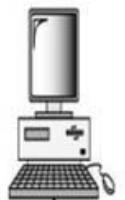
Согласно законодательству РФ, **квалифицированная электронная подпись** – это эквивалент подписи, проставляемой «от руки», обладающей полной юридической силой. Помимо квалифицированной в России представлены еще два вида ЭЦП:

- неквалифицированная – обеспечивает юридическую значимость документа, но только после заключения дополнительных соглашений между подписантами о правилах применения и признания ЭЦП, позволяет подтвердить авторство документа и проконтролировать его неизменность после подписания;
- простая – не придает подписанному документу юридическую значимость до заключения дополнительных соглашений между подписантами о правилах применения и признания ЭЦП и без соблюдении законодательно закрепленных условий по ее использованию.

Криптосредства и стеганография в ИБ

Криптографические схемы передачи информации

Передатчик (Алиса)



Открытый текст



Шифр

Приёмник (Боб)



Открытый текст



СКЗИ

Системы аутентификации электронных данных

Системы идентификации и аутентификации пользователей

Системы шифрования дисковых данных

Средства управления ключевой информацией

Системы шифрования данных, передаваемых по сети



Средства криптографической защиты информации (СКЗИ) – аппаратные, программные и аппаратно-программные средства, системы и комплексы, реализующие алгоритмы криптографического преобразования информации. Шифрование по ГОСТ 28147-89. Вычисление/проверка ЭП по ГОСТ Р 34.10-94

Метод замены наименее значащего бита

Представим пиксель тремя байтами в битовом виде:

1	1	0	0	0	0	1	1
0	0	1	0	0	0	0	0
0	1	1	0	0	1	1	0

R — 195

G — 32

B — 102



Младшие биты (выделены бледным, справа) дают незначительный вклад в изображение по сравнению со старшими. Замена одного или двух младших бит для человеческого глаза будет почти незаметна.

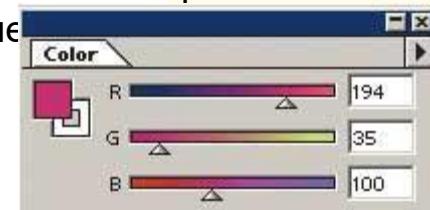
Пусть необходимо в этом пикселе скрыть 6 бит — 101100. Разделим их на 3 пары и заменим этими парами младшие биты в каждом канале

1	1	0	0	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0

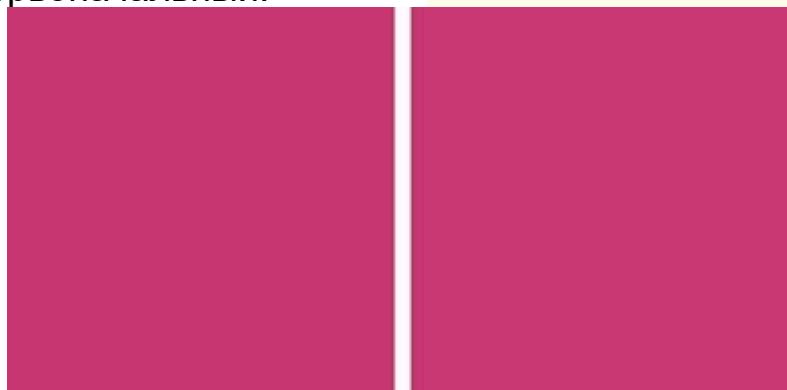
R — 194

G — 35

B — 100



Получили новый цвет, очень похожий на первоначальный:



Право на доменное имя

Приведем пример адреса сайта нашего Университета: www.bmstu.ru Часть „ru" – это домен первого уровня, также называемая "зоной". Часть „bmstu" – домен второго уровня. Часть "www" – домен третьего уровня или субдомен. Как правило (по умолчанию), www можно не набирать, и отображаются одинаковые сайты как под доменными именами с www, так и без них, хотя **технически есть возможность настроить их по-разному.** Так, **долгие годы сайт Федеральной налоговой службы отображался только по адресу www.nalog.ru, но не отображался под доменным именем nalog.ru (сейчас ситуация уже исправлена).** Доменные имена давно стали полноценным объектом гражданского оборота: они продаются, сдаются в аренду, развиваются системы субдоменов, которые также могут выступать объектами оборота (например, в популярных географических зонах spb.ru, msk.ru и др.). Несмотря на то, что рынку доменных имен более 20 лет, нет четкой определенности в отношении природы прав на доменное имя. Многие до сих пор отрицают оборотоспособность доменных имен, их самостоятельный правовой характер (в том числе и ряд судей). В то же время, немало судебных решений, в которых признаются действительными сделки по передаче доменных имен в пользование (аренду), по продаже доменных имен.

Спекуляция доменными именами (англ. *domaining*, от *domain* — домен) — сложившаяся практика идентификации и регистрации или приобретения доменных имён Интернета для последующей их перепродажи с целью получения прибыли.

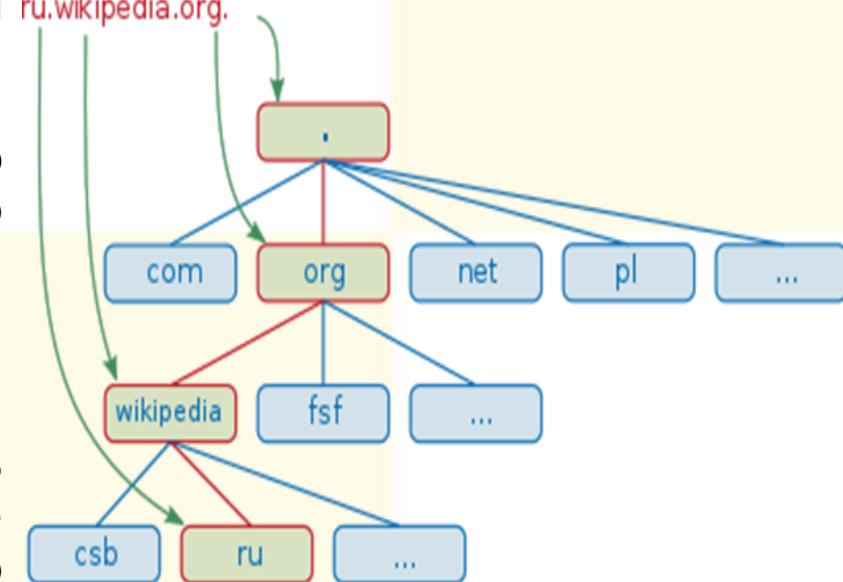
Киберсквоттинг (англ. *cybersquatting*) — регистрация доменных имён, содержащих торговую марку, принадлежащую другому лицу с целью их дальнейшей перепродажи или недобросовестного использования.

Система доменных имен

Домены - **domains** (доменные имя, доменные адреса) в интернете — это позывные для сайтов. Набрав в строке браузера определенную комбинацию, например, «bmstu.ru» (это и есть **домен**), мы попадём на **сайт**: www.bmstu.ru

Домен — «название» сайта. Слова «домен» и «сайт» часто путают, но это не одно и то же. Сайт — это веб-страницы, которые отображаются в интернете, т. е. контент. А домен сайта — это его уникальный «адрес». Если у сайта не будет домена, пользователи просто не найдут к нему дорогу и не увидят содержимое.

DNS (англ. Domain Name System «**система доменных имён**») — компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты, обслуживающих узлах для протоколов в домене (SRV-запись). Распределённая база данных DNS поддерживается с помощью иерархии DNS-серверов, взаимодействующих по определённому протоколу.



Инциденты ИБ, предпосылки их возникновения

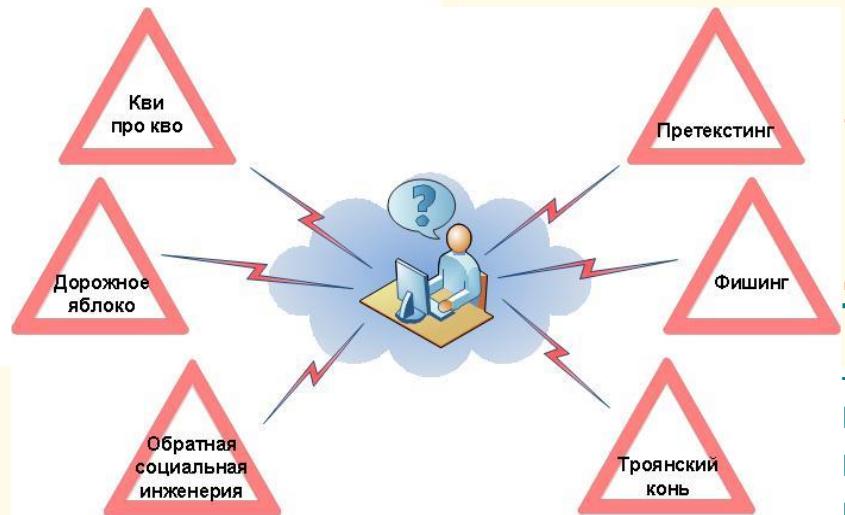
Международные регламенты, действующие в сфере сертификации менеджмента информационных систем, дают свое определение этому явлению. Согласно им инцидентом информационной безопасности является единичное событие нежелательного и непредсказуемого характера, которое способно повлиять на бизнес-процессы компании или отдельной личности, скомпрометировать их или нарушить ИБ. Среди основных типов событий присутствуют:

- нарушение порядка взаимодействия с Интернет-провайдерами, хостингами, почтовыми сервисами, облачными сервисами и другими поставщиками телекоммуникационных услуг;
- отказ оборудования по любым причинам;
- нарушение работы программного обеспечения;
- нарушение правил обработки, хранения, передачи информации, как электронной, так и документов;
- неавторизованный или несанкционированный доступ третьих лиц к информационным ресурсам;
- выявление внешнего мониторинга ресурсов;
- выявление вирусов или других вредоносных программ;
- любая компрометация системы, например, попадание пароля от учетной записи в открытый доступ.

Виды социальной инженерии

Социальная инженерия – метод получения необходимого доступа к информации, основанный на особенностях психологии людей. Основной целью социальной инженерии является получение доступа к конфиденциальной информации, паролям, банковским данным и другим защищенным системам.

Претекстинг - это набор действий, отработанных по определенному, заранее составленному сценарию, в результате которого жертва может выдать какую-либо информацию или совершить определенное действие



Дорожное яблоко – этот метод представляет собой адаптацию троянского коня и состоит в использовании физических носителей (CD, флэш-накопителей).

Фишинг – техника интернет-мошенничества, направленная на получение конфиденциальной информации пользователей - авторизационных данных различных систем.

Кви про кво (услуга за услугу) – данная техника предполагает обращение злоумышленника к цели. Злоумышленник может представиться, например, сотрудником технической поддержки, информировать о возникновении технических проблем на рабочем месте и вызвать желательные для себя действия.

Троянский конь – это техника основывается на любопытстве, страхе или других эмоциях пользователей. Злоумышленник отправляет письмо жертве по электронной почте, во вложении которого находится «полезная информация»

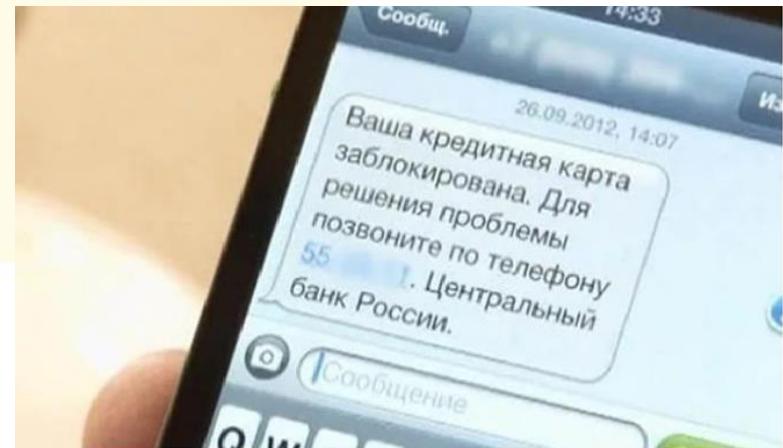
Обратная социальная инженерия - данный вид атаки направлен на создание такой ситуации, при которой жертва вынуждена будет сама обратится к злоумышленнику за «помощью».

Как определить атаку социального инженера

- представление себя другом-сотрудником либо новым сотрудником с просьбой о помощи;
- представление себя сотрудником поставщика, партнерской компании, представителем закона;
- представление себя кем-либо из руководства;
- представление себя поставщиком или производителем операционных систем, звонящим, чтобы предложить обновление жертве для установки;
- предложение помочь в случае возникновения проблемы и последующее провоцирование возникновения проблемы, которое принуждает жертву попросить о помощи;
- использование внутреннего сленга и терминологии для возникновения доверия;
- отправка вируса или троянского коня в качестве приложения к письму;
- использование фальшивого всплывающих окна, с просьбой аутентифицироваться еще раз, или ввести пароль;
- предложение приза за регистрацию на сайте с именем пользователя и паролем;
- записывание клавиш, которые жертва вводит на своём компьютере или в своей программе (кейлоггинг);
- подбрасывание различных носителей данных (флэш-карт, дисков и т. д.) с вредоносным ПО на стол жертвы;
- подброс документа или папки в почтовый отдел компании для внутренней доставки;
- видоизменение надписи на факсе, чтобы казалось, что он пришел из компании;
- просьба секретаря принять, а затем отослать факс;
- просьба отослать документ в место, которое кажется локальным (то есть находится на территории организации);
- подстройка голосовой почты, чтобы работники, решившие перезвонить, подумали, что атакующий — их сотрудник;

Фишинг и редирект

Фишинг ([англ.](#) *phishing* от *fishing* «рыбная ловля, выуживание») — вид компьютерного мошенничества. Целью фишинга является получение доступа к конфиденциальным данным пользователей — логинам и паролям. Это достигается путём проведения массовых рассылок электронных писем от имени популярных брендов, а также личных сообщений внутри различных сервисов, например, от имени банков или внутри социальных сетей. Фишинг по телефону — **вишинг (vishing)**.



В письме часто содержится прямая ссылка на сайт, внешне неотличимый от настоящего, либо на сайт с [редиректом](#). **Редирект** — автоматическое перенаправление пользователей с одного сайта на другой. Выглядит это следующим образом — пользователь набирает в адресной строке браузера один адрес, а оказывается на сайте, адрес которого совсем другой. Кроме собственно процесса, редиректом называется скрипт, выполняя который, браузер перенаправляет пользователя на другой сайт. После того как пользователь попадает на поддельную страницу, мошенники пытаются различными психологическими приёмами побудить пользователя ввести на поддельной странице свои логин и пароль, которые он использует для доступа к определённому сайту, что позволяет мошенникам получить доступ к аккаунтам и банковским счетам.



Борьба с кибермошенничеством

Преступления в сфере информационных технологий включают как распространение вредоносных вирусов, взлом паролей, кражу номеров банковских карт и других банковских реквизитов, фишинг, так и распространение противоправной информации через интернет, а также вредоносное вмешательство через компьютерные сети в работу различных систем.

Основные правила защиты от сетевых угроз:

- крайне осторожно относитесь к программам и документам, изготовленных с помощью пакета Microsoft Office, которые вы получаете из глобальных сетей. Перед тем, как запустить файл на выполнение или открыть документ/таблицу/презентацию/базу данных, обязательно проверьте их на наличие вирусов;
- для уменьшения риска заразить файл на сервере администраторам сетей следует активно использовать стандартные возможности защиты сети: ограничение прав пользователей; установку атрибутов "только на чтение" или даже "только на запуск" для всех выполняемых файлов; скрытие (закрытие) важных разделов диска и директорий и т.д.;
- приобретать дистрибутивные копии программного обеспечения у официальных продавцов;
- не запускайте непроверенные файлы, особенно полученные из сети. Желательно использовать только программы, полученные из надежных источников. Перед запуском новых программ обязательно проверьте их одним или несколькими антивирусами;
- пользуйтесь утилитами проверки целостности информации. Такие утилиты сохраняют в специальных базах данных информацию о системных областях дисков (или целиком системные области) и информацию о файлах (контрольные суммы, размеры, атрибуты, даты последней модификации файлов и т.д.).
- Периодически сохраняйте на внешнем носителе файлы (backup-копии), с которыми ведется работа.

Технические каналы реализации угроз

При реализации угроз безопасности злоумышленник может воспользоваться самыми различными *каналами реализации угроз* – каналами НСД, каналами утечки. Под *каналом утечки информации* понимают совокупность источника информации, материального носителя или среды распространения, несущего указанную информацию сигнала и средства выделения информации из сигнала или носителя. Средство выделения информации из сигнала или носителя может располагаться в пределах контролируемой зоны, охватывающей АСОИ или вне ее.

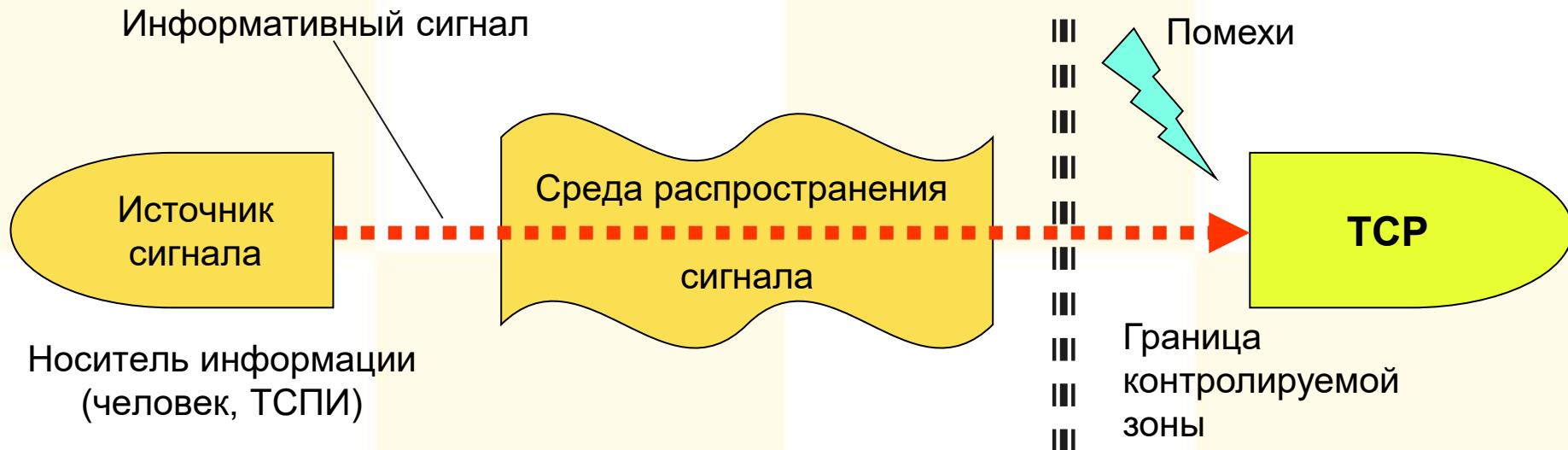
Применительно к АСОИ выделяют следующие основные каналы утечки информации [4]:

1. Электромагнитный канал.
2. Вибраакустический канал.
3. Визуальный канал .
4. Информационный канал.

Не останавливаясь на первых каналах утечки информации, рассмотрим последний, который связан с возможностью локального или удаленного доступа злоумышленника к элементам АСОИ, к носителям информации, к программному обеспечению, к линиям связи. Данный канал условно может быть разделен на следующие каналы:

- канал коммутируемых линий связи,
- канал выделенных линий связи,
- канал локальной сети,
- канал машинных носителей информации,
- канал терминальных и периферийных устройств.

Угрозы утечки информации по техническим каналам



Технический канал утечки информации

совокупность носителя информации, технического средства, с помощью которого осуществляется перехват информации, и физической среды распространения информативного сигнала.

Носитель информации:

Материальный объект, в том числе физическое поле, в котором информация находит свое отображение в виде символов, образов, сигналов, технических решений и процессов, количественных характеристик физических величин

Информативный сигнал

сигнал, по параметрам которого может быть определена защищаемая информация.

Классификация технических каналов утечки информации

Технические каналы утечки информации

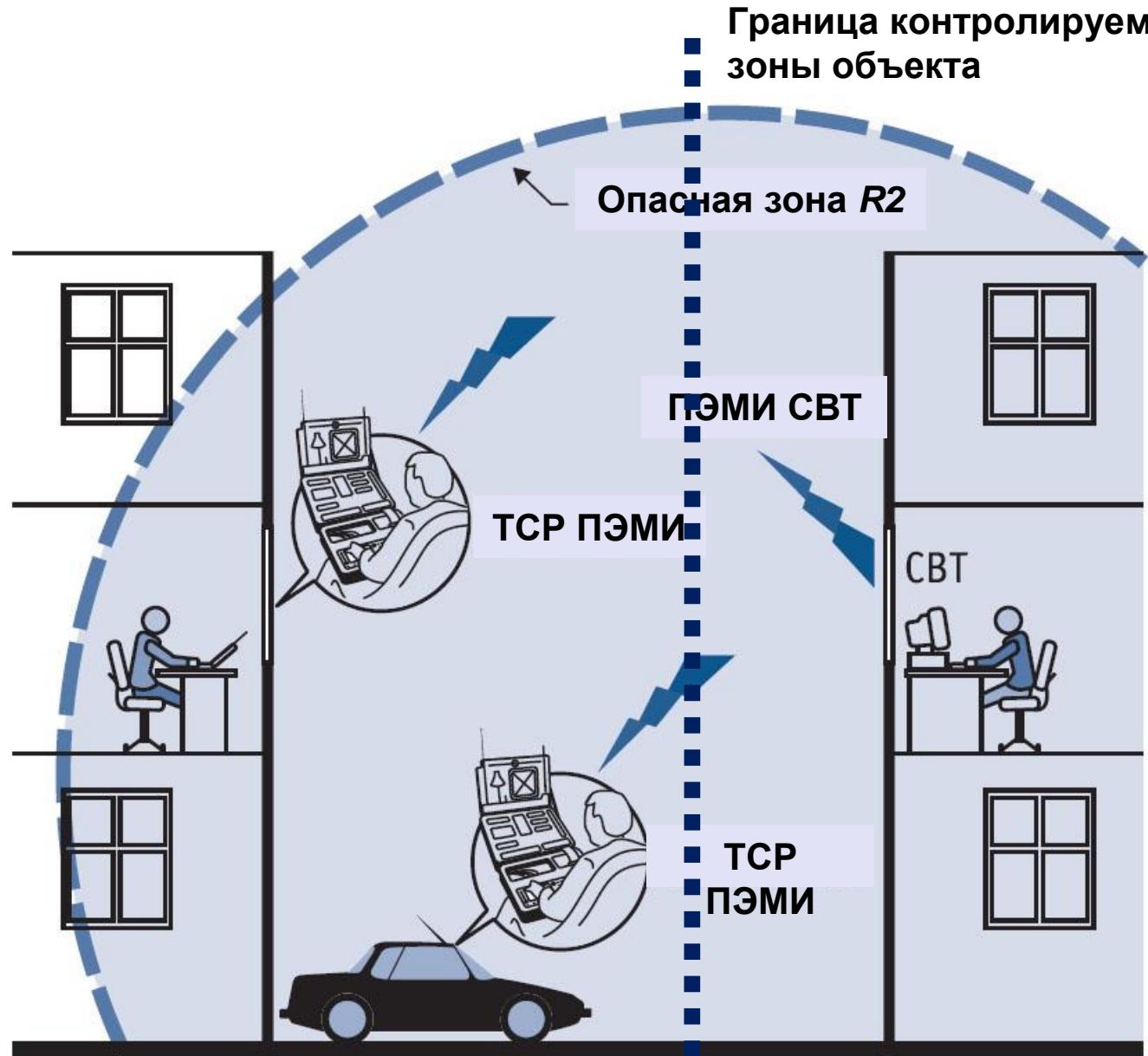
Технические каналы утечки информации, обрабатываемой техническими средствами

- возникающие за счет ПЭМИ;
- возникающие за счет наводок ПЭМИ;
- создаваемые путем «высокочастотного облучения» ТСПИ.
- создаваемые путем внедрения в ТСПИ закладных устройств

Технические каналы утечки речевой информации из выделенных помещений

- прямые акустические;
- акустовибрационные;
- акустооптический (лазерный);
- акустоэлектрические;
- акустоэлектромагнитные.

Технические каналы утечки видовой информации (скрытое видеонаблюдение и съемка)



Перехват побочных электромагнитных излучений ТСПИ средствами разведки
ПЭМИН (электромагнитный канал утечки информации) 43

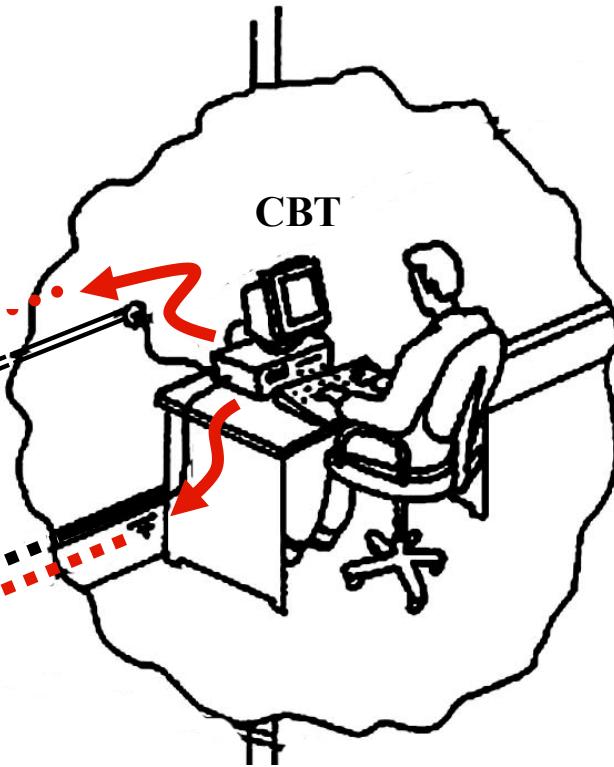
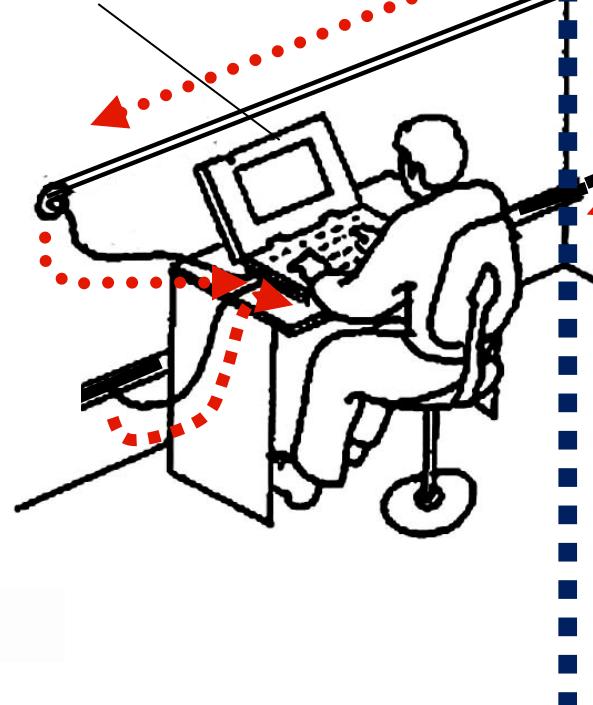
**Перехват наведенных
информационных сигналов с
инженерных коммуникаций**



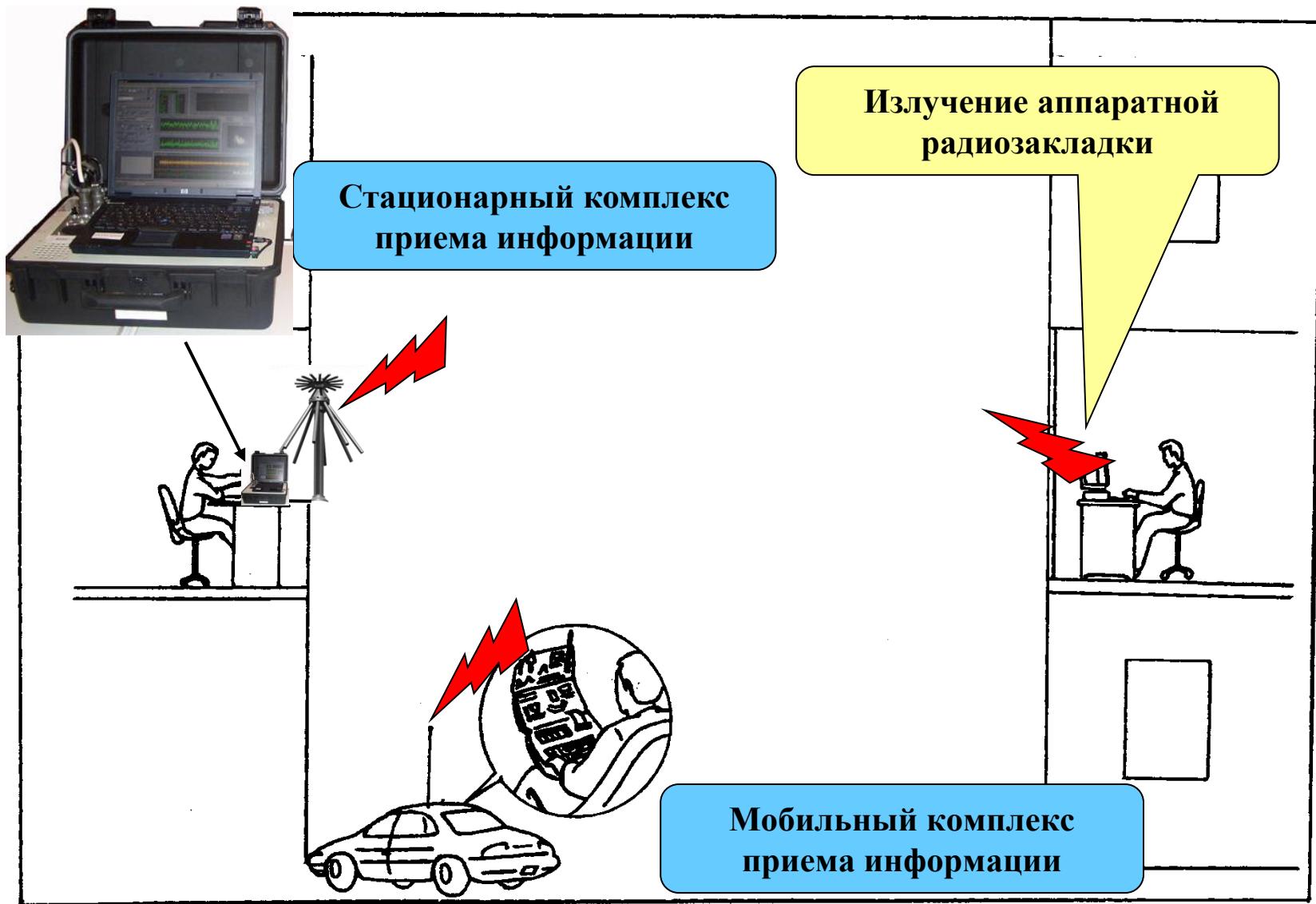
Граница контролируемой зоны объекта

Наводки информативных сигналов в цепях электропитания СВТ

TCP ПЭМИН



Перехват наведенных информационных сигналов с цепей заземления и электропитания ТСПИ



Перехват информации путем внедрения СВТ электронных устройств перехвата информации (закладных устройств)

Методы и способы защиты информации от утечки по техническим каналам

- ☞ проведение организационно-режимных мероприятий;
- ☞ использование специальных технических средств защиты ТСПИ;
- ☞ выявление электронных устройств перехвата информации (закладных устройств), внедренных в ТСПИ.

Организационное мероприятие – это мероприятие по защите информации, проведение которого не требует применения специально разработанных технических средств защиты.

Техническое мероприятие – это мероприятие по защите информации, предусматривающее применение специальных технических средств, а также реализацию технических решений.

Выявление закладных устройств осуществляется проведением специальных обследований объектов ТСПИ, а также специальных проверок ТСПИ.

Специальные обследования объектов ТСПИ проводятся путём визуального осмотра помещений и ТСПИ без применения технических средств.

Специальная проверка ТСПИ проводится с использованием специальных технических средств.

Способы перехвата речевой информации из выделенных помещений по прямому акустическому каналу

Без проникновения в пределы контролируемой зоны (КЗ) объекта

Перехват акустических колебаний, возникающих при ведении разговоров, **направленными микрофонами**, размещенными за пределами КЗ в ближайших строениях или транспортных средствах

С проникновением в пределы контролируемой зоны (КЗ) объекта

Прослушивание разговоров без применения технических средств (из-за недостаточной звукоизоляции ограждающих конструкций выделенных помещений и их инженерно-технических систем) **посторонними лицами**, при их нахождении в коридорах или смежных помещениях (**непреднамеренное прослушивание**)

Перехват акустических колебаний, возникающих при ведении разговоров, скрытно установленными в выделенных помещениях **закладными устройствами**

(с датчиками микрофонного типа), передающими информацию по:

- радиоканалу;
- оптическому каналу (в ИК-диапазоне);
- специально проложенному кабелю;
- сети электропитания 220 В;
- телефонной линии и т.д.

Запись речевой информации **цифровыми диктофонами**, скрытно установленными в выделенных помещениях

Защита объектов ТСПИ от утечки информации, возникающей **за счет побочных электромагнитных излучений**, достигается: экранированием и заземлением технических средств и их соединительных линий, экранированием помещений, а также использованием систем пространственного электромагнитного зашумления.

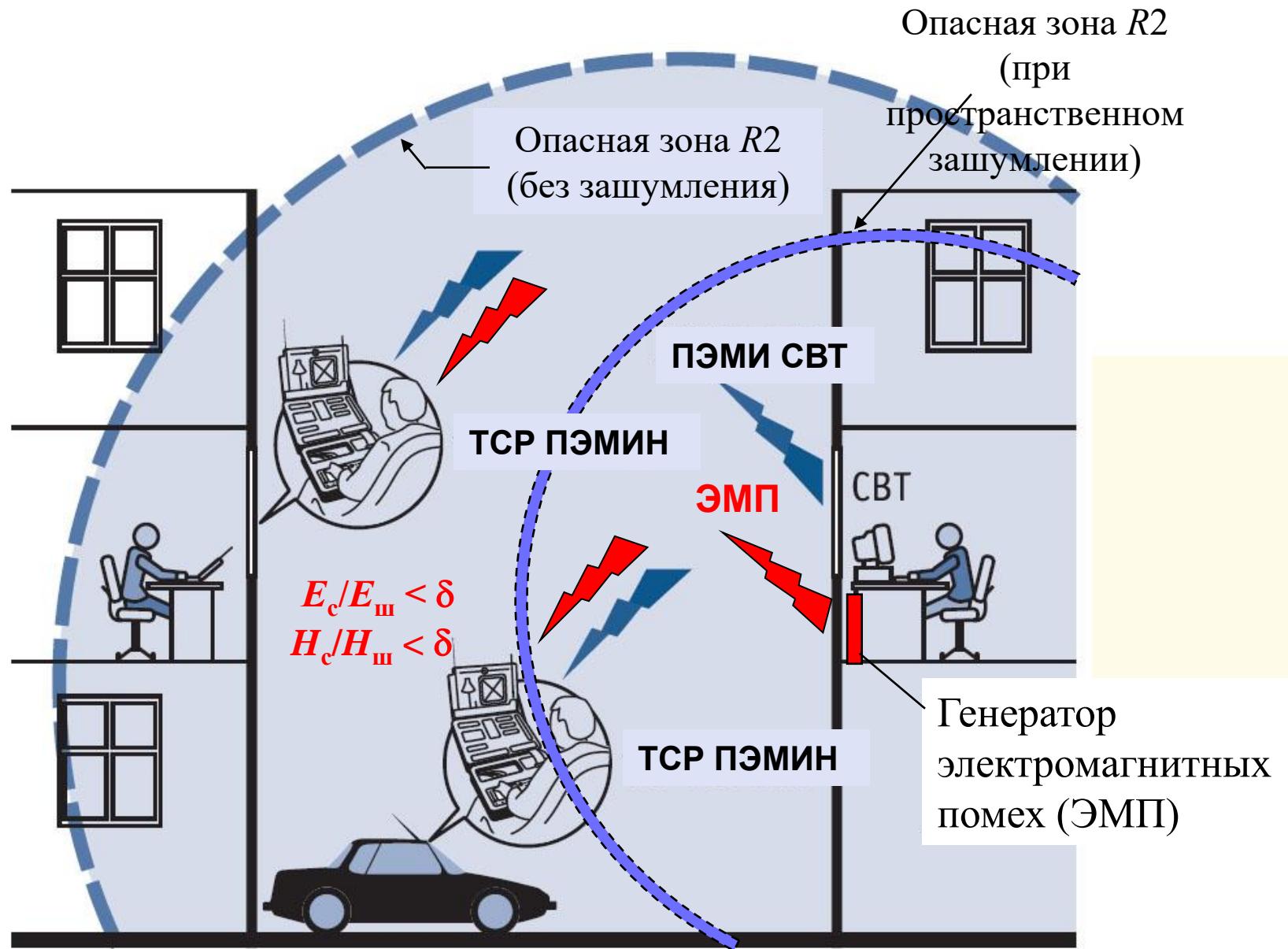
Защита объектов ТСПИ от утечки информации, возникающей **за счет наводок побочных электромагнитных излучений**, достигается: установкой помехоподавляющих фильтров в цепях электропитания ТСПИ, диэлектрических вставок в инженерные коммуникации и экраны кабелей электропитания, а также использованием систем линейного электромагнитного зашумления.

Генератор шума «Гном – 3»



Генератор шума «Гном – 3М»

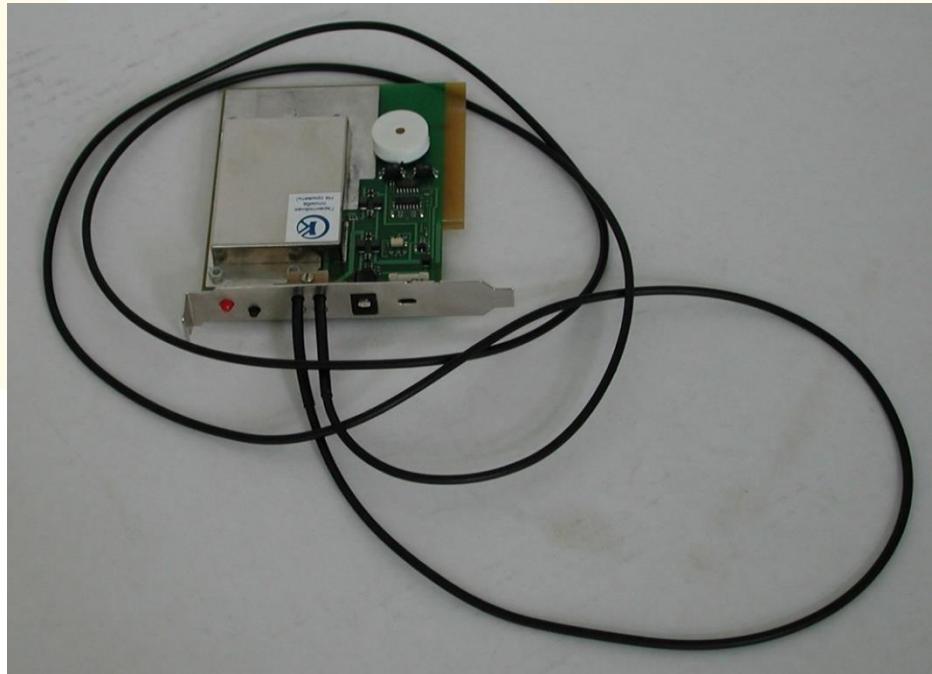






Дополнительная
антенна

Устройство защиты объектов информатизации от утечки информации за счет ПЭМИН "Соната- Р2"



Генератор шума ГШ – К – 1000М



Генератор шума SEL SP - 21

Генератор шума SEL SP - 113

Особенности мероприятий по защите информации:

- Регулярное наблюдение за АС с непредсказуемым графиком;
- Обучение пользователей и администраторов правилам безопасности;
- Смена паролей через определенные промежутки времени.

№п/п	Тематическая группа	Частота выбора пароля человеком, %	Раскрываемость пароля, %
1	Номера документов (паспорт, пропуск, удостоверение личности, зачетная книжка, страховой полис и т.п.)	3,5	90
2	Последовательность клавиш ПК, повторяющиеся символы	14,1	72,3
3	Номера телефонов	3,5	66,6
4	Адрес места жительства (или часть адреса – индекс, город, улица и пр.), место рождения	4,7	55,0
5	Имена, фамилии и производные от них	22,2	54,5
6	Дата рождения или знак Зодиака пользователя либо его родственников (возможно, в сочетании с именем, фамилией или производными от них)	11,8	54,5
7	Интересы (спорт, музыка, хобби)	9,5	29,2
8	Прочее	30,7	5,7 ⁵³

Литература

1. Л.Л. Попов, Ю.И. Мигачев, С.В. Тихомиров «Информационное право», учебник, 2010 г.
2. В.Л. Цирлов «Основы информационной безопасности», учебное пособие для вузов, М.: Феникс, 2018 г.
3. Зайцев А.П., Шелупанов А.А. Технические средства и методы защиты информации. Учебник для вузов. – М.: ООО «Изд-во Машиностроение», 2019 – 508 с.
4. Хорев А.А. Техническая защита информации. Учебное пособие. М.: «Аналитика», 2018.
5. Защита информации Кн. 2 Обеспечение информационной безопасности в экономической и телекоммуникационной сферах Коллектив. моногр. В. М. Алдошин, Р. Н. Акиншин, И. Р. Ашурбейли и др.; Под ред. Е. М. Сухарева. - М.: Радиотехника, 2013. - 211 с.
6. Модели технических разведок и угроз безопасности информации Текст коллект. моногр. Р. Н. Акиншин, А. В. Анищенко, И. Р. Ашурбейли и др.; под ред. Е. М. Сухарева. - М.: Радиотехника, 2013. - 142 с.
7. Бузов, Г. А. Защита информации ограниченного доступа от утечки по техническим каналам. - М.: Горячая линия - Телеком, 2015. - 585 с.
8. Торокин, А. А. Инженерно-техническая защита информации Учеб. пособие для вузов. - М.: Гелиос АРВ, 2015. – 458 с.
9. Хорев, А. А. Программно-аппаратная защита информации Текст учеб. пособие для вузов по направлению 10.03.01 "Информ. безопасность" 2-е изд., испр. и доп. - М.: Форум : ИНФРА-М, 2017. - 351 с.
10. Шаньгин, В. Ф. Защита информации в компьютерных системах и сетях Текст В. Ф. Шаньгин. - М.: ДМК ПРЕСС, 2012. - 592 с. ил.



НЕ БОЛТАЙ ! СТРОГО ХРАНИ ВОЕННУЮ И ГОСУДАРСТВЕННУЮ ТАЙНУ



БОЛТАТЬ — ВРАГУ ПОМОГАТЬ!



СТРОГО ХРАНИ ГОСУДАРСТВЕННУЮ И ВОЕННУЮ ТАЙНУ !

СПАСИБО ЗА ВНИМАНИЕ!

*Медведев Николай Викторович,
к.т.н., доцент каф.ИУ8*

(499) 263 69 36
medvedevnick54@yandex.ru



Московский государственный технический
университет им. Н.Э. Баумана

Цикл лекций по дисциплине
«Методология программной инженерии»

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук

rtn@bmstu.ru

Москва - 2024



Лекция 7

*Верификация, тестирование и оценивание
корректности программных компонентов.*

Принципы верификации и тестирования программ.



Верификация – это процесс для определения, выполняют ли программные средства и их компоненты требования, наложенные на них в последовательных этапах ЖЦ ПС.

Анализ, просмотры (обзоры) и тестирование от требований являются важнейшей частью верификации и установления корректности программ.

Основная цель верификации ПС состоит в том, чтобы обнаружить, зарегистрировать и устраниить дефекты и ошибки, которые внесены во время последовательной разработки или модификации программ.

- Для эффективности затрат ресурсов при ее реализации, верификация должна быть интегрирована как можно раньше с процессами проектирования, разработки и сопровождения.
- Обычно она проводится сверху вниз, начиная от общих требований, заданных в ТЗ и/или спецификации на всю ИС до детальных требований на программные модули и их взаимодействие.

**Верификация соответствия спецификации требований
к программному средству требованиям к информационной системе**



**Верификация соответствия архитектуры комплекса программ
требованиям к программному средству**



**Верификация соответствия спецификаций требований
к функциональным компонентам требованиям
к программному средству**



**Верификация соответствия спецификаций требований
к программным и информационным модулям требованиям
к функциональным компонентам**



**Верификация соответствия объектного кода программ
спецификациям требований к модулям**

**Верификация соответствия спецификации требований
к технологическому обеспечению жизненного цикла программ
требованиям к программному средству**

**Верификация соответствия спецификации требований
к эксплуатационной и технологической документации требованиям
к программному средству**

Информация о процессе верификации включает:

- ❖ требования к системе,
- ❖ требования к ПС и к его архитектуре,
- ❖ данные о прослеживаемости последовательного преобразования требований,
- ❖ исходный текст программ,
- ❖ исполняемый объектный код,
- ❖ план верификации ПС,
- ❖ план квалификационного тестирования ПС.

***Документы верификации должны
содержать следующее:***

- выполненные процедуры верификации ПС,
- описание и отчет о квалификационном тестировании ПС и его компонентов.



Просмотры и анализы требований высокого уровня предназначены для того, чтобы обнаруживать, регистрировать и устранять дефекты и ошибки, которые внесены в процессе последовательной разработки и детализации спецификаций требований к ПС.

Эти просмотры и анализы должны подтвердить корректность и согласованность требований высокого уровня.



Эти просмотры должны гарантировать что:

- ✓ полностью определены функции ИС, которые она должна выполнять;
- ✓ требования по функциональности, эффективности и к качеству системы детализированы в исходных требованиях высокого уровня к ПС;
- ✓ правильно определены производные требования и обоснована их необходимость;
- ✓ каждое требование высокого уровня к ПС является точным, однозначным и достаточно детализированным;
- ✓ требования не конфликтуют друг с другом;

- не существует никаких конфликтов между требованиями высокого уровня и возможностями аппаратных и программных средств такими, как реактивность системы и характеристики аппаратуры ввода/вывода;
- процесс разработки требований к ПС полностью соответствует стандартам на создание спецификаций требований и любые отклонения от стандартов обоснованы;
- функциональные и конструктивные характеристики качества полностью включены в требования высокого уровня.

Просмотры и анализы исходных текстов программ предназначены для выявления и регистрации дефектов и ошибок, которые внесены в процессе программирования компонентов ПС.

Эти процедуры должны подтверждать :

- тексты программ, являются точными, полными и могут быть проверены;
- должна определяться корректность текста программ по отношению к исходным требованиям и к архитектуре ПС;
- коды программ соответствуют стандартам на программирование.

Тестирование ПС от требований имеет две взаимодополняющие цели:

1. Показать, что ПС удовлетворяет заданным требованиям к нему.
2. Показать с высокой степенью доверия, что устраниены дефекты и ошибки, которые могли бы привести к возникновению отказов, влияющих на корректность и надежность системы.

Цель интеграционного тестирования гарантировать, что программные компоненты взаимодействуют друг с другом корректно и удовлетворяют требованиям к ПС и к его архитектуре.



Только совместное и систематическое применение различных методов тестирования позволяет достигать высокое качество функционирования сложных комплексов программ!!!

При выборе методов тестирования необходимо учитывать общие особенности:

- относительно высокая доля творческого труда тестировщиков,
- непредсказуемость видов и мест выявляемых ошибок в программах;
- разнообразие возможных мест расположения и видов ошибок, при относительно редком их обнаружении;



- *высокая сложность отлаживаемых ПК*, затрудняют и ограничивают возможную точность оценки полноты проведенного тестирования и достигнутого качества компонентов;
- активное участие в тестировании специалистов, различающихся по квалификации, опыту, темпераменту и творческим возможностям;
- различия архитектуры, сложности ПС и их функций не позволяют регламентировать трудоемкость и выбор технологий и средств автоматизации тестирования.

При тестировании выделяют
следующие виды тестирования:

1. **Тестирование для обнаружения ошибок**, то есть выявление отклонений результатов функционирования реальной программы от заданных требований и эталонных значений.
2. **Тестирования для диагностики и локализации ошибок** предназначено для выявления причин и источников отклонения результатов от эталонов.

Методы и стратегии тестирования ПК используются для обработки текстов программ в различной форме и для представления информации о результатах тестирования в виде удобном для анализа специалистами.

Программы в процессе тестирования
используются в двух различных
формах представления:

1. текст программы **на языке программирования** или формализованного описания спецификаций требований (графическое представление);
2. Текст в **машинном коде** конкретной ЭВМ (объектное представление), пригодном для автоматической обработки определенных кодовых исходных данных и неудобном для их анализа человеком.

1. Тестирование потоков управления

(структуры программы) должно быть начальным этапом, так как при некорректной структуре возможны наиболее грубые искажения выходных результатов.

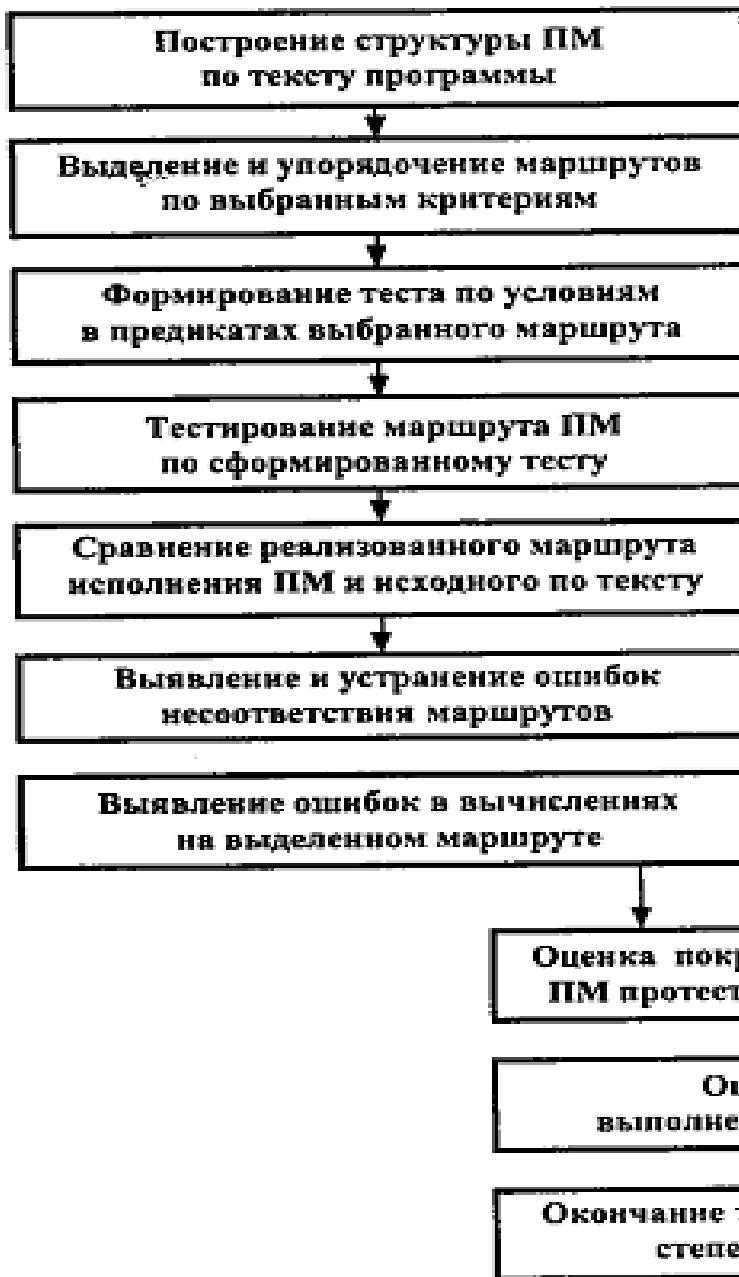
Строится управляющий граф программы (УПГ) и по нему составляются тесты для проверки всех путей передачи управления в программе.

1. Проводится анализ обработки данных, определяющих значения логических предикатов.
 2. Производится проверка вычислений по аналитическим формулам.
- В качестве эталонов используются результаты ручных или автоматизированных расчетов по тем же или близким по содержанию формулам.

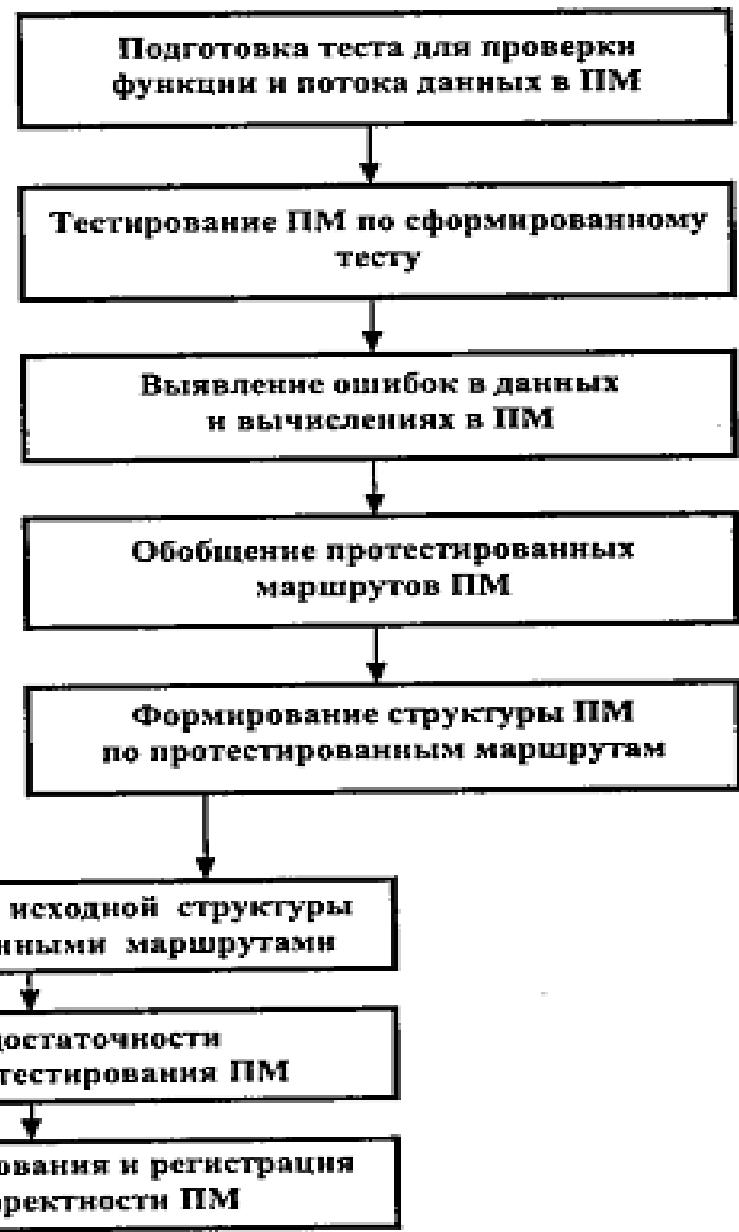
Особенности описаний и реализации
программ, а также *мышления*
программистов – на основе функций и
процедур исполнения программ, существенно
отличаются от представлений и методов
описаний тех же функций программ
независимыми тестировщиками –
создателями сценариев тестирования!!!



Стратегия 1



Стратегия 2



При *первой стратегии* за основу принимается структура ПМ, построенная по тексту программы в виде графа (УГП).

1. В УПГ по некоторым критериям выделяются и упорядочиваются маршруты исполнения программы и условия-предикаты, при которых они могут быть реализованы.
2. Эти условия используются для подготовки тестовых наборов, каждый из которых должен реализоваться по маршруту, принятому за эталон при подготовке теста.
3. Отклонение исполнения теста от первоначально выбранного маршрута рассматривается как ошибка.

При *второй стратегии* за основу принимаются требования спецификаций, конкретные тестовые и эталонные значения, которые подготавливаются специалистами путем анализа переменных и предикатов в тексте программы.

При каждом teste программа исполняется по определенному маршруту, который регистрируется. При этом реализованный, в соответствии с анализируемыми требованиями, маршрут и поток данных рассматриваются как эталонные компоненты структуры программы.

По мере тестирования отмечаются проверенные операторы, и оценивается полнота *покрытия тестами требований спецификаций* на маршрутах тестирования.

Проблема: когда завершить верификацию и тестирование?

- На основе экспериментальных данных *созданы математические модели*, которые позволяют прогнозировать интервалы времени между последовательными обнаружениями дефектов или ошибок в ПК при некотором методе и этапе верификации или тестирования.
- Например, если при тестировании определенного модуля или компонента ПС выявлено определенное число дефектов (например, 5 или 10), то модели позволяют оценить ресурсы (время), необходимое для обнаружения следующего дефекта и рентабельность продолжения тестирования используемым методом.

Для отладки сложных ПС и пригодных для повторного использования в различных проектах, необходим комплекс средств автоматизации, использующий основные современные методы выявления ошибок и дефектов в программах.

Системы автоматизации тестирования
можно разделить на два вида:

1. *статические* системы, которые анализируют спецификации и исходные тексты программ в объектном коде без их исполнения на ЭВМ (анализаторы);
2. *Динамические системы*, при использовании которых программы функционируют в объектном коде и пригодны для их реального применения на ЭВМ.

В группу средств *статической отладки* входят также *средства расчета длительностей исполнения модулей и компонентов.*

Эти средства позволяют получать ориентировочные значения и распределения длительностей счета программы аналитически, без ее исполнения на ЭВМ.

В результате расчетов выявляются компоненты программы, требующие избыточно большого времени счета на реализующей ЭВМ, а также подготавливаются данные для общей *проверки реализуемости ПС в реальном времени.*



Группу средств *динамической отладки*
можно разделить на два типа:

1. Основные средства, непосредственно обеспечивающие исполнение программ в соответствии с отладочными заданиями.

В основные входят средства:

- трансляции программ, тестов и заданий с языка отладки;
- исполнения программы по отладочному заданию в соответствии с планом и сценарием тестирования;
- регистрации данных о результатах тестирования.

2. Вспомогательные средства, которые анализируют выполненное тестирование, его результаты и проведенные корректировки.

Отображение результатов тестирования в мнемонической и графической формах, может обеспечиваться унифицированными средствами интерактивного взаимодействия пользователей с ЭВМ.

Характеристики сложности тестирования областей в совокупности с характеристиками сложности тестирования структуры программы и циклов позволяют оценить реализуемость плана тестирования конкретного программного модуля или компонента. (Учебное пособие «Тестирование программного обеспечения» Романова Т.Н.)





**Федеральное государственное бюджетное образовательное учреждение высшего
образования «Московский государственный технический университет
имени Н. Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)**

Цикл лекций по дисциплине **«Методология программной инженерии»**

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук.

rtn@bmstu.ru

Россия, Москва - 2024



Лекция 8

Сопровождение и мониторинг ПС.

Организация и методы сопровождения ПС.

*Задачи и процессы переноса программ
и данных на иные платформы.*

При организации сопровождения крупных ПС следует учитывать важные психологические факторы:

1. Эта деятельность требует очень высокой квалификации и больших умственных затрат, связанных с необходимостью одновременного широкого охвата и анализа множества компонентов ПС и их взаимосвязей, находящихся в различных состояниях завершенности модификаций.
2. Корректируемые компоненты зачастую разрабатывались в прошлом в разное время, различными специалистами, в различном стиле и с неодинаковой полнотой документирования, что усложняет освоение их содержания при внесении изменений и устранении дефектов;

3. Приходится анализировать программы, разработанные ранее другими специалистами, которые проще разработать заново.
4. Комплексы программ, прошедшие широкие испытания и эксплуатацию у заказчиков гарантируют достигнутое качество результатов функционирования, и любые в них изменения имеют высокий риск внесения дополнительных ошибок и ухудшения этого качества, что ограничивает возможность коренных модификаций.

5. Выполняемые работы требуют особой, скоординированной тщательности корректировок и четкого регламентированного взаимодействия ряда специалистов, различающихся квалификацией и уровнем ответственности.

6. Процессы и результаты сопровождения не отличаются наглядностью и внешним эффектом, проявлением их размера и сложности, вследствие чего не престижны среди рядовых программистов и недооцениваются руководителями проектов.

- Опыт последних лет показал, что во многих случаях для сопровождения и мониторинга версий необходимо практически такое же, или даже большее, число специалистов, чем разработало первую версию ПС.
- Это приводит к тому, что по мере накопления эксплуатируемых ПС и их компонентов все большее число специалистов переходит из области непосредственного программирования новых программ в область системного проектирования и создания новых версий ПС *на базе повторно используемых компонентов.*

По некоторым оценкам
непосредственным программированием
новых компонентов в мире занято около
15 – 20% специалистов, участвующих
в создании программных продуктов.

Целью сопровождения является:

1. Выявление и устранение обнаруженных дефектов и ошибок в программах и данных.
2. Введение новых функций и компонентов в программный продукт.
3. Анализ состояния и корректировка документации.
4. Тиражирование и контроль распространения версий ПС.
5. Актуализация и обеспечение сохранности документации и физических носителей.

Основная задача - изменить и улучшить существующий программный продукт, сохраняя его целостность и функциональную пригодность.

* Для сохранения и повышения качества сложных комплексов программ *необходимо регламентировать процессы модификации и совершенствования ПС*, а также поддерживать их соответствующим тестированием и контролем качества.

Сопровождаемость – возможность регламентированной модификации, отражающей возможность и простоту внесения изменений в программный продукт после его ввода в эксплуатацию.

Сопровождаемость должна быть определена до начала первичной разработки проекта ПС соответствующим *соглашением между заказчиком и разработчиком-поставщиком*.

*Разработчик должен подготовить план обеспечения сопровождения, в котором отражены конкретные методы и соответствующие ресурсы.

Требования к процессам сопровождения определяются группой основных факторов:

- *требования на изменения,*
- *изменяемые функции,*
- *размер (масштаб) изменений,*
- *стратегия модификаций,*
- *ресурсы для их реализации.*

- Основным критерием оценки сопровождения является *совершенствование функциональной пригодности* и улучшение характеристик качества ПС.
- Процесс сопровождения ПС проводится в соответствии со стандартом ISO 12207 и детализацией этого раздела в стандарте ISO 14764.
- Стоимость процесса сопровождения может составлять наибольшую часть стоимости ЖЦ ПС.
- Период значительного изменения размера, функций и характеристик качества в крупных проектах комплексов программ составляет обычно 1-2 года.

Спецификация требований на изменения ПС должна исчерпывающе и однозначно описывать требования к ПС ,к его модификациям и отражать характеристики качества.

Факторы, влияющие на Сопровождаемость:

- определение и описание новых функций;
- точность и логическая организация данных;
- интерфейсы (системные, компонентов и пользователей), особенно новые и перспективные интерфейсы;

- требования к функциям и рабочим характеристикам, включая влияния корректировок и дополнений;
- требования, налагаемые внешней средой;
- план обеспечения качества модифицированного ПС, в котором особое внимание должно быть уделено документам на изменения и их согласованность.

Цели и состав процессов сопровождения программных средств

Причины и виды изменения программных средств при сопровождении

**Организация процессов и передача на сопровождение
программного средства**

**Заключение договора между заказчиком и исполнителем
на сопровождение программного средства**

**Разработка концепции методов и процессов сопровождения
программного продукта**

**Разработка спецификаций требований на модификации
при сопровождении программного средства**

**Оценка и распределение ресурсов, финансовых и специалистов
для сопровождения программного продукта**

**Разработка требований к обеспечению качества при сопровождении
программного продукта**

**Утверждение заказчиком концепции, договора и технического задания
на сопровождение программного продукта**

**Организация контроля реализации концепции
и договора на сопровождение программного продукта**

*Требования к функциональным характеристикам и качеству, *утвержденные после проектирования концепции*, могут быть закреплены в ТЗ как обязательные для детального и рабочего проектирования модификаций.

*Эти данные могут использоваться при последующего оценивания качества в процессе квалификационных испытаний, сертификации модификаций или новой базовой версии программного продукта.

Этапы и процедуры при сопровождении ПС

В соответствии с требованиями стандарта ISO 12207 (ISO 14764) по развитию и модификации программного продукта в ЖЦ **должен быть организован процесс его сопровождения.**

Работы по сопровождению ПС:

- * подготовка процесса;
- * анализ проблем и изменений;
- * внесение изменений;
- * проверка и приемка при сопровождении;
- * перенос на иные платформы;
- * снятие с эксплуатации.

Общий план сопровождения должен определять:

- причины необходимости сопровождения;
- состав исполнителей работ по сопровождению;
- роли и обязанности каждого субъекта, вовлеченного в сопровождение;
- как должны быть выполнены основные процессы;
- какие имеются и какие необходимы ресурсы для сопровождения;
- методы и средства организации работ по управлению, выпуску продукта и синхронизации работ;

- перечень всех проектных результатов и продуктов, подлежащих поставке заказчику;
- критерии завершения соответствующей деятельности, работ и задач;
- состав отчетных материалов по этапам, затратам и графикам проведения работ;
- периодичность и способы выдачи отчетных материалов;
- состав отчетных материалов по проблемам и устранным дефектам;
- время начала и длительность сопровождения.

- * Пользователь представил отчеты о дефектах.
- * Сопроводитель должен *продублировать и верифицировать* возникшие проблемы, выполнив следующие этапы решения данной задачи:
 - 1) разработать стратегию верификации и квалификационного тестирования для проверки и устранения конкретной проблемы – дефекта;
 - 2) провести тестирование для проверки наличия проблемы;
 - 3) документально оформить результаты тестирования.- 4) Если конкретная проблема не может быть повторена сопроводителем, он должен проверить правила, политики и документы обеспечения ЖЦ ПС на предприятии.*

На основе проведенного анализа
Сопроводитель должен разработать ***варианты реализации изменения:***

- * присвоить соответствующий приоритет проблеме (дефекту) или предложению о модификации;
- * установить наличие возможностей и средств для решения проблемы;
- * оценить масштаб и трудоемкость данной модификации;
- * разработать варианты реализации конкретного изменения;
- * определить влияние данных вариантов на функциональную пригодность и технические средства системы;
- * выполнить анализы риска для каждого варианта модификации.

Всё задокументировать должен Сопроводитель!!!

Задачи и процессы переноса программ и данных на иные платформы

- ❖ Многочисленное дублирование, по существу, одних и тех же программных средств и информации баз данных на подобных или разных платформах сопряжено со значительными нерациональными затратами на их разработку и с увеличением длительностей создания информационных систем.
- ❖ Для их сокращения необходима организация, технология и инструментарий, обеспечивающие *эффективный перенос* готовых программ и данных в пределах одной операционной и аппаратной среды или с иных платформ.
- ❖ Для этого создаются методологии и технологии переноса, а также стандарты, поддерживающие процессы и разработку переносимых программ и данных.
- ❖ В каждом конкретном случае *необходима оценка рентабельности переноса* по сравнению с полной разработкой аналогичного программного продукта на новой платформе.

Под мобильностью – переносимостью понимаются:

- 1) процессы переноса программ и данных из одной аппаратной, операционной и пользовательской среды в иную по архитектуре и характеристикам среду с сохранением их целостности или небольшими изменениями функций системы;
- 2) процессы повторного использования готовых программных компонентов и средств, а также информации баз данных, возможно, в пределах одной архитектуры аппаратной и операционной среды *для расширения и изменения функций системы и программного продукта.*

Задачи повторного использования и переноса программ и данных:

1. Встраивание готового ПС и информации базы данных в создаваемую новую систему при условии, что их поставщики гарантируют функционирование на выбранной платформе;
2. Перенос программ и данных на новую платформу;
3. Обеспечение доступа к информационным ресурсам других распределенных систем и сетей.

При переносе свойства ПС практически всегда изменяются. Это следует учитывать при анализе целесообразности переноса, поскольку необходимы их отладка, испытания и сертификация в новой среде.

Выделяются следующие уровни переноса и повторного использования ПС:

- модели предметной области и спецификации требований, возможно, реализуемые разными способами на этапе проектирования ПС;
- проектные спецификации требований на этапе разработки;
- исходные тексты программ на языках программирования, применявшихся при разработке повторно используемых программных компонентов;

- объектные коды программ, когда обеспечена структурная и аппаратная совместимость между исходной и целевой платформами;
- структуры файлов и информация баз данных;
- тесты проверки функционирования компонентов;
- тесты проверки соответствия повторно используемых программ стандартизованным интерфейсам.

В зависимости от степени программной совместимости, между исходной и новой (целевой) платформами, можно рассматривать следующие варианты применения мобильности:

- при полной несовместимости платформ может потребоваться разработка всего комплекса программ заново (возможно, с использованием имеющихся спецификаций);
- при несовместимости языков программирования или диалектов одного языка требуется переписывание программ ПС на том языке, который принят для проекта новой системы (возможно, с использованием имеющихся проектных спецификаций и встраиваемых повторно используемых компонентов);

- при несовместимости аппаратно-программных платформ, поддерживающих один и тот же язык программирования, требуется перекомпиляция текстов ПС на новой платформе;
- при обеспечении двоичной совместимости архитектуры исходной и новой платформ перенос достигается непосредственным исполнением ПС на новой платформе (возможно, использующей средства эмуляции некоторых компонентов архитектуры исходной платформы).

Процессы переноса программных средств и баз данных регламентируются рядом процедур и документов, стандартизованных в ISO 14764.

Специалисты, которые проводят перенос по рекомендациям этих стандартов, должны:

- 1) разработать план переноса,
- 2) известить пользователей,
- 3) обучить персонал,
- 4) предупредить о завершении переноса,
- 5) оценить влияние новой версии и внешней среды,
- 6) архивировать соответствующие данные.

Отчетными документами по переносу

ПС и БД являются:

1. Перенесенный программный продукт на новой платформе;
2. План реализации переноса;
3. Инструментальные средства для переноса;
4. Извещения о намерениях по переносу;
5. Уведомление о завершении переноса;
6. Архивные данные процессов и результатов переноса.

- Уникальное, заказное ПС, основная часть ЖЦ которого приходится на разработку, может не планировать затраты на сопровождение.
- Однако многократно модернизированные, и широко тиражируемые ПС требуют больших затрат на сопровождение.
- Вследствие длительного срока сопровождения и эксплуатации (в ряде случаев более 10 лет), а также большого числа версий совокупные затраты на сопровождение в некоторых случаях значительно превышают затраты на первичную разработку.

На основе анализа и оценивания рассчитанных характеристик ресурсов для сопровождения следует выполнять заключительное ***технико-экономическое обоснование необходимости сопровождения конкретного программного продукта и определять:***

- целесообразно ли продолжать работы по сопровождению и мониторингу ПС или следует его прекратить, вследствие недостаточных ресурсов специалистов, времени или большой трудоемкости разработки модификаций;
- проводить ли маркетинговые исследования рентабельности создания очередной версии программного продукта и поставки её на рынок;

- достаточно ли полно и корректно формализованы концепция и требования к модификациям версий программного продукта, на основе которых проводились экспертные оценки и расчеты затрат, или их следует откорректировать и выполнить повторный анализ с уточненными исходными данными;
- есть ли возможность применить готовые повторно используемые компоненты ПС,
- рентабельно ли их применять в конкретной версии программного продукта или весь проект разрабатывать как полностью новый.

Принципы управления распределенной информацией

«Ядром» системы управления распределенными информационными ресурсами являются распределенная база данных (РаБД) и система управления распределенной базой данных (СУРБД).

Распределенная база данных - это совокупность логически взаимосвязанных баз данных, распределенных в компьютерной сети.

Система управления распределенной базой данных - это программная система, которая обеспечивает управление РаБД (распределенной базой данных) и прозрачность ее распределенности для пользователей.

Правила построения РаБД (Дейт)

- 1. Локальная автономность.**
- 2. Никакой конкретный сервис не должен возлагаться на специально выделенный центральный узел.**
- 3. Независимость от местоположения.**
- 4. Независимость от фрагментации.**
- 5. Независимость от тиражирования.**
- 6. Распределенная обработка запросов.**
- 7. Управление распределенными транзакциями**
- 8. Независимость от оборудования.**
- 9. Независимость от ОС.**
- 10. Независимость от сети.**
- 11. Независимость от СУБД**

Технические требования к системе

- Оптимизация.
- Расширение традиционных функций СУБД(параллельный доступ, безопасность) с учетом распределенного характера данных.
- Распространение принципов обработки на метаданные.

Операции доступа к распределенным данным и их модификации должны оптимизироваться с целью их максимально эффективного выполнения.

Возможные варианты:

1. Оптимизация может достигаться за счет удаленного извлечения данных и их последующей пересылки на некоторый промежуточный пункт, где они будут скомбинированы с другими данными и переданы по назначению;
2. Все требуемые блоки данных будут пересылаться непосредственно на запрашивающий узел, где будет произведена их обработка;
3. Возможна и другая альтернатива

Соответствующие решения должны приниматься в каждом конкретном случае с применением некоторого алгоритма предварительной оценки затрат.

Однородные и неоднородные системы

- **Однородные распределенные системы** - имеют в своей основе один продукт СУБД, обычно с единственным языком баз данных (например, SQL с расширениями для управления распределенными данными).

Используется метод построения БД «сверху вниз»

- **Неоднородные распределенные системы** – противоположность однородным. Включают в себя 2 или более продукта управления данными.

Используется метод построения БД «снизу вверх»

Архитектура однородной РаБД

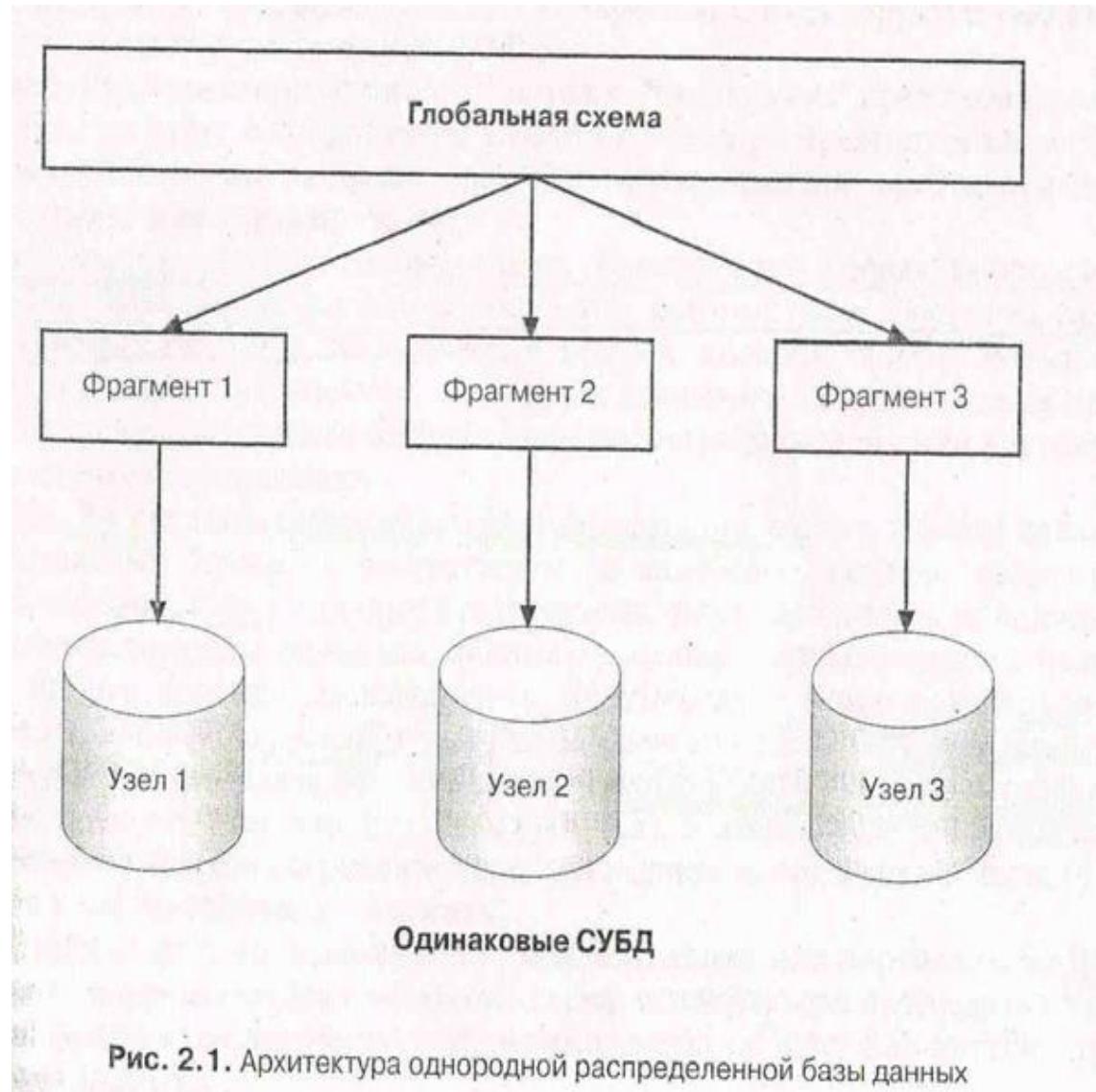


Рис. 2.1. Архитектура однородной распределенной базы данных

Архитектура неоднородной РаБД

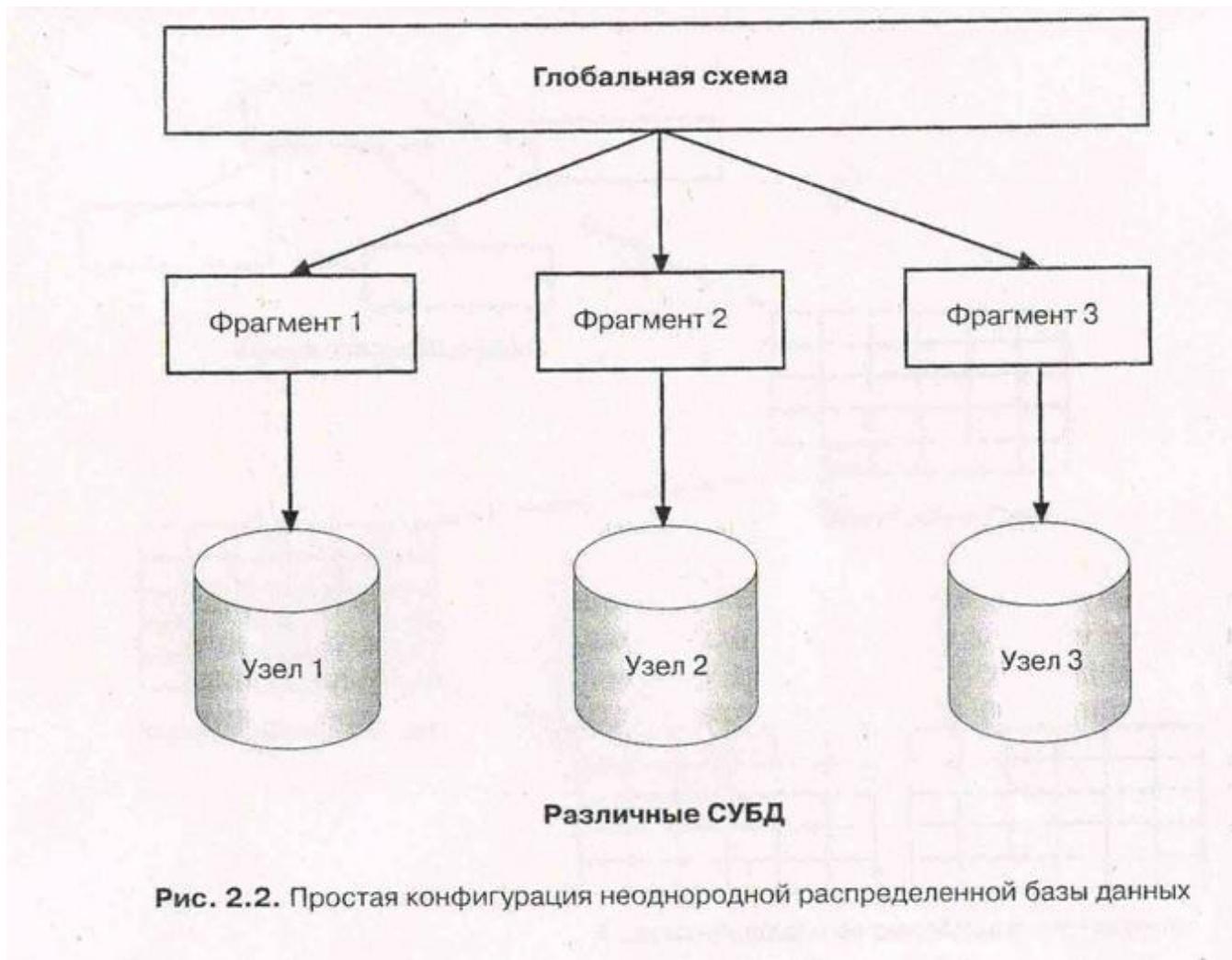


Рис. 2.2. Простая конфигурация неоднородной распределенной базы данных

Модель построения данных «сверху вниз»

Используется для однородных РабД.
Методы распределения данных:

- **Фрагментация** – декомпозиция объектов БД на 2 или более частей. Может быть горизонтальной или вертикальной
- **Тиражирование (или репликация)** означает создание дубликатов данных.

Схема построения «сверху вниз»

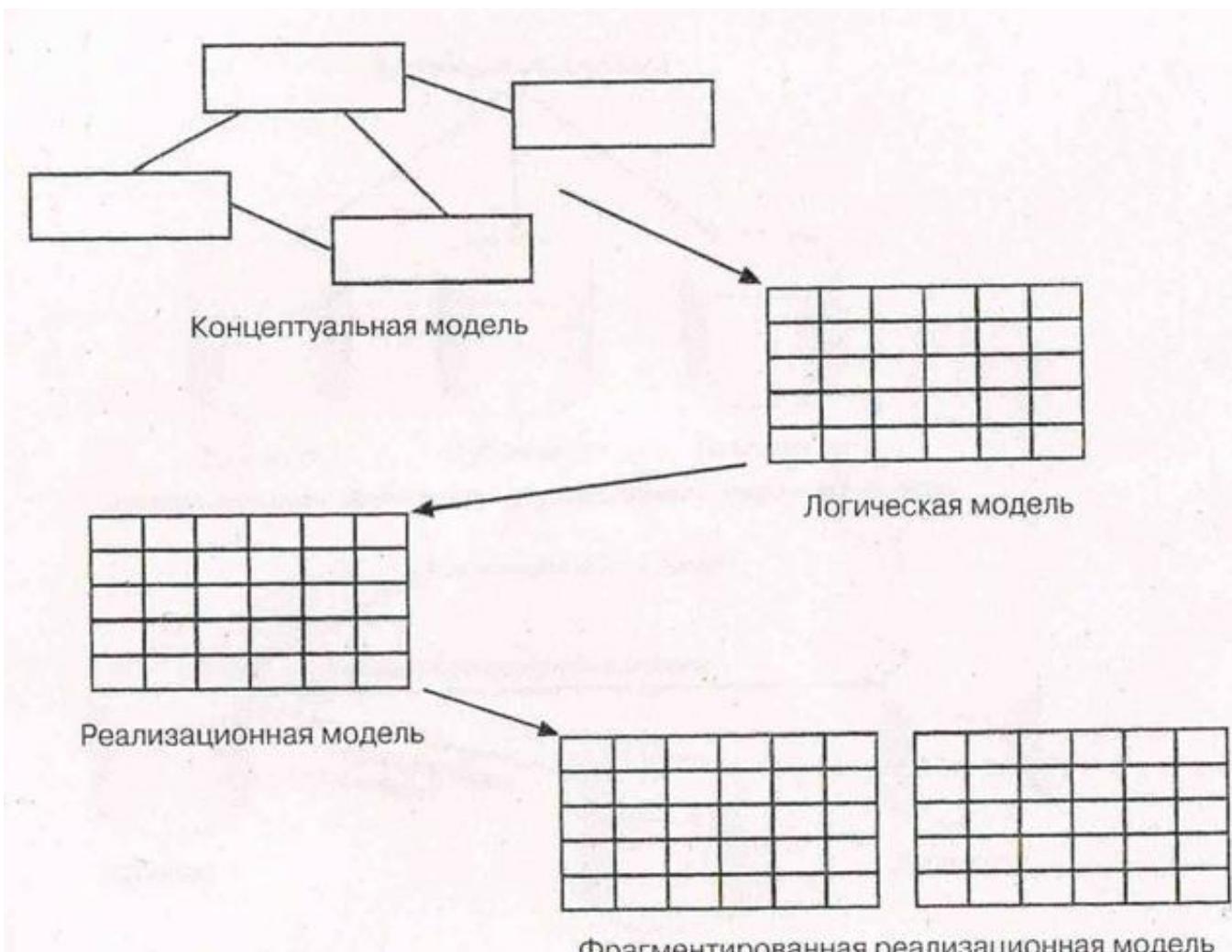


Рис. 2.3. Построение распределенной базы данных методом «сверху вниз»

Фрагментация

- **Горизонтальная** – в таблице делаются горизонтальные срезы в соответствии со значением какого-либо столбца. Также может осуществляться не по значениям, а по принципу «карусели»
- **Вертикальная** – разбиение не по строкам а по столбцам

Горизонтальная фрагментация



Рис. 2.4. Горизонтальная фрагментация

Вертикальная фрагментация

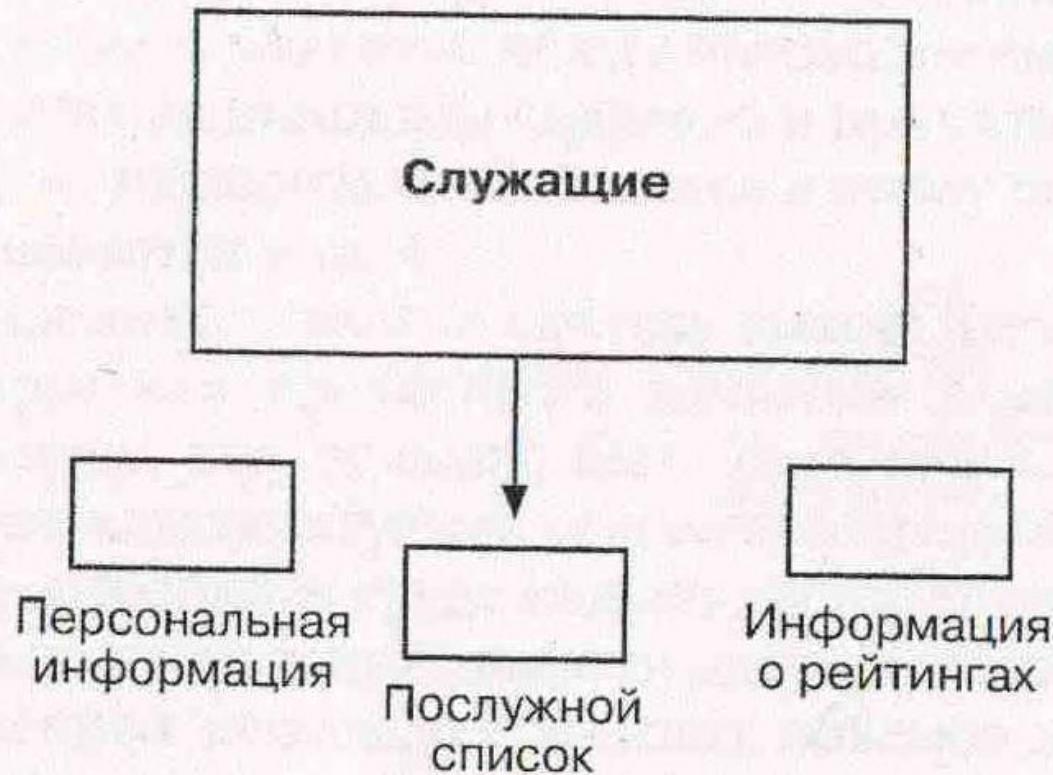


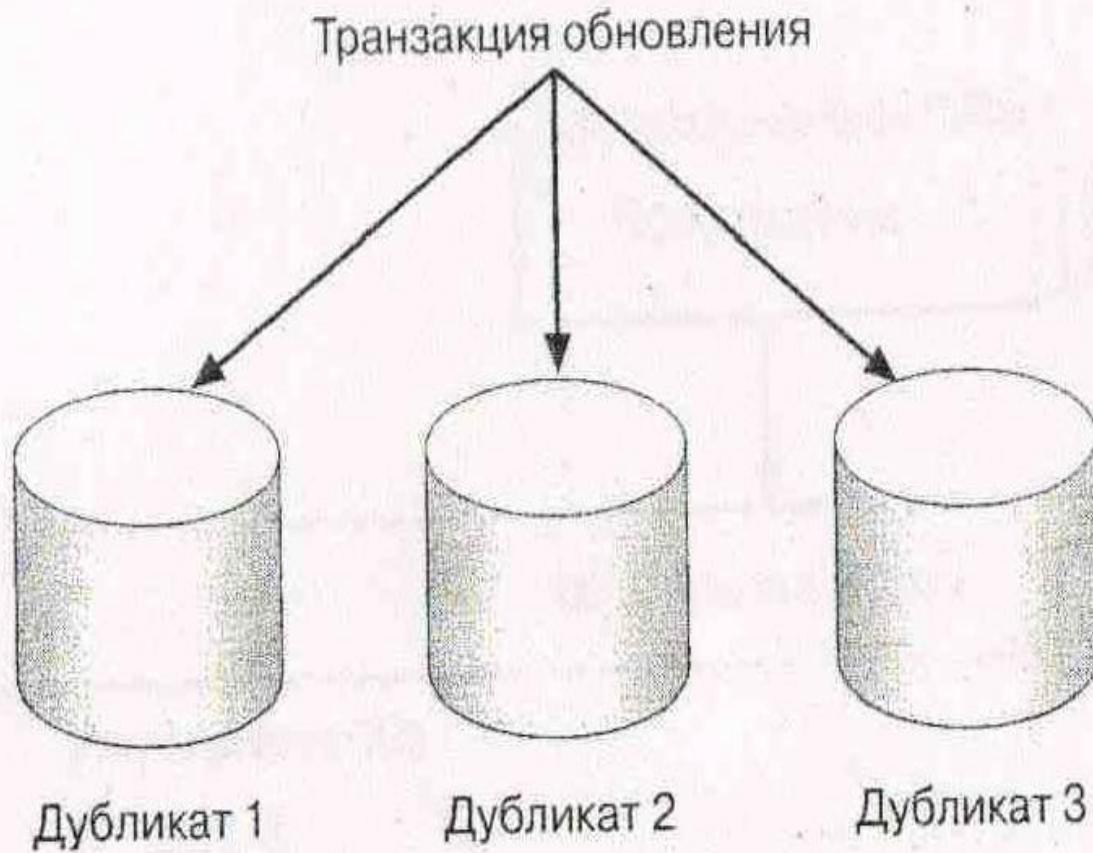
Рис. 2.5. Вертикальная фрагментация

Тиражирование

- **Тиражирование (или репликация)** означает создание дубликатов данных.
- **Репликаты** - это множество различных физических копий некоторого объекта базы данных (обычно таблицы), для которых в соответствии с определенными в базе данных правилами поддерживается синхронизация (идентичность) с некоторой "главной" копией.

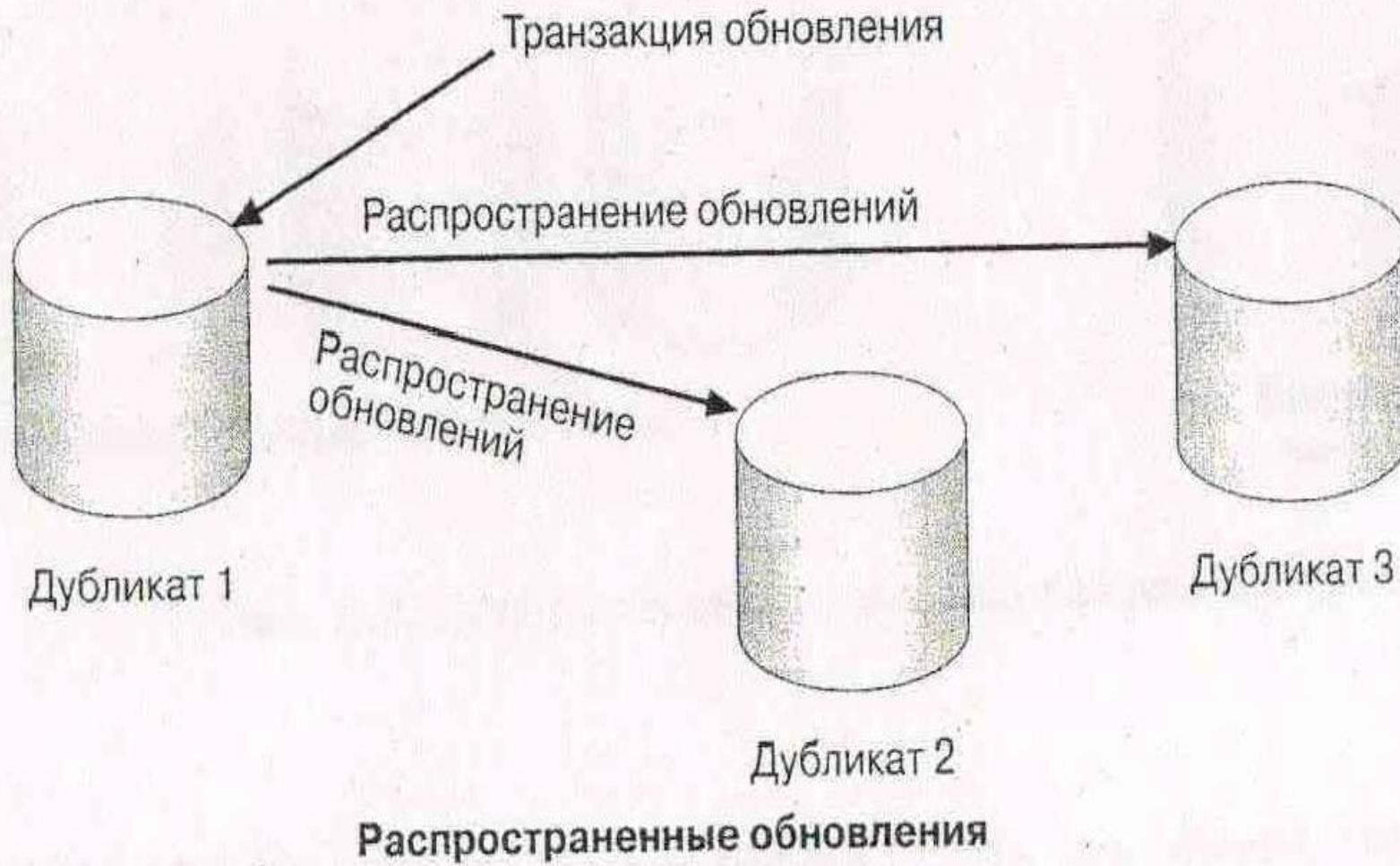
Варианты репликаций.

1. Одновременное обновление

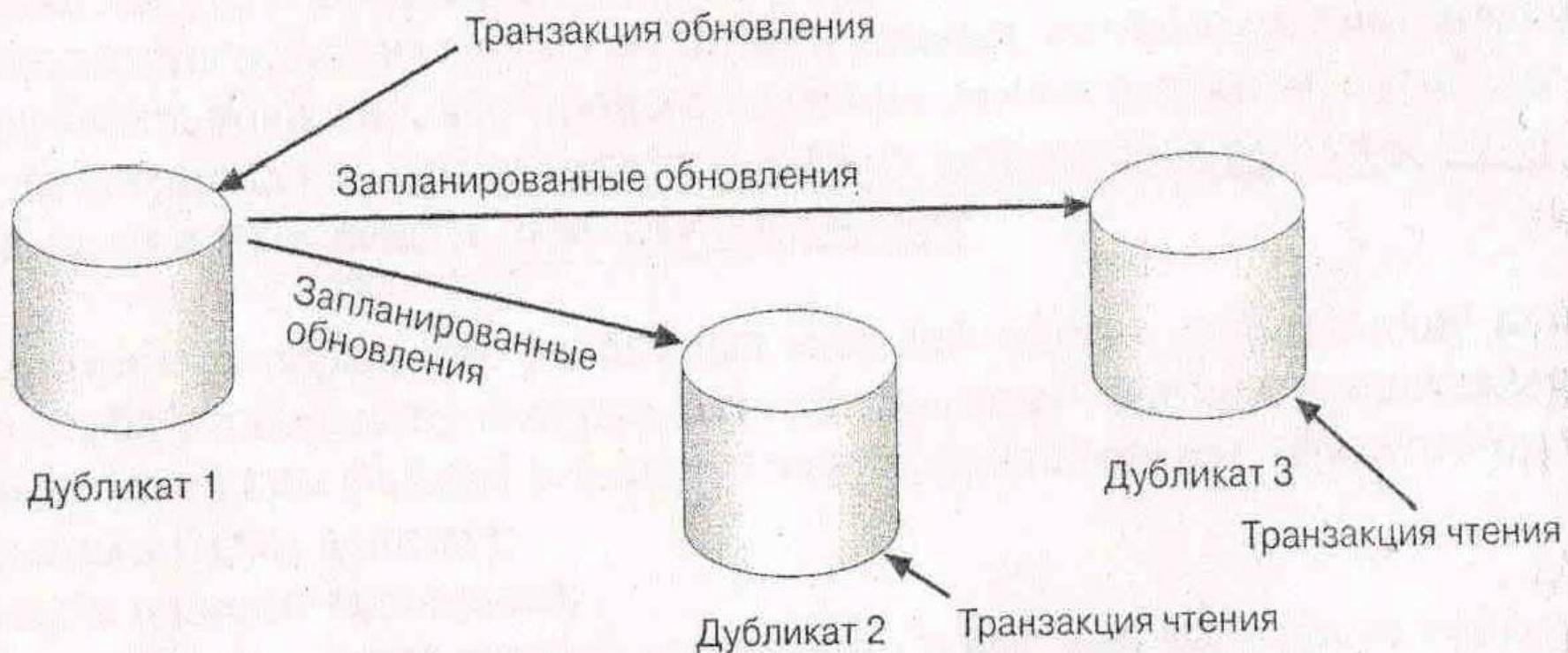


Одновременное обновление (с управлением параллелизмом)

2. Распространенные обновления



3. Запланированная синхронизация



Запланированная синхронизация дубликатов только для чтения

Модель построения данных «снизу вверх»

Подход «сверху вниз» используется для однородных систем.

При решении задач создания интегрированной среды на основе существующих систем используется модель «снизу вверх».

Основная проблема – объединение существующих БД для предоставления приложениям доступа ко всем ресурсам данных.

Схема «снизу вверх»

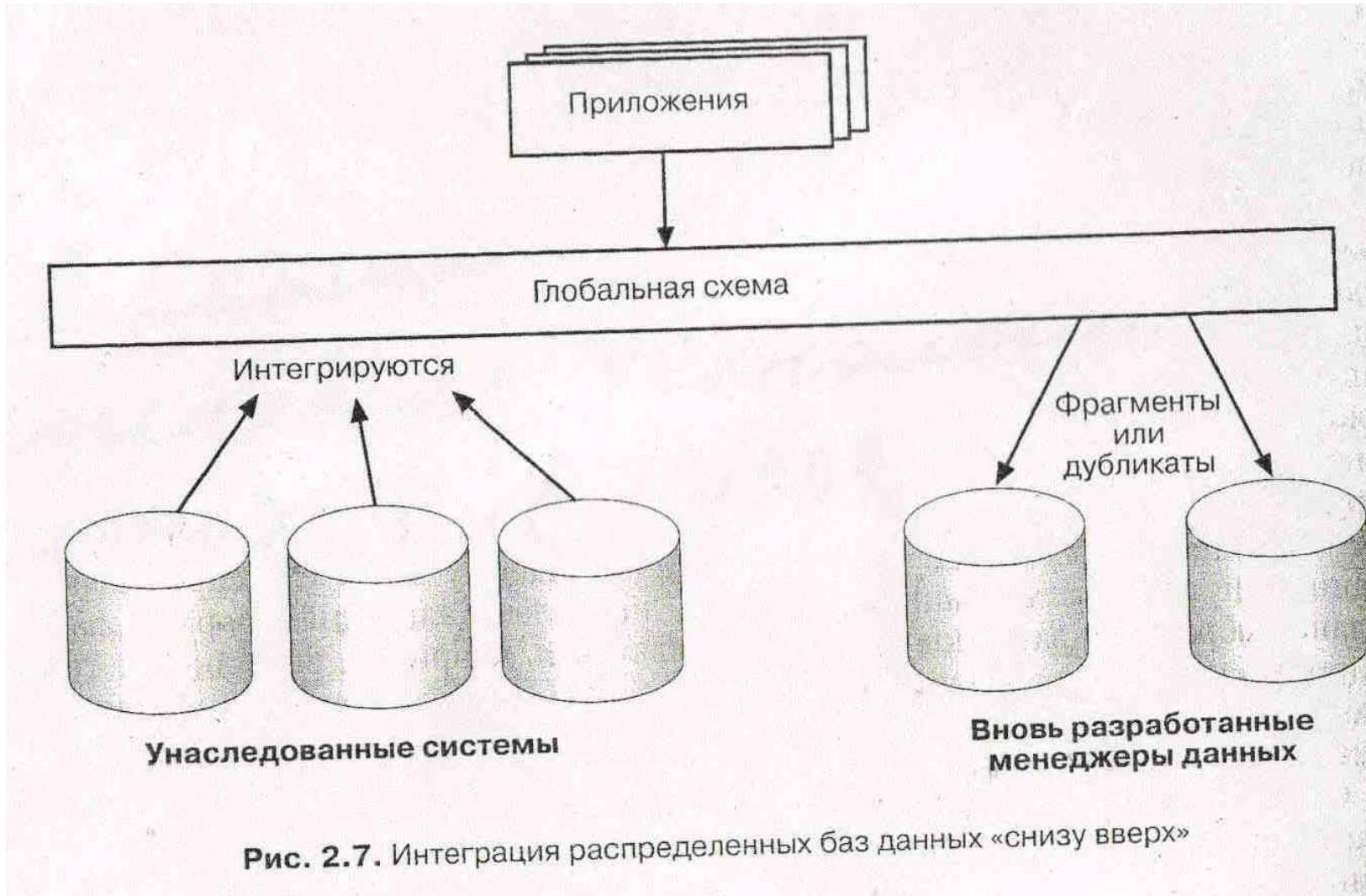


Рис. 2.7. Интеграция распределенных баз данных «снизу вверх»

Технические проблемы подхода

1. *Взаимное отображение различных моделей организации данных*, т.е. наличие некоторого способа глобального доступа к множественным формам представления данных: к плоским файлам, к иерархическим, реляционным, объектно-ориентированным БД).
2. *Управление метаданными;*
3. *Разрешение несоответствий impedance*
(таких, как различные представления некоторого типа объекта данных в разных БД.)
Например, элемент MOVIE_TYPE - тип фильма - в одной БД имеет числовое представление, а в другой - символьное).

Заключение

Краткосрочные перспективы

- Превращение распределенности во всеобщее свойство СУБД и сред управления информацией.
- Рост числа продуктов, для которых распределенность будет внутренним свойством, а не дополнительной возможностью
- Достижение нового понимания старых концепций(таких как тиражирование, фрагментация) и появление новых способов распределения данных.

Заключение

Долгосрочные перспективы

- Решение ряда проблем – масштабируемость, управление глобальной схемой, оптимизация распределенных запросов и тд.
- Создание возможностей, позволяющих значительно более легкими способами, чем в существующих системах, обращаться с высокой степенью неоднородности поддерживающих компонентов.

Достоинства и преимущества case-систем (Computer-Aided Software/System Engineering)

1. Единый графический язык.
2. Единая база данных проекта (репозиторий). Репозиторий может сохранять свыше 100 типов проекта, это описание данных, модели данных, исходные коды, меню.
3. Интеграция средств. Могут интегрироваться разные инструменты – БД, программные пакеты и т.д
4. Поддержка коллективной разработки и управления проектами. Case-технологии поддерживает групповую работу, работу сети, обмен информацией, позволяют осуществлять контроль за работой.

5. Макетирование. Case-система дает возможность быстро построить макеты будущей системы, показать заказчику, оценить и осуществить необходимые изменения.
6. Генерация документации. Автоматически генерируется документация в нужных стандартах, отражающая ход разработки на данный момент.
7. Верификация проекта. Case-технология обеспечивает автономную верификации проекта на ранних стадиях разработки. На этапе проектирования найти ошибку – это в 100 раз дешевле, чем при внедрении.
8. Автоматическая генерация объектного кода. Case-системы делают это автоматически, используя различные языки программирования. (85-90%).
9. Сопровождение и реинжиниринг.

Какими критериями следует руководствоваться при выборе CASE-системы? С первого-то года никакая система не дает никаких положительных эффектов.

1. Выбор зависит от класса решаемых задач. Системы реального времени, коммерческие системы, системы связанные с САПР и т.д. – это все разные классы.
2. Поддержка полного жизненного цикла система.

3. Обеспечение целостности проекта и контроля за его состоянием. Поддерживает ли версионность, верификации.
4. Независимость от программно-аппаратной платформы и выбранных СУБД.
5. Открытая архитектура. Если case-система обеспечена этим, то это значит, можно подключать другие компиляторы, подключение новых СУБД и т.д.
6. Качество технической поддержке в той стране, где идет работа, стоимость приобретения и поддержки новых версий, опыт успешного использования этой case-системы в данной стране.
7. Простота освоения и использования.
Если ее освоить трудно, а квалифицированных кадров немного – то стоит подумать над покупкой более простой.

Критерии классификации CASE-систем

По категориям, то есть по следующим признакам:

- Интегрируемость – оценивается степень интегрированности case-системы с другими средствами разработки: СУБД, средами визуального проектирования и кодирования, ОС.
- Применяемые модели организации данных (реляционные, объектно-ориентированные модели и др.)
- Доступным платформам.

Классификация по типам

1. Upper CASE – это case-системы верхнего уровня, средства анализа предметной области. (Design/IDEF0, BPWin /ERWin).
2. Middle CASE - case-системы среднего уровня. Включают средства анализа и проектирования.
(CASE-Аналитик, Design/2000, Silveran, PRO-IV).
3. Case-системы, которые позволяют спроектировать схемы БД и сгенерировать сами БД.
(ERWin, S-Designer (Oracle)).
4. CASE-системы, включающие средства разработки приложений. Это языки четвертого поколения (4GL)-среды визуального программирования.



Федеральное государственное бюджетное образовательное учреждение высшего
образования «Московский государственный технический университет
имени Н. Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

Цикл лекций по дисциплине

«Методология программной инженерии»

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук

rtn@bmstu.ru

Москва - 2023



Лекция 10

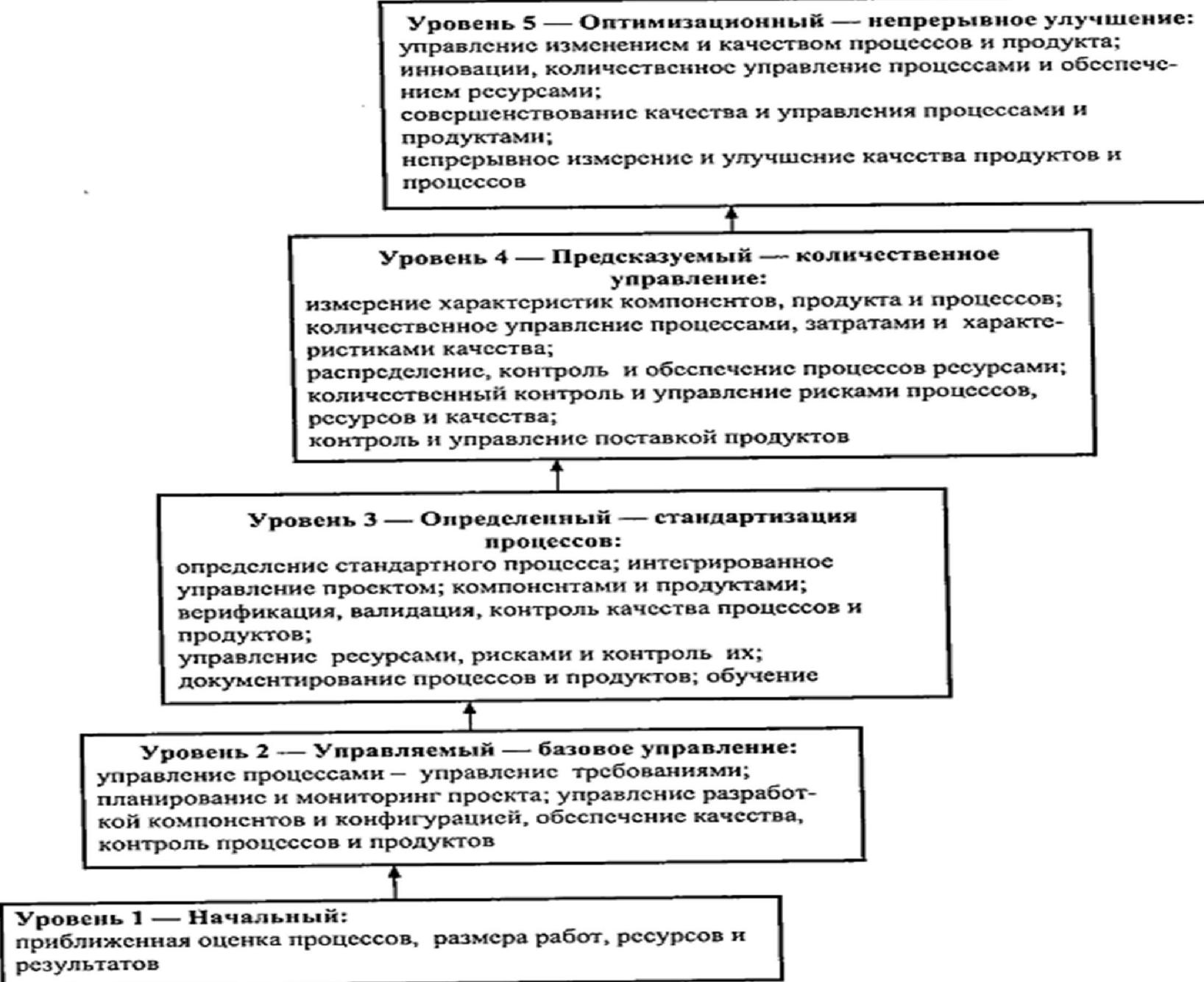
**Модели и процессы управления
проектами программных средств.**

Управление проектами ПС в системе – CMMI.

*Стандарты открытых систем, регламентирующие
структуру и интерфейсы программных средств (POSIX)*

Назначение методологии СММ/СММІ – системы и модели оценки зрелости – состоит в представлении необходимых общих рекомендаций и инструкций предприятиям, производящим ПС, по выбору стратегии совершенствования качества процессов и продуктов, путем анализа степени их производственной зрелости и оценивания факторов, в наибольшей степени влияющих на качество ЖЦ ПС, а также посредством выделения процессов, требующих модернизации.

Для достижения устойчивых результатов ПИ в процессе развития технологии и организации управления жизненным циклом ПС в стандарте



•уровень 1 – Начальный.

- ❖ Массовые разработки проектов ПС характеризуются относительно небольшими размерами программ в несколько тысяч строк, создаваемых несколькими специалистами.
- ❖ Они применяют простейшие не формализованные технологии с использованием типовых инструментальных компонентов

•уровень 2 – Управляемый – базовое управление.

- ❖ Для сложных проектов ПС объемом в десятки и сотни тысяч строк, в которых участвуют десятки специалистов разной квалификации, необходимы организация, регламентирование технологии и унификация процессов деятельности каждого из них.
- ❖ Процессы на этом уровне заранее планируются, их выполнение контролируется, чем достигается предсказуемость результатов и времени

Уровень 3 – Определенный – стандартизация процессов.

❖ При высоких требованиях заказчика и пользователей к конкретным характеристикам качества сложного ПС и к выполнению ограничений по использованию ресурсов, необходимо дальнейшее совершенствование и повышение уровня зрелости процессов УЦПС

• Уровень 4 – Предсказуемый – количественное управление.

- ❖ Для реализации проектов крупных, особенно сложных ПС, в жестко ограниченные сроки и с высоким гарантированным качеством, необходимы активные меры для предотвращения и выявления дефектов и ошибок на всех этапах ЖЦ ПС.
- ❖ Управление должно обеспечивать выполнение процессов в соответствии с текущими требованиями к

•Уровень 5 . Оптимационный – непрерывное совершенствование и улучшение.

- ❖ Дальнейшее последовательное совершенствование и модернизация технологических процессов ЖЦ ПС для повышения качества их выполнения и расширение глубины контроля за их реализацией.
- ❖ Одна из основных целей этого уровня – сокращение проявлений и потерь от случайных дефектов и ошибок путем выявления сильных и слабых сторон используемых процессов.
- ❖ При этом приоритетным является анализ

- В 2003 году американский *Институт программной инженерии* (SEI) опубликовал **новую комплексную модель СММ**, уточняющую и совершенствующую предшествовавшие модели СММ, а также частично учитывающую основные *требования существующих международных стандартов* в области менеджмента программных средств.
- Внедрение этой модели акцентировано на улучшении процессов управления проектами ПС, обеспечении их высокого качества и конкурентоспособности.
- Основная цель – сделать процессы проектов

Модели СММІ представляют помощь специалистам при организации технологии и совершенствовании их продуктов, а также для упорядочения и обслуживания процессов разработки и сопровождения ПС.

Концепция этих моделей покрывает :

- управление,
- оценивание зрелости сложных систем,
- процессы интеграции ПС и совершенствование их разработки.

Компоненты непрерывной и поэтапной моделей в значительной степени подобны, могут выбираться и применяться в разном составе и последовательности методов разви-

Варианты описания моделей построены по единой схеме, которая содержит *общие разделы*:

1. Введение;
2. Модель компонентов;
3. Терминология;
4. Разработка целей и процедур;
5. Структура взаимодействия процессов.
6. Использование модели СММІ – краткие рекомендации для пользователей по применению модели и обучению.

7. Подробные рекомендации для реализации каждого из перечисленных в нем множества процессов, которые учитывают особенности конкретной модели.

Это самый большой раздел, он занимает около 500 страниц из полного объема документа, который составляет свыше 700 страниц.

Стандарт ISO 15504:1-5:2003-2006 регламентирует оценку и аттестацию зрелости процессов создания, сопровождения и совершенствования программных средств и систем, выполняемых предприятиями:

- для установления состояния собственных технологических процессов и их совершенствования;
- для определения пригодности собственных процессов для выполнения определенных требований или классов

Применение стандарта ISO 15504:1-5:2003-2006 направлено на выработку предприятиями и специалистами *культуры постоянного совершенствования зрелости технологий* обеспечения ЖЦ ПС, отвечающих бизнес-целям проектов и оптимизации использования доступных ресурсов.

Серия стандартов ISO 9000:2000

Серия стандартов ISO 9000:2000 разработана, чтобы помочь предприятиям всех типов и размеров внедрить и использовать эффективные *системы менеджмента (административного управления) качества*.

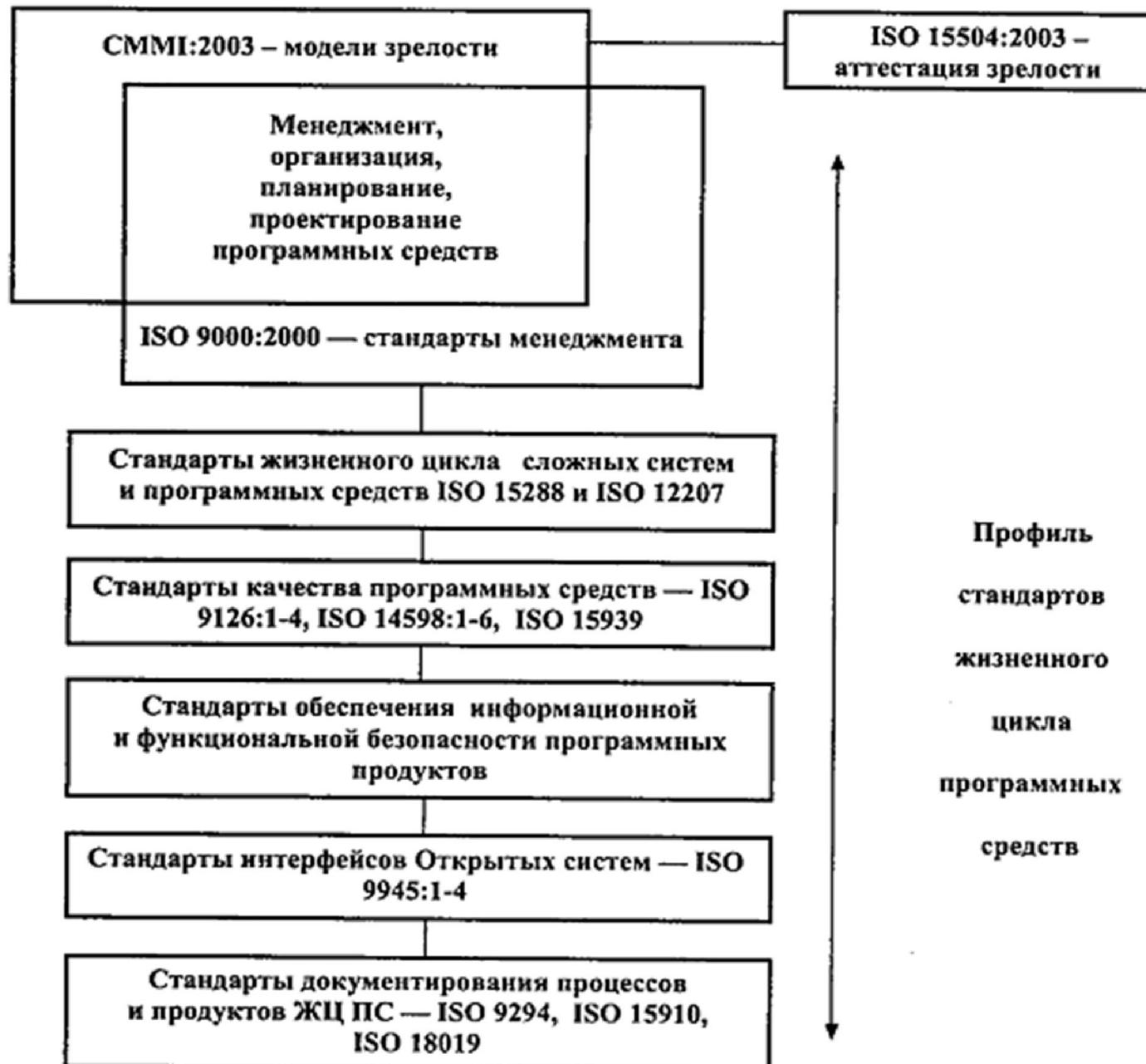
Совместно они образуют комплект согласованных

- 1.**ISO 9000:2000** – представляет введение в системы управления качеством продукции и услуг и словарь качества;
- 2.**ISO 9001:2000** – устанавливает детальные требования для систем управления качеством, достаточные в случае необходимости продемонстрировать способность предприятия, обеспечить соответствие качества продукции и услуг требованиям заказчика;
- 3.**ISO 9004:2000** – содержит руководство по внедрению и применению широко развитой системы управления качеством, чтобы достичь постоянного улучшения деловой деятельности и результатов предприятия.

Стандарты серии ISO 9000:2000 применяют процессный подход в административном управлении системами качества предприятий, а также рассматривают способы быстрого выявления и реализации возможностей для их улучшения.

Структура основных требований и рекомендаций в этих стандартах сведена к четырем объединенным крупным процессам:

- 1.Обязанности и ответственность администрации управления качеством*
- 2.Административное управление*



Стандарт ISO 9003:2004 –
Рекомендации по применению
стандарта ISO 9001:2000 для
программных средств.
Предназначены для регламентирования
менеджмента при приобретении,
поставке, разработке, применении,
сопровождении ***сложных***
программных средств и при их
обслуживании.
Стандарт не содержит ограничений и

- Полное или частичное применение стандарта ISO 9003 целесообразно в различных ситуациях, с учетом технологии, модели жизненного цикла, процессов разработки, последовательности действий и организационной структуры предприятия.
- Его рекомендуется применять как *поддержку процессов программной инженерии* в ISO 9001:2000, совместно со стандартами ISO 12207, ISO 15504, ISO 9126, ISO 14598, ISO 15939.
- Первые четыре раздела практически повторяют содержание аналогичных разделов в ISO 9001:2000

Стандарты открытых систем, регламентирующие структуру и интерфейсы программных средств

- Рядом зарубежных организаций и промышленных фирм под руководством IEEE с 1990 года ведется активная разработка последовательных версий *стандартов интерфейсов открытых систем POSIX (Portable operating system interfaces)*.
- Выполнена большая работа по пересмотру, расширению и реорганизации около двадцати базовых спецификаций POSIX 1990 - 1998 года IEEE 1003. Улучшена систематизация и структура стандартов, усовершенствовано удобство их применения пользователями.
- В результате подготовлен *комплексный проект фундаментального международного стандарта из четырех крупных частей ISO 9945:1-4:2003 (IEEE 1003.1 – 2003)*, объемом свыше трех тысяч страниц.
- Настоящий стандарт – совместная разработка IEEE и The Open Group, он является одновременно стандартом IEEE, стандартом ISO и стандартом Open Group Technical.

Цель документа – стандартизация в программной инженерии обеспечения переносимости программ на уровне исходных текстов.

В нем определены основные интерфейсы операционных систем и окружения, интерфейсы командного интерпретатора, а также программы общих утилит.

Три отдельных крупных тома включают:
базовые определения;

- ❖ системные интерфейсы;
- ❖ команды управления и сервисные программы (утилиты).

Стандарты открытых систем – **POSIX** регламентируют совокупность базовых, системных сервисов для обеспечения унифицированных интерфейсов прикладных программ, специфицированных для языка C, командного языка и совокупности служебных программ.

Основная цель – сделать программы переносимыми на уровне различных исходных языков.

У каждого интерфейса программ существует вызывающая и вызываемая сторона,

POSIX

*Мобильность приложений должна
обеспечиваться благодаря применению
большого числа стандартизованных
системных интерфейсных сервисов и
возможности динамического выяснения
характеристик целевой платформы и
подстройки под них интерфейсов
приложений.*

При формировании концепции стандартов POSIX были поставлены следующие задачи:

- содействовать облегчению и автоматизации переноса кода готовых прикладных программ на иные платформы;
- способствовать определению и унификации интерфейсов программных компонентов заранее при проектировании программных средств, а не только в процессе их реализации;
- сохранять по возможности и учитывать все главные, созданные ранее, унаследованные и используемые программные средства и компоненты;
- определять необходимый минимум интерфейсов компонентов и комплексов программ, для ускорения создания и расширения программных продуктов, а также для анализа, одобрения и утверждения документов;
- развивать стандарты в направлении обеспечения коммуникационных сетей, распределенной обработки данных и защиты информации;
- рекомендовать ограничивать использование объектного кода для программ в простых системах.

Цели данной версии стандарта:

- ❑ минимизировать дополнительную работу для разработчиков прикладных программ,
- ❑ продвинуть мобильность прикладных программ по всей области применения операционной системы **UNIX**.

Новую версию *международных стандартов POSIX* составляет стандарт **ISO 9945:1-4:2003** – ИТ. Интерфейсы переносимых операционных систем.

- Ч.1. Базовые определения.
- Ч.2. Системные интерфейсы.
- Ч.3. Команды управления и сервисные программы.
- Ч.4. Обоснование.

Параллельно с подготовкой и внедрением новых четырех стандартов группы **POSIX**, действуют и применяются **международные стандарты**, углубляющие некоторые возможности и облегчающие создание мобильных приложений:

1) **Стандарт ISO 14252:1996 – Руководство по POSIX окружению открытых систем (OSE).**

- В нем изложена идеология и модель создания мобильных ПС, которые детализирует для пользователей модель комплекса стандартов POSIX.
- Считается, что прикладные программы непосредственно не взаимодействуют с внешним окружением, а связаны с ним только через операционную систему.
- Модель отражает принципы построения интерфейсов прикладных программ с платформой – операционной системой, через которую осуществляется взаимодействие с компонентами внешнего окружения.

Разработка приложений предполагается в кроссрежиме, то есть платформа разработки (инструментальная) может не совпадать с платформой исполнения программ (объектной или целевой).

Результат компиляции программы на инструментальной платформе может быть перенесен для исполнения на целевую платформу.

Определяющими являются два интерфейса между тремя базовыми компонентами:

- 1) Интерфейс между прикладными программами и платформой (операционной системой) – (API).
- 2) Интерфейс между платформой (ОС) и внешним окружением - (EEI).

Внешнее окружение (EEI) включает компоненты:

- человека-машинного взаимодействия с пользователями,
- компоненты информационного

Стандарт ISO 14252:1996 включает также общие принципы и руководство по обеспечению непротиворечивости ПП и их интерфейсов с внешним окружением:

- поддерживающие непосредственное взаимодействие ПП с пользователями;
- регламентирующие интерфейсы с виртуальными терминалами и графическими системами;
- обеспечивающие административное управление файловыми системами и различными, в том числе, распределенными базами данных;
- регламентирующие телекоммуникацию и обмен данными на верхних уровнях **эталонной модели ВОС**;
- обеспечивающие аттестацию и безопасность применения информационных технологий, программ и данных в соответствии со стандартами ВОС и криптографии.

Эталонная модель взаимодействия открытых систем (ВОС) определяет уровни взаимодействия систем, дает им стандартные имена и указывает, какие функции должен выполнять каждый уровень.

Средства взаимодействия делятся на 7 уровней:

- прикладной,
- представления,
- сеансовый,
- транспортный,
- сетевой,
- Канальный,
- физический.

Эталонная модель взаимодействия открытых систем

Перемещение информации между компьютерами различных схем является чрезвычайно сложной задачей. В начале 1980 гг. Международная Организация по Стандартизации (ISO) и Международный Консультативный Комитет по Телеграфии и Телефонии (МККТТ) признали необходимость в создания модели сети, которая могла бы помочь поставщикам создавать реализации взаимодействующих сетей. В тесном сотрудничестве была разработана эталонная модель "Взаимодействие Открытых Систем" (ЭМВОС). Эта модель была описана в рекомендациях X.200 (МККТТ) и ISO 7498 (ISO). [Соответствие ЭМВОС МККТТ и ИСО.](#)



ЭТАЛОННАЯ МОДЕЛЬ ВЗАИМОДЕЙСТВИЯ ОТКРЫТЫХ СИСТЕМ (OSI)

7	Уровень приложений	Обеспечивает интерфейс взаимодействия программ, работающих на компьютерах в сети. С помощью этих программ пользователь получает доступ к сетевым услугам.
6	Уровень представлений	Определяет форматы передаваемой информации. Задачей данного уровня является перекодировка, сжатие и распаковка данных, их шифрование и дешифрование.
5	Сеансовый уровень	Позволяет сетевым приложениям устанавливать, поддерживать и завершать соединение, называемое сетевым сеансом. Обеспечивает синхронизацию. Отвечает за восстановление аварийно прерванных сеансов связи.
4	Транспортный уровень	Сегментирует и повторно собирает данные в один поток. Обеспечивает надежную доставку информации между узлами сети.
3	Сетевой уровень	Обеспечивает соединение и выбор маршрута между двумя конечными системами, которые могут находиться в сетях, расположенных в разных концах земного шара, обеспечивает единую систему адресации.
2	Канальный уровень	Обеспечивает надежную передачу данных через физический канал связи. Решает вопросы физической адресации, доступа к среде передачи, сообщений об ошибках, порядка доставки кадров и управления потоком данных.
1	Физический уровень	Выполняет передачу неструктурированного потока бит по физической среде. Отвечает за топологию, поддержание связи и описывает электрические, оптические, механические и функциональный интерфейсы со средой передачи: напряжения, частоты, длины волн, разъемы, число и функциональность контактов, схемы кодирования сигналов.

Стандарт ISO 13210:1998 – Методы
тестирования для оценки соответствия
стандартам **POSIX** – содержит общую
методологию тестирования для проверки
соответствия интерфейсов прикладных
программ стандартам **POSIX**.



Московский государственный технический
университет им. Н.Э. Баумана

Цикл лекций по дисциплине «Методология программной инженерии»

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук.

rtn@bmstu.ru

Москва - 2022

Лекция 11



**Искусственный интеллект.
Нормативные документы РФ**

1. Национальный стандарт РФ ГОСТ Р 59525-2021 "Информатизация здоровья. Интеллектуальные методы обработки медицинских данных. Основные положения"

Настоящий стандарт устанавливает общие положения комплекса национальных стандартов по применению интеллектуальных методов обработки медицинских данных и определяет для этого комплекса:

1.1. Основные цели и задачи стандартизации интеллектуальных методов обработки медицинских данных;

1.2. Организацию работ по стандартизации интеллектуальных методов обработки медицинских данных;

1.3. Направления применения искусственного интеллекта, используемые в медицине;

1.4. Классификации случаев применения искусственного интеллекта в здравоохранении.

* Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

- * **искусственный интеллект (ИИ):** Отрасль информатики, посвященная разработке систем обработки данных, выполняющих функции, обычно ассоциируемые с человеческим интеллектом, такие как рассуждение, обучение и самосовершенствование.
- * **Примечание** - Комплекс технологических решений, позволяющий имитировать когнитивные функции человека (включая самообучение и поиск решений без заранее заданного алгоритма) и получать при выполнении конкретных задач результаты, сопоставимые, как минимум, с результатами интеллектуальной деятельности человека. Комплекс технологических решений включает в себя информационно-коммуникационную инфраструктуру, программное обеспечение (в том числе, в котором используются методы машинного обучения), процессы и сервисы по обработке данных и поиску решений.
- * (Национальная стратегия развития искусственного интеллекта на период до 2030 г. (Утверждена Указом Президента Российской Федерации от 10 октября 2019 г. N 490)

* 1.1 Основные цели комплекса национальных стандартов интеллектуальных методов обработки медицинских данных:

- повышение эффективности лечебного процесса и управление им в медицинских организациях, учреждениях здравоохранения, органах исполнительной власти в сфере здравоохранения;
- контроль и повышение качества медицинской помощи;
- создание условий для стабильного устойчивого развития предприятий и организаций, обеспечивающих разработку, внедрение и эксплуатацию интеллектуальных систем в сфере здравоохранения;
- создание условий для максимальной доступности и своевременности медицинской помощи населению вне зависимости от социального статуса граждан, уровня их доходов, места жительства и иных факторов;
- создание условий для получения населением Российской Федерации информационных, медицинских и иных услуг, способствующих укреплению здоровья;
- развитие международного сотрудничества и обмена опытом по стандартизации интеллектуальных методов обработки медицинских данных.

1.1. Основные задачи:

- ❖ установление требований к составу и структуре баз знаний о состоянии здоровья, лечебно-диагностическом процессе, ресурсах системы здравоохранения, а также к процессам хранения, обработки и представления этой информации для последующей их обработки методами искусственного интеллекта;
- ❖ установление требований к процессам создания, разметки (подготовки), к структуре, порядкам применения и хранения эталонных наборов данных;
- ❖ установление требований к организации терминологических ресурсов и представлению медицинских знаний;
- ❖ установление требований к информационному взаимодействию между медицинскими приборами, интеллектуальными системами и другими системами автоматизации, используемыми в здравоохранении;
- ❖ установление требований к систематизации стандартов и разработке профилей;

- ❖ установление требований к процессам и результатам технических и клинических испытаний, пострегистрационного, эксплуатационного контроля программного обеспечения и программно-аппаратных комплексов на основе технологий искусственного интеллекта;
- ❖ установление требований к жизненному циклу, менеджменту качества и безопасности программного обеспечения и программно-аппаратных комплексов на основе технологий искусственного интеллекта;
- ❖ установление требований к форме и содержанию описания результатов работы программного обеспечения и программно-аппаратных комплексов на основе технологий искусственного интеллекта в соответствии с решаемыми задачами в сфере медицины и здравоохранения.

Таблица 1 - Технологические категории использования случаев искусственного интеллекта в медицине и здравоохранении

Технология	Область применения
Медицинские вмешательства	Обеспечение высокого качества профилактики, диагностики, лечения и медицинского ухода за счет повышения доступности, точности и аккуратности медицинских вмешательств
Цифровой помощник	Выполнение надлежащего лечения в течение установленных норм времени за счет постоянного мониторинга состояния пациента и оповещения медицинских работников
Машинное обучение	Прогнозирование течения патологического процесса с помощью анализа данных, влияющих на результаты лечения
Глубокое обучение	Возможность обработки большого количества биомедицинских данных разных типов для уменьшения неопределенности при принятии клинических решений о лечении
Обработка изображений	Обработка больших объемов медицинских изображений для выявления заболеваний, диагностики, повышения качества и интенсивности обработки и т.д.
Обработка естественных языков	Перевод длинных описательных наборов символов, например, при интерпретации записей электронных медицинских карт, извлечение и структурирование информации
Распознавание звука	Голосовой ввод данных в медицинскую документацию
Статистические данные	Возможность анализа большого объема медицинских данных с целью прогнозирования состояния пациента, контроля качества медицинской помощи
Анализ больших данных (Big data)	Обработка больших объемов данных из медицинских и прочих информационных систем в целях организации и управления системой здравоохранения, в целях управления здоровьем и качеством жизни населения
Прогнозное моделирование	Применение моделирования для прогнозирования течения и исходов патологического процесса, рисков осложнений, эффективности и исходов лечения (в том числе, в сравнении)

Направление	Область применения
Лучевая диагностика	Автоматизированный контроль качества выполненных исследований (полученных изображений), приоритизация результатов исследований, выявление признаков патологических процессов, поддержка принятия решений при дифференциальной диагностике, морфометрия, сравнительный анализ исследований, выполненных в динамике, формирование проектов описаний результатов исследований, голосовое заполнение медицинской документации
Патоморфология цитология	и Автоматизированный контроль качества выполненных исследований (полученных изображений), приоритизация результатов исследований, выявление признаков патологических процессов, поддержка принятия решений при дифференциальной диагностике, морфометрия, сравнительный анализ исследований, выполненных в динамике, формирование проектов описаний результатов исследований, голосовое заполнение медицинской документации
Дermатология	Обнаружение и классификация злокачественных новообразований кожи (в том числе, при скрининге)
Офтальмология	Обнаружение и классификация глазных болезней по диагностическим изображениям
Терапия	Выявление рисков, прогнозирование осложнений/результатов; система поддержки врачебных решений; подбор терапии. Роботизированное выполнение инвазивных и неинвазивных манипуляций, содействие в уходе за пациентом.
Кардиология	Количественное определение, компьютерное обнаружение патологий, диагностика и дифференциальная диагностика; выявление рисков, прогнозирование исходов, результатов; система поддержки клинических решений
Неврология, урология, хирургия	Прогнозирование осложнений/результатов; система поддержки врачебных решений. Роботизированное выполнение инвазивных и неинвазивных манипуляций, содействие в уходе за пациентом
Анестезиология, отделение интенсивной терапии (ОИТ)	Система непрерывного мониторинга; прогнозирование осложнений/результатов. Роботизированное выполнение инвазивных и неинвазивных манипуляций, содействие в уходе за пациентом
Неотложная помощь	Система транспортировки и сортировки; система непрерывного мониторинга; прогнозирование осложнений/результатов. Роботизированное выполнение инвазивных и неинвазивных манипуляций

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ТЕХНИЧЕСКОМУ
РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ**

**НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ
ГОСТР 59277– 2020**

Системы искусственного интеллекта

**КЛАССИФИКАЦИЯ СИСТЕМ
ИСКУССТВЕННОГО ИНТЕЛЛЕКТА**

Официальное издание

Утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 23 декабря 2020 г. № 1372-ст

Принципы классификации систем искусственного интеллекта

- * Искусственный интеллект как область знаний охватывает все области человеческой деятельности: информатику, математику, философию, психологию, термодинамику, лингвистику, здравоохранение, инженерию, экономику, когнитивные науки и др.
- * Эти знания используются в таких приложениях, как: системы управления, системы принятия решений, многоагентные системы, системы обработки естественного языка, распознавание образов, распознавание речи, обработка знаний, интеллектуальный анализ данных, логистика и другие приложения.
- * Классификация должна отражать существенные (значимые) характеристики системы искусственного интеллекта (СИИ), включая особенности контура управления, в рамках которого используется СИИ и технологии построения и использования знаний.

Базовые классы СИИ целесообразно группировать на основе следующих принципов:

- 1) по классам и категориям объектов в управлении;**
- 2) по технологиям построения, приобретения и использования знаний;**
- 3) по функциям, которые выполняет СИИ в контуре управления;**
- 4) по методам и технологиям, используемым в СИИ;**
- 5) по методам и средствам взаимодействия СИИ с другими системами и человеком-оператором.**

Эти подходы к классификации являются основными.

Каждый из них может иметь иерархическую структуру.

Основания для классификации	Классы
1. По степени автономности	<i>1.1 Автономные системы</i> <i>1. 2. Встроенные системы</i> <i>1.3 Гибридные системы</i>
2. По степени автоматизации	<i>1. Автоматизированные системы</i> <i>2. Автоматические системы</i>
3. По архитектурному принципу	<i>1. Централизованные системы</i> <i>2. Распределенные системы</i>
4. По видам деятельности	<i>1. Государственное управление</i> <i>2. Безопасность</i> <i>3. Общеотраслевое регулирование</i> <i>4. Промышленность</i> <i>5. Здравоохранение</i> <i>6. Торговля</i> <i>7. Финансы и банки</i> <i>8. Транспорт и логистика</i> <i>9. Сельское хозяйство</i> <i>10. «Умный город»</i> <i>11. Экология</i> <i>12. Образование и наука</i> <i>13. Нефть и газ</i>
5. По функциям контура управления	<i>1. Системы с обратной связью</i> <i>2. Системы реального времени</i> <i>3. Адаптивные системы</i> <i>4. Системы формирования цели (Системы целеполагания)</i> <i>5. Системы формирования контура управления и обучения</i>

Основания для классификации	Классы
5. По функциям контура управления (продолжение)	<p>7. <i>Системы идентификации и диагностики</i></p> <p>8. <i>Системы когнитивного моделирования</i></p> <p>9. <i>Системы логического вывода</i></p> <p>10. <i>Системы принятия (поддержки) решений</i></p> <p>11. <i>Экспертно-аналитические системы</i></p> <p>12. <i>Системы оценки достижения цели</i></p> <p>13. <i>Ситуационные центры</i></p> <p>14. <i>Системы прогнозирования</i></p> <p>15. <i>Прочее</i></p>
6. По специализации систем	<p>1. <i>Экспертные системы (управление знаниями)</i></p> <p>2. <i>Игровые системы</i></p> <p>3. <i>Систем естественного языка</i></p> <p>4. <i>Систем компьютерного зрения</i></p> <p>5. <i>Промышленные роботы</i></p> <p>6. <i>Беспилотные аппараты</i></p> <p>7. <i>Прочее</i></p>
7. По комплексности и сложности систем	<p>1. <i>Многоагентные системы</i></p> <p>2. <i>Систем «Большие данные»</i></p> <p>3. <i>Промышленный интернет вещей</i></p> <p>4. <i>Киберфизические системы</i></p> <p>5. <i>Систем жизненного цикла</i></p> <p>6. <i>Систем сетевой экспертизы</i></p> <p>7. <i>Распределенные систем управления</i></p> <p>8. <i>Система распределенных ситуационных центров</i></p> <p>9. <i>Прочее</i></p>

8. По методам обработки информации

1. *Нейросети*
2. *Обучение на примере*
3. *Эволюционные и генетические алгоритмы*
4. *Муравьиные алгоритмы*
5. *Иммунные вычисления*
6. *Глубокое обучение*
7. *Роевые вычисления*
8. *Метод Байеса*
9. *Уменьшение размерности*
10. *Природные вычисления*
11. *Мягкие вычисления*
12. *Кластеризация*
13. *Дерево решений*
14. *Регуляризация*
15. *Аналоговая обработка данных*
16. *Обработка фурье-образов*
17. *Регрессия*
18. *Решение обратных задач*
19. *Система правил*
20. *Прочее*

9. По управлению
знаниями, моделям и
методам обучения

1. *Процедурные*
2. *Декларативные*
3. *Онтологические*
4. *Семантические*
5. *Продукционные*
6. *Фреймовые*
7. *Нейросетевая*
8. *Генетическая*
9. *Логическая*
10. *Статистическая*
11. *Нечеткие знания*
12. *Классификации*
13. *Многомерное представление (3Д, 4Д)*
14. *Функциональные*
15. *Технологические*
16. *Методологические*
17. *Комбинированное обучение*
18. *Непрерывное обучение*
19. *Единовременное обучение*
20. *Прочее*

10. По методам достижения
интеграции и интероперабельности

1. *Системы с интеграцией на базе онтологий*
2. *Системы на базе профилирования*
3. *Системы, использующие классификаторы*
4. *Прочее*

11 По опасности последствий*

1. *Социальная*
2. *Политическая*
3. *Экономическая*
4. *Технологическая*
5. *Техногенная*
6. *Экологическая*
7. *Безопасность государства*

12 По конфиденциальности**

*12.1 Уровень
конфиденциальности (0—3)*

Классификация в соответствии с категорированием

объектов критической информационной

инфраструктуры:

- (1) Социальной значимости (здоровье и жизнь людей);**
- (2) Политической значимости (причинение ущерба государству);**
- (3) Экономической значимости (ущерб субъектам и/или бюджетам);**
- (4) Экологической значимости (воздействие на окружающую среду);**
- (5) Значимость для обороны/безопасности. Правопорядка.**

** Классификация соответствует следующим уровням конфиденциальности:

- (0) - Открытая информация;*
- (1) - Внутренняя информация;*
- (2) - Конфиденциальная информация;*
- (3) - Секретная информация.*

Классы можно характеризовать различными дополнительными аспектами или подклассами, например:

- * наличием/отсутствием внешнего наблюдения, осуществляемого человеком-оператором либо другой автоматизированной системой;
- * степенью понимания системы;
- * степенью реактивности/ отзывчивости;
- * уровнем устойчивости функционирования;
- * степенью надежности и безопасности;
- * видом аппаратной реализации;
- * степенью приспособляемости к внутренним или внешним изменениям;
- * способностью оценивать свою собственную работоспособность/пригодность;
- * способностью принимать решения и планировать.



НАЦИОНАЛЬНЫЙ
СТАНДАРТ
РОССИЙСКОЙ
ФЕДЕРАЦИИ

ГОСТ Р
56823—
2015

Интеллектуальная собственность
СЛУЖЕБНЫЕ РЕЗУЛЬТАТЫ
ИНТЕЛЛЕКТУАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

Издание официальное

1 РАЗРАБОТАН Автономной некоммерческой организацией «Республиканский научно-исследовательский институт интеллектуальной собственности» (РНИИИС)

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 481 «Интеллектуальная собственность»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 3 декабря 2015 г. № 2102-ст

4 ВВЕДЕН ВПЕРВЫЕ

5 ПЕРЕИЗДАНИЕ. Апрель 2017 г.

В настоящее время в Российской Федерации одним из самых сложных вопросов в разрешении проблемы обеспечения баланса интересов заказчика, исполнителя и автора охраняемого результата интеллектуальной деятельности (РИД) является распределение прав на служебные РИД, а также определение условий, размера и порядка выплаты вознаграждения автору за создание и использование служебного РИД. От этого напрямую зависит инновационная мотивация по реализации полученных РИД в инновационные технологии и инновационные продукты.

1 Область применения

Настоящий стандарт устанавливает общие правила регулирования отношений в части создания, выявления, идентификации, выбора способа и форм правовой охраны, установления эффективного режима правовой охраны, выбора вариантов использования служебных РИД и распоряжения правами на служебные РИД, определения порядка, размера и условий выплаты вознаграждения автору служебных РИД в научно-технической сфере, сфере искусства и культуры.

Применение настоящего стандарта позволяет:

- установить единообразные положения (правила) создания, выявления, идентификации, выбора способа и форм правовой охраны, установления эффективного режима правовой охраны служебных РИД;
- выделить критерии охраняемых РИД, которые могут быть признаны служебными;
- обеспечить закрепление авторства на служебные РИД и расширить возможности его защиты при согласовании баланса интересов участников правоотношений, возникающих при создании и использовании служебных РИД;

- создать дополнительные условия (организационно-технические предпосылки) для обеспечения гибкости и выбора вариантов распределения интеллектуальных прав на служебные РИД, эффективного хозяйственного использования служебных РИД и распоряжения правами на служебные РИД в ходе хозяйственной деятельности;
- установить единообразные положения (правила) определения порядка, размера и условий выплаты вознаграждения автору служебного РИД за его создание и использование, как работодателем, так и иными лицами по договору с работодателем;
- снизить риски и управлять рисками работодателя в отношениях с его работниками — авторами служебных РИД, относительно выявления и распределения прав, правовой охраны и использования служебных РИД, в т. ч. экономические издержки и ущерб при выплатах пени по решению суда.



* Лекция 12
*

**Искусственный интеллект.
История развития искусственного интеллекта.
Подходы к пониманию проблемы**

Существуют следующие определения искусственного интеллекта:

- Научное направление, в рамках которого ставятся и решаются задачи аппаратного или программного моделирования тех видов человеческой деятельности, которые традиционно считаются интеллектуальными.
- Интеллектуальная система — это техническая или программная система, способная решать задачи, традиционно считающиеся творческими, принадлежащие конкретной предметной области, знания о которой хранятся в памяти такой системы.
- Направление в информатике, задачей которого является воссоздание с помощью вычислительных систем и иных искусственных устройств разумных рассуждений и действий.
- Способность системы правильно интерпретировать внешние данные, извлекать уроки из таких данных и использовать полученные знания для достижения конкретных целей и задач при помощи гибкой адаптации.

Национальная стратегия развития искусственного интеллекта на период до 2030 года, утвержденная Указом Президента РФ от 10 октября 2019 г. № 490 *"О развитии искусственного интеллекта в Российской Федерации"* определяет следующие основные понятия:

1. Искусственный интеллект - комплекс технологических решений, позволяющий имитировать когнитивные функции человека (включая самообучение и поиск решений без заранее заданного алгоритма) и получать при выполнении конкретных задач результаты, сопоставимые, как минимум, с результатами интеллектуальной деятельности человека.

Комплекс технологических решений включает в себя информационно-коммуникационную инфраструктуру, программное обеспечение (в том числе, в котором используются методы машинного обучения), процессы и сервисы по обработке данных и поиску решений.

2. Технологии искусственного интеллекта - технологии, основанные на использовании искусственного интеллекта, включая компьютерное зрение, обработку естественного языка, распознавание и синтез речи, интеллектуальную поддержку принятия решений и перспективные методы искусственного интеллекта.

3. Перспективные методы искусственного интеллекта - методы, направленные на создание принципиально новой научно-технической продукции, в том числе в целях разработки универсального (сильного) искусственного интеллекта.

Это понятие включает:

- * автономное решение различных задач,
- * автоматический дизайн физических объектов,
- * автоматическое машинное обучение,
- * алгоритмы решения задач на основе данных с частичной разметкой и (или) незначительных объемов данных,
- * обработка информации на основе новых типов вычислительных систем, интерпретируемая обработка данных и другие методы.

4. Смежные области использования искусственного интеллекта - технологии и технологические решения, в которых искусственный интеллект используется в качестве обязательного элемента, включая робототехнику и управление беспилотным транспортом.

5. Набор данных - совокупность данных, прошедших предварительную подготовку (обработку) в соответствии с требованиями законодательства РФ об информации, информационных технологиях и о защите информации необходимых для разработки ПО на основе искусственного интеллекта.

6. Разметка данных - этап обработки структурированных и неструктурированных данных, в процессе которого данным (в том числе текстовым документам, фото- и видеоизображениям) присваиваются идентификаторы, отражающие тип данных (классификация данных), и (или) осуществляется интерпретация данных для решения конкретной задачи, в том числе с использованием методов машинного обучения.

7. Аппаратное обеспечение - система взаимосвязанных технических устройств, предназначенных для ввода (вывода), обработки и хранения данных.

8. Вычислительная система - предназначенные для решения задач и обработки данных (в том числе вычислений) программно-аппаратный комплекс или несколько взаимосвязанных комплексов, образующих единую инфраструктуру.

9. Архитектура вычислительной системы - конфигурация, состав и принципы взаимодействия (включая обмен данными) элементов вычислительной системы.

10. Общедоступная платформа - информационная система для сбора, обработки, хранения и опубликования наборов данных, доступная в информационно-телекоммуникационной сети «Интернет».

11. Открытая библиотека искусственного интеллекта - набор алгоритмов, предназначенных для разработки технологических решений на основе искусственного интеллекта, описанных с использованием языков программирования и размещенных в сети "Интернет".

12. Технологическое решение - технология, программа для ЭВМ, база данных или их совокупность, а также сведения о наиболее эффективных способах их использования

История развития искусственного интеллекта (ИИ)

*История искусственного интеллекта как нового научного направления начинается в середине XX века. К этому времени уже было сформировано множество предпосылок его зарождения. Среди философов давно шли споры о природе человека и процессе познания мира, нейрофизиологи и психологи разработали ряд теорий относительно работы человеческого мозга и мышления, экономисты и математики задавались вопросами оптимальных расчётов и представления знаний

В 1950 году один из пионеров в области вычислительной техники, английский учёный Алан Тьюринг, пишет статью под названием «**Может ли машина мыслить?**», в которой описывает процедуру, с помощью которой можно будет определить момент, когда машина сравняется в плане разумности с человеком, получившую название **теста Тьюринга**.

*Наконец зародился фундамент математической теории вычислений — теории алгоритмов и были созданы первые компьютеры.

История развития искусственного интеллекта в СССР и России

* В 1832 году С. Н. Корсаков опубликовал описание пяти изобретённых им механических устройств, так называемых «интеллектуальных машин», для частичной механизации умственной деятельности в задачах поиска, сравнения и классификации.

* В конструкции своих машин Корсаков впервые в истории информатики применил перфорированные карты, игравшие у него своего рода роль баз знаний, а сами машины по существу являлись предтечами экспертных систем.

* В СССР работы в области искусственного интеллекта начались в 1960-х годах. В Московском университете и Академии наук был выполнен ряд пионерских исследований, возглавленных Вениамином Пушкиным и Д. А. Поспеловым.

* С начала 1960-х М. Л. Цетлин с коллегами разрабатывали вопросы, связанные с обучением конечных автоматов.

* В 1964 году была опубликована работа логика Сергея Маслова «**Обратный метод установления выводимости в классическом исчислении предикатов**», в которой впервые предлагался метод автоматического поиска доказательства теорем в исчислении предикатов.

* В 1966 году В. Ф. Турчиным был разработан язык рекурсивных функций **Рефал**.

* Большой вклад в развитие искусственного интеллекта в СССР внес академик Г. С. Поспелов. Им были разработаны принципы создания комплекса взаимосвязанных человеко-машинных систем планирования разного уровня, построенных на основе специализированных систем экономико-математических моделей и принципов искусственного интеллекта, предложено обоснование необходимости создания систем коллективного диалогового пользования в задачах планирования, управления и проектирования, основанных на принципах искусственного интеллекта.

* *Развитие искусственного интеллекта в России и в мире (Указ Президента РФ)*

* Развитие информационных систем, помогающих человеку принимать решения, началось с появления в 1950-х годах экспертных систем, описывающих алгоритм действий по выбору решения в зависимости от конкретных условий.

* На смену экспертным системам пришло машинное обучение, благодаря которому информационные системы самостоятельно формируют правила и находят решение на основе анализа зависимостей, используя исходные наборы данных (без предварительного

*Увеличение вычислительных возможностей, использование графических процессоров и распределенных архитектур вычислительных систем привело к появлению машинного обучения организованному по принципу нейронных сетей (по аналогии с человеческим мозгом), что привело к значительному повышению качества разрабатываемых технологических решений.

Машинное обучение характеризуется рядом особенностей:

1. Для поиска вычислительной системой непредвзятого решения требуется ввести репрезентативный, релевантный и корректно размеченный набор данных.
2. Алгоритмы работы нейронных сетей крайне сложны для интерпретации и, следовательно, результаты их работы могут быть подвергнуты сомнению и отменены человеком. Отсутствие понимания того, как искусственный интеллект достигает результатов, является одной из причин низкого уровня доверия к современным технологиям искусственного интеллекта и может стать препятствием для их развития.

Подходы к пониманию проблемы

* Нет точного критерия достижения компьютерами «разумности», хотя на заре ИИ был предложен ряд гипотез, например, тест Тьюринга или гипотеза Ньюэлла — Саймона.

Выделяют два основных подхода к разработке ИИ:

* **нисходящий (англ. *Top-Down AI*), семиотический** — создание экспертных систем, баз знаний и систем логического вывода, имитирующих высокоуровневые психические процессы: мышление, рассуждение, речь, эмоции, творчество и т. д.;

* **восходящий (англ. *Bottom-Up AI*), биологический** — изучение нейронных сетей и эволюционных вычислений, моделирующих интеллектуальное поведение на основе биологических элементов, а также создание соответствующих вычислительных систем: нейрокомпьютер или биокомпьютер. Последний подход не относится к науке об ИИ — их объединяет только общая конечная цель.

Тест Тьюринга и интуитивный подход

Эмпирический тест был предложен Аланом Тьюрингом в статье «Вычислительные машины и разум» опубликованной в 1950 году в философском журнале «*Mind*». Целью данного теста является определение возможности искусственного мышления, близкого к человеческому.

Стандартная интерпретация этого теста звучит следующим образом:

«Человек взаимодействует с одним компьютером и одним человеком. На основании ответов на вопросы он должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы — ввести человека в заблуждение, заставив сделать неверный выбор».

Все участники теста не видят друг друга.

Подходы к пониманию проблемы

1. Символьный подход

Исторически символый подход был первым в эпоху цифровых машин, так как именно после создания Лисп, **первого языка символьных вычислений**, у его автора возникла уверенность в возможности практически приступить к реализации этими средствами интеллекта. Символьный подход позволяет оперировать слабоформализованными представлениями и их смыслами.

Успешность и эффективность решения новых задач зависит от умения выделять только существенную информацию, что требует гибкости в методах абстрагирования. Однако гибкость и универсальность выливается в значительные затраты ресурсов для не типичных задач, то есть система от интеллекта

2. Логический подход

- * Логический подход к созданию систем ИИ основан на моделировании рассуждений. Теоретической основой служит логика.
- * Логический подход может быть проиллюстрирован применением для этих целей языка и системы логического программирования Пролог.
- * Программы, записанные на языке Пролог, представляют наборы фактов и правил логического вывода без жесткого задания алгоритма как последовательности действий, приводящих к необходимому результату

3. Агентно-ориентированный подход

Этот подход развивается с начала 1990-х годов и именуется *подходом, основанным на использовании интеллектуальных (рациональных) агентов.*

* Согласно этому подходу, интеллект — это вычислительная часть (грубо говоря, планирование) способности достигать поставленных перед интеллектуальной машиной целей. Сама такая машина будет интеллектуальным агентом, воспринимающим окружающий его мир с помощью датчиков, и способной воздействовать на объекты в окружающей среде с помощью исполнительных механизмов.

* Этот подход акцентирует внимание на тех методах и алгоритмах, которые помогут интеллектуальному агенту выживать в окружающей среде при выполнении его задачи. Так, здесь значительно тщательнее изучаются алгоритмы поиска пути и принятия решений.

Гибридный подход

Гибридный подход предполагает, что только синергийная комбинация нейронных и символьных моделей достигает полного спектра когнитивных и вычислительных возможностей. Например, экспертные правила умозаключений могут генерироваться нейронными сетями, а порождающие правила получают с помощью статистического обучения. Сторонники данного подхода считают, что гибридные информационные системы будут значительно более сильными, чем сумма различных концепций по отдельности.

Символьное моделирование мыслительных процессов

Долгие годы развитие этой науки двигалось по пути **моделирования рассуждений**.

Моделирование рассуждений подразумевает создание символьных систем, на входе которых поставлена некая задача, а на выходе требуется её решение. Как правило, предлагаемая задача уже формализована, то есть, переведена в математическую форму, но либо не имеет алгоритма решения, либо он слишком сложен и трудоёмок. В это направление входят:

- доказательство теорем,
- принятие решений,
- *теория игр*,
- планирование,
- диспетчеризация,
- прогнозирование.

Работа с естественными языками

Немаловажным направлением является **обработка естественного языка**, в рамках которого проводится анализ возможностей понимания, обработки и генерации текстов на «человеческом» языке.

В рамках этого направления ставится цель такой обработки естественного языка, которая была бы в состоянии приобрести знание самостоятельно, читая существующий текст, доступный по Интернету.

Некоторые прямые применения обработки естественного языка включают информационный поиск (в том числе, глубокий анализ текста) и машинный перевод.

Представление и использование знаний

Направление **инженерия знаний** объединяет задачи получения знаний из простой информации, их систематизации и использования.

Это направление исторически связано с созданием **экспертных систем** — программ, использующих специализированные базы знаний для получения достоверных заключений по какой-либо проблеме. Производство знаний из данных — одна из базовых проблем интеллектуального анализа данных.

Существуют различные подходы к решению этой проблемы, в том числе — на основе нейросетевой технологии, использующие процедуры вербализации нейронных сетей.

Машинное обучение

Проблематика *машинного обучения* касается процесса *самостоятельного* получения знаний интеллектуальной системой в процессе её работы. Это направление было центральным с самого начала развития ИИ.

К области машинного обучения относится большой класс задач на *распознавание образов*. Например, это распознавание символов, рукописного текста, речи, анализ текстов.

Многие задачи успешно решаются с помощью биологического моделирования. Особо стоит упомянуть *компьютерное зрение*, которое связано ещё и с робототехникой.

Биологическое моделирование искусственного интеллекта

Сторонники данного подхода считают, что феномены человеческого поведения, его способность к обучению и адаптации есть следствие именно биологической структуры и особенностей её функционирования.

Сюда можно отнести несколько направлений:

- ❖ *Нейронные сети* используются для решения нечётких и сложных проблем, таких как распознавание геометрических фигур или кластеризация объектов.
- ❖ *Генетический подход* основан на идее, что некий алгоритм может стать более эффективным, если позаимствует лучшие характеристики у других алгоритмов («родителей»).
- ❖ *Агентный подход* - ставится задача создания автономной программы — агента, взаимодействующей с внешней средой.

Робототехника

Области *робототехники* и ИИ тесно связаны друг с другом. Интегрирование этих двух наук, создание интеллектуальных роботов составляют ещё одно направление ИИ.

Интеллектуальность требуется роботам, чтобы манипулировать объектами, выполнять навигацию с проблемами локализации (определять местонахождение, изучать ближайшие области) и планировать движение (как добраться до цели).

Машинное творчество

Природа человеческого творчества ещё менее изучена, чем природа интеллекта. Однако эта область существует, и здесь поставлены проблемы написания компьютером музыки, литературных произведений и художественное творчество.

Создание реалистичных образов широко используется в кино и индустрии игр.

Отдельно выделяется изучение проблем технического творчества систем искусственного интеллекта. Теория решения изобретательских задач, предложенная в 1946 году Г. С. Альтшуллером, положила начало таким исследованиям.

Другие области исследований

Существует масса приложений искусственного интеллекта, каждое из которых образует почти самостоятельное направление.

В качестве примеров можно привести:

программирование интеллекта в компьютерных играх, нелинейное управление техническими системами, интеллектуальные системы информационной безопасности.

В перспективе предполагается тесная связь развития ИИ с разработкой квантового компьютера, так как некоторые свойства искусственного интеллекта имеют схожие принципы действия с квантовыми компьютерами.

Можно выделить два направления развития ИИ:

- ❖ решение проблем, связанных с приближением специализированных систем ИИ к возможностям человека, и их интеграции, которая реализована природой человека (*Усиление интеллекта*);

- ❖ создание искусственного разума, представляющего интеграцию уже созданных систем ИИ в единую систему, способную решать проблемы человечества (*Сильный и слабый искусственный интеллект*).

Сильный и слабый искусственные интеллекты —
гипотеза в философии искусственного интеллекта, согласно
которой некоторые формы ИИ могут обосновывать и решать
проблемы.

- теория **сильного** искусственного интеллекта предполагает, что
компьютеры могут приобрести способность мыслить
и осознавать себя как отдельную личность (понимать
собственные мысли). При не обязательно, что их мыслительный
процесс будет подобен человеческому процессу мышления.
- теория **слабого** искусственного интеллекта отвергает такую
возможность.

Термин «сильный ИИ» был введён в 1980 году *Джоном Сёрлом* (в
работе, описывающей мысленный эксперимент «Китайская
комната»), впервые охарактеризовавшим его следующим образом:

**Соответствующим образом запрограммированный компьютер
с нужными входами и выходами и будет разумом, в том смысле,
в котором человеческий разум — это разум.**

Основными задачами развития ИИ являются:

1. Поддержка научных исследований в целях обеспечения опережающего развития искусственного интеллекта.
2. Разработка и развитие ПО, в котором используются технологии ИИ.
3. Повышение доступности и качества данных, необходимых для развития технологий искусственного интеллекта;
4. Повышение доступности аппаратного обеспечения, необходимого для решения задач в области ИИ.
5. Повышение уровня обеспечения российского рынка технологий ИИ квалифицированными кадрами и уровня информированности населения о возможных сферах использования таких технологий.
6. Создание комплексной системы регулирования общественных отношений, возникающих в связи с развитием и использованием технологий ИИ.

**Утверждена в 2019 году ДОРОЖНАЯ КАРТА РАЗВИТИЯ «СКВОЗНОЙ»
ЦИФРОВОЙ ТЕХНОЛОГИИ (СЦТ) «НЕЙРОТЕХНОЛОГИИ И ИИ»**

Нейротехнологии – технологии, которые используют или помогают понять работу мозга, мыслительные процессы, высшую нервную деятельность, в том числе технологии по усилению, улучшению работы мозга и психической деятельности.

В СЦТ выделены семь субтехнологий:

- 1) компьютерное зрение;
- 2) обработка естественного языка;
- 3) распознавание и синтез речи;
- 4) рекомендательные системы и интеллектуальные системы поддержки принятия решений;
- 5) перспективные методы и технологии в ИИ;
- 6) нейропротезирование;
- 7) нейроинтерфейсы, нейростимуляция и нейросенсинг.

Для каждой суб-СЦТ определен текущий уровень готовности (УГТ) (в соответствии с ГОСТ Р 57194.1-2016) выделены наиболее перспективные потенциальные научно-технические и технологические решения (target use-cases).

Субтехнология	УГТ	Сопоставление с мировым уровнем
Компьютерное зрение	6	УГТ по ряду технологических решений в России достигает 6, что соответствует мировому уровню
Обработка естественного языка	6	УГТ по ряду технологических решений в России достигает 6, что соответствует мировому уровню
Распознавание и синтез речи	5	УГТ по ряду технологических решений в России достигает 5, что соответствует мировому уровню
Рекомендательные системы и интеллектуальные системы поддержки принятия решений	7	УГТ по ряду технологических решений в России достигает 5, что соответствует мировому уровню
Перспективные методы и технологии в ИИ	2	УГТ по ряду технологических решений в России достигает 2, что соответствует мировому уровню
Нейропротезирование	5	УГТ по ряду технологических решений в России достигает 5, что соответствует мировому уровню
Нейроинтерфейсы, нейростимуляция и нейросенсинг	3	УГТ по ряду технологических решений в России достигает 3, что соответствует мировому уровню

Направление Развития	Ключевые технические характеристики
1. Компьютерное зрение	<ul style="list-style-type: none"> – Скорость обработки и передачи информации – Требования к качеству фото и видео данных – Объем данных для обучения – Точность анализа (вероятность ошибки, по сравнению с человеком и др. устройствами) – Оптические возможности (определение цветов, расстояний и размеров, поиск по шаблонам и др.) – Требования к аппаратному обеспечению (увеличение разрешения видеосенсоров, динамический диапазон и объем вычислительной мощности для обработки)
2. Обработка естественного языка	<ul style="list-style-type: none"> – Скорость обработки и передачи информации – Необходимый объем текстовых библиотек для обучения системы – Точность анализа (вероятность ошибки, по сравнению с человеком и др. устройствами) – Требования к аппаратному обеспечению (качество и количество устройств ввода, требования к памяти, CPU) – Пословная ошибка
3. Рекомендательные системы и системы поддержки принятия решений	<ul style="list-style-type: none"> – Скорость обработки и передачи информации – Необходимое качество данных для обучения – Объем данных для обучения – Точность (качество вывода) – Интерпретируемость ответа (объяснимость) – Способность адаптироваться к изменениям входных данных (самообучаемость) – Требования к аппаратному обеспечению – Потенциал масштабирования
4. Распознавание и синтез речи	<ul style="list-style-type: none"> – Объем данных для обучения – Требования к качеству аудио данных – Скорость обработки данных – Пословная ошибка – Точность анализа (вероятность ошибки, по сравнению с человеком и др. устройствами) – Акустические возможности (определение частоты, тембра, силы., исключение шумов) – Требования к аппаратному обеспечению (качество и количество устройств ввода, требования к памяти, CPU)
5. Перспективные методы и технологии в ИИ	<ul style="list-style-type: none"> – Энергопотребление – Решение задач экспоненциальной сложности – Количество задач, решаемых с помощью ИИ – Количество успешных решений с применением перспективного метода – Требования к аппаратному обеспечению

1.4. Основные разработчики в России в разрезе суб-СЦТ

В рамках подготовки дорожной карты для каждой из рассматриваемых суб-СЦТ были определены примеры российских решений и их краткое описание.

Примеры разработчиков и решений в рамках суб-СЦТ «Компьютерное зрение»:

- «Яндекс» разрабатывает систему управления беспилотным автомобилем, которая использует лидары, камеры, радары, GPS и IMU (гиростабилизатор) для достижения пятого уровня автономности (полной автономности автомобиля);
- VisionLabs специализируется на создании программных решений и сервисов на базе технологий компьютерного зрения. Основной продукт компании - платформа распознавания лиц VisionLabs Luna.

Примеры разработчиков и решений в рамках суб-СЦТ «Обработка естественного языка»:

- ABBYY — российская компания-разработчик решений в области распознавания текстов (OCR) и лингвистики. Наиболее известные продукты в данной области — система потокового ввода данных ABBYY FlexiCapture и анализа/понимания текста ABBYY Compreno;

- Алиса — виртуальный голосовой помощник, созданный компанией «Яндекс». Алиса распознает естественную речь, имитирует живой диалог, дает ответы на вопросы пользователя и, благодаря запрограммированным навыкам, решает прикладные задачи;

- DeepPavlov — библиотека диалогового ИИ, которая используется для обработки естественного языка и разработки сложных диалоговых систем. Команда проекта стала одной из десяти команд, отобранных для участия в соревновании Alexa Prize Socialbot Grand Challenge 3, многомиллионном университетском конкурсе по улучшению взаимодействия между человеком и компьютером.

Примеры разработчиков и решений в рамках суб-СЦТ «Рекомендательные системы и интеллектуальные системы поддержки принятия решений»:

- Робот Вера — российское программное обеспечение, предназначенное для эффективного подбора кандидатов и автоматизированного проведения интервью. Система анализирует различную информацию о кандидатах и производит поиск похожих кандидатов в доступных источниках информации (социальные сети, работные сайты).
- MyTarget – система персонализированной рекламы для пользователей с использованием ИИ от Mail Group.
- Smart Machine – программное обеспечение, предоставляющее аналитические сервисы клиентам из финансовой сферы, имеющим потребность в получении широкого поведенческого профиля каждого абонента мобильной связи страны с использованием ИИ от OneFactor.

Примеры разработчиков и решений в рамках суб-СЦТ «Распознавание и синтез речи»:

- Алиса/Yandex.SpeechKit – Голосовой помощник от компании «Яндекс», умеет распознавать речь человека, вести простые разговоры, управлять большим количеством ПО. Yandex.SpeechKit – набор инструментов для распознавания и синтезирования речи, позволяющий сторонним разработчикам создавать свои приложения.
- Продукты от компании ЦРТ – различные продукты, позволяющие распознавать речь и звуки, производить автоматическое обслуживание клиентов и другие функции.

Примеры разработчиков и решений в рамках суб-СЦТ «Нейропротезирование и нейроинтерфейсы»:

– ЭкзоАтлет разрабатывает продукты в области протезирования и экзоскелетов. Их продукты помогают людям с ограниченными возможностями, а также улучшают физические способности пользователя.

Примеры разработчиков и решений в рамках суб-СЦТ «Нейросенсинг и Нейростимуляция»:

– Викиум разрабатывает программное обеспечение для отслеживания, анализа, визуализации мозговой активности и усиления когнитивных способностей пользователя.

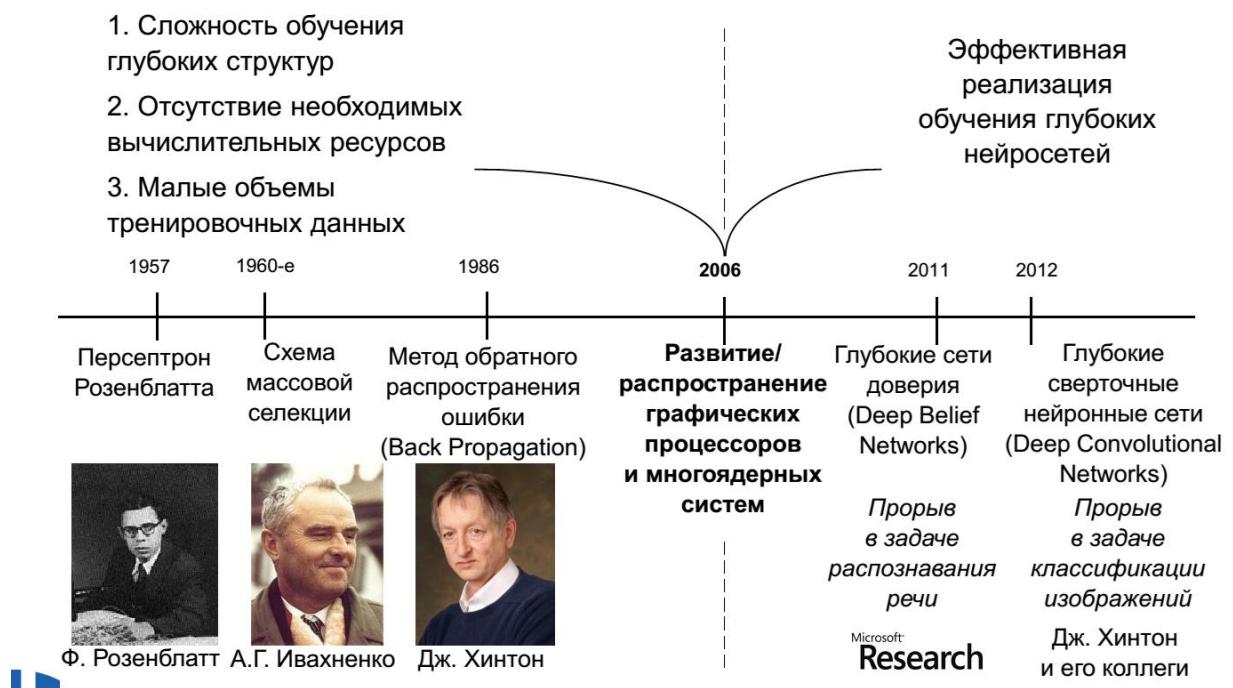
Основными тенденциями развития рынка искусственного интеллекта являются:

- Развитие роботизированных сервисов с помощью ИИ и избавление от человеческого фактора и освобождения человека от монотонной работы путем автоматического создания программного обеспечения.
- Расширение вычислительных и функциональных возможностей программных продуктов.
 - Новые методы машинного обучения, которые ускоряют разработку и реализацию решений в области ИИ в условиях ограниченного количества данных.
 - Повсеместное применение ИИ.

Основными драйверами рынка искусственного интеллекта являются:

- Увеличение объема данных для анализа и повышение доступности данных надлежащего качества.
- Развитие вычислительной архитектуры следующего поколения.
- Развитие перспективных методов анализа данных.

История возникновения и развития



Примеры практических задач (1)

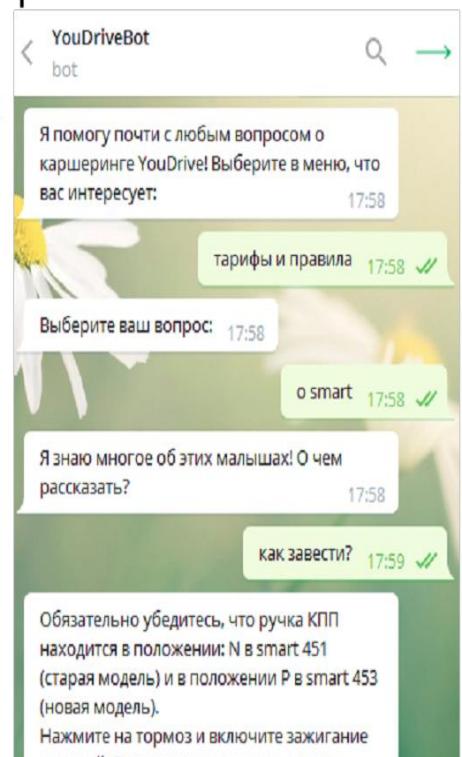
- Наиболее известные примеры успешного практического применения глубокого обучения:
 - Создание искусственного интеллекта, который победил лучшего игрока в AlphaGo



- Технологии автономных автомобилей (Google, Tesla, Uber)



- Рекомендательная система для пользователей онлайн-магазина Amazon
 - Рекомендательная система для пользователей сервиса просмотра видео Netflix
 - Голосовой поиск Google
 - «Персональный помощник» Alexa от Amazon и Cortana от Microsoft. «Персональный помощник» принимает голосовые команды для формирования списка дел, упорядочивает команды, создает напоминания
 - Технология распознавания лиц DeepFace социальной сети Facebook
-
- Задачи из области распознавания естественного языка (онлайн-переводчики, генераторы текста)
 - Задачи из области компьютерного зрения (классификация изображений, детектирование объектов, семантическая сегментация)
- **Генераторы текста** – программы, которые обеспечивают автоматическую генерацию текста, корректного с точки зрения большинства языковых норм, но, как правило, лишенного смысла
- Используются при разработке виртуальных собеседников (чат-ботов и ботов-комментаторов в социальных сетях и блогах)
- Примеры:
- Бот каршеринга YouDrive



Задача классификации изображений

- Задача классификации изображений состоит в том, чтобы поставить в соответствие изображению класс объектов



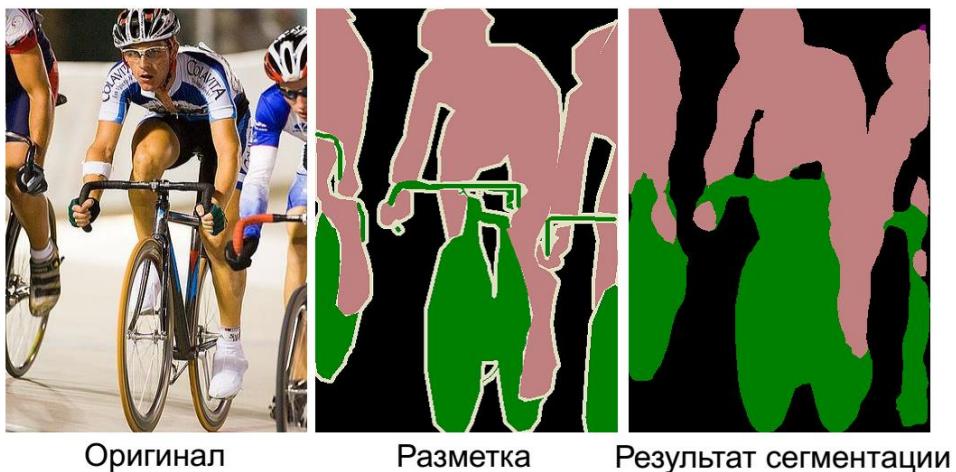
Задача детектирования объектов на изображениях

- Задача детектирования объектов состоит в том, чтобы определить положение прямоугольника, окаймляющего объект заданного класса

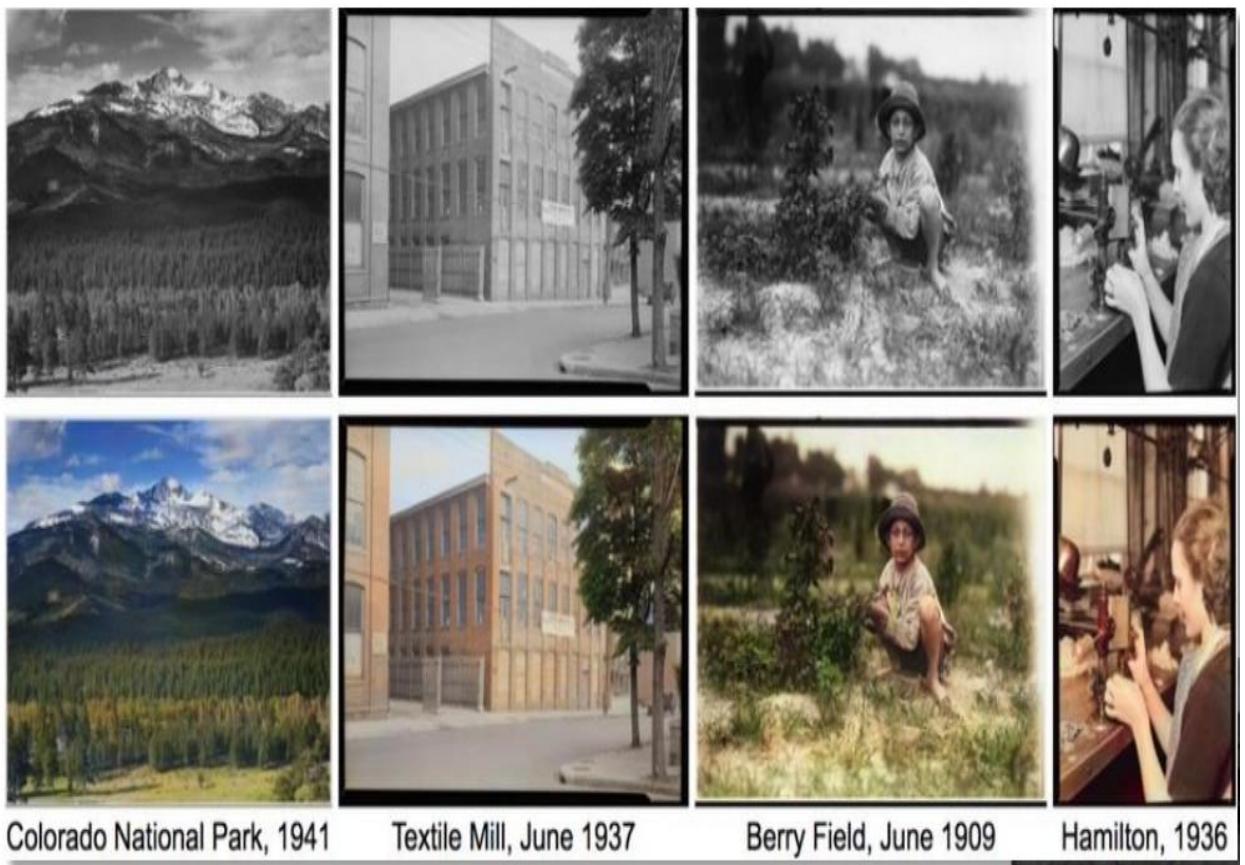


Задача семантической сегментации изображений

- Задача семантической сегментации состоит в том, чтобы каждому пикслю изображения поставить в соответствие класс объектов, которому он принадлежит

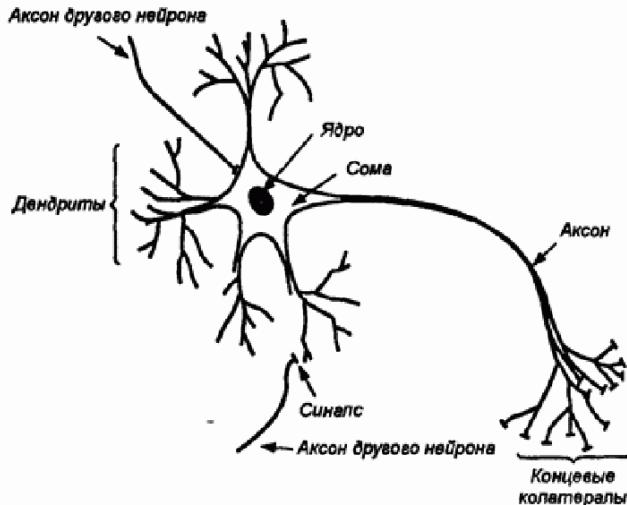


«Раскрашивание» фото и видео



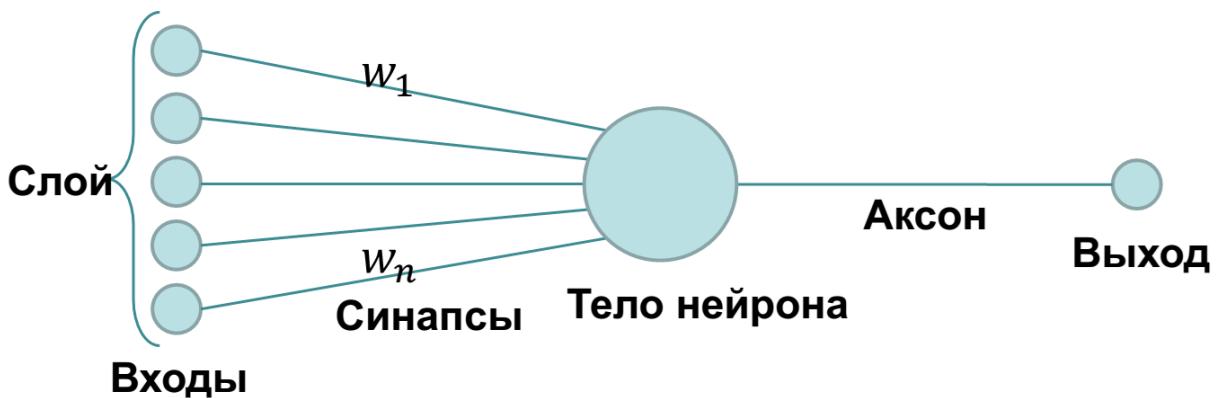
Биологические основы функционирования нейронов в искусственных нейронных сетях (1)

- Искусственная нейронная сеть моделирует способ обработки информации мозгом



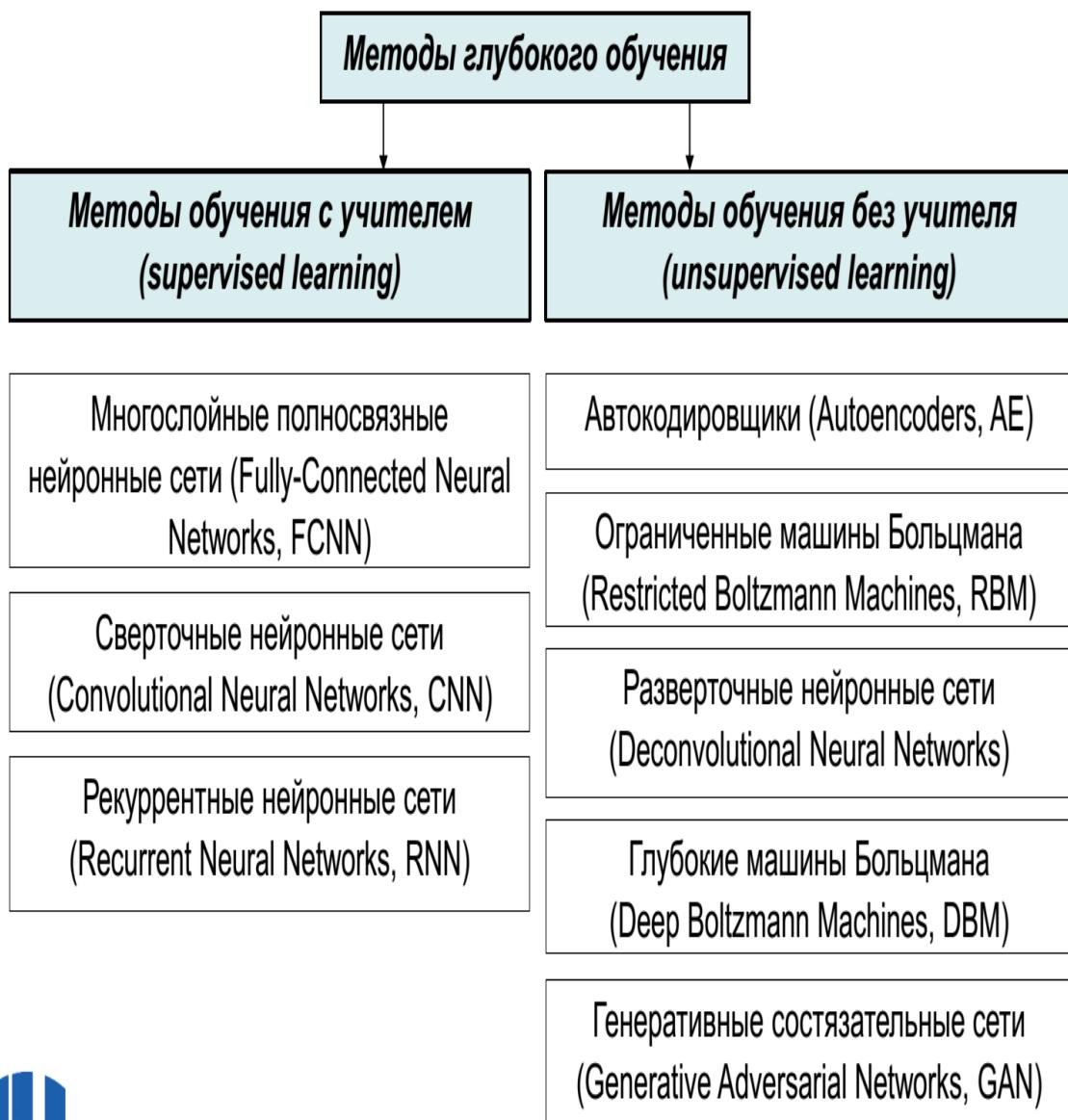
*Осовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика. – 2002. – 344 с.

Модель нейрона искусственной нейросети



- Для каждого текущего нейрона аксоны входных нейронов являются синапсами, аксон текущего нейрона – синапсом выходного нейрона
- Нейроны одного уровня образуют **слой**
- Обучение нейронной сети сводится к настройке весов синаптических каналов

Классификация моделей по способу обучения



Материалы из образовательного курса:

Введение в глубокое обучение

При поддержке компании Intel

Кустикова Валентина,
к.т.н., ст.преп. каф. МОСТ ИИТММ,
ННГУ им. Н.И. Лобачевского

Задачи, решаемые с помощью машинного обучения и искусственного интеллекта

Все задачи, решаемые с помощью машинного обучения (machine learning, ML) и искусственного интеллекта (artificial intelligence, AI), относятся к одной из следующих категорий:

- 1) Задача регрессии – прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число.**
- 2) Задача классификации – получение категориального ответа на основе набора признаков. Имеет конечное количество ответов (как правило, в формате «да» или «нет»).**
- 3) Задача кластеризации – распределение данных на группы.**
- 4) Задача уменьшения размерности – сведение большого числа признаков к меньшему (обычно 2–3) для удобства их последующей визуализации.**
- 5) Задача выявления аномалий – выявление отклонений от стандартных случаев. Эта задача похожа на задачу классификации, но более сложна в обучении.**

Модели искусственного интеллекта/машинного обучения – это математические алгоритмы, которые с помощью вклада человека (эксперта) «обучаются» на наборах данных воспроизводить решение, которое эксперт примет при анализе такого же набора данных. Обучившись воспроизводить решение эксперта, модель в дальнейшем функционирует самостоятельно, позволяя таким образом автоматизировать решение задач. В идеале модель должна также обосновывать своё решение, чтобы помочь интерпретировать процесс принятия решения.

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Искусственная нейронная сеть (ИНС) — математическая модель, а также ее программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей.

- В общем случае ИНС может состоять из нескольких слоев простейших процессоров (нейронов), каждый из которых осуществляет некоторое математическое преобразование (вычисляет результат математической функции) над входными данными и передает полученный результат на следующий слой или на выход сети.
- *ИНС решают проблемы распознавания образов, выполнения прогнозов, оптимизации, ассоциативной памяти и управления.*
- Простейшая ИНС способна к обучению и может находить простые взаимосвязи в данных. Более сложная модель ИНС будет иметь несколько скрытых слоев нейронов, перемежаемых слоями, которые выполняют сложные логические преобразования. Каждый последующий слой сети ищет взаимосвязи в предыдущем. Такие нейросети способны к глубокому (глубинному) обучению (рис. 1).

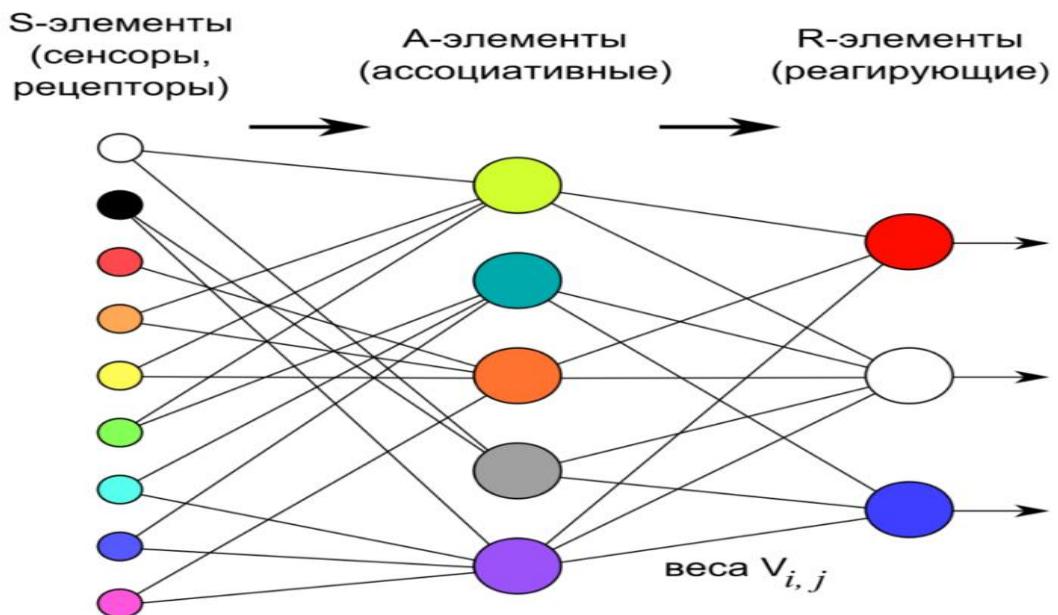


Рис. 1. Модель нейронной сети

1. **Персептрон** – это простейшая модель нейросети, состоящая из одного нейрона. Нейрон может иметь произвольное количество входов, а один из

них обычно тождественно равен 1. Этот единичный вход называют **смещением**. Каждый вход имеет свой собственный вес. При поступлении сигнала в нейрон вычисляется взвешенная сумма сигналов, затем к сигналу применяется функция активации и сигнал передается на выход. Такая простая сеть способна решать ряд задач: выполнять простейший прогноз, регрессию данных и т.п., а также моделировать поведение несложных функций. Главное для эффективности работы этой сети – линейная разделимость данных.

2. Многослойный персептрон – обобщение однослоистого персептрана. Слоем называют объединение нейронов; на схемах, как правило, слои изображаются в виде одного вертикального ряда (в некоторых случаях схему могут поворачивать, и тогда ряд будет горизонтальным). Многослойный персептрон состоит из некоторого множества входных узлов (рис. 2), нескольких скрытых слоев вычислительных нейронов и выходного слоя.

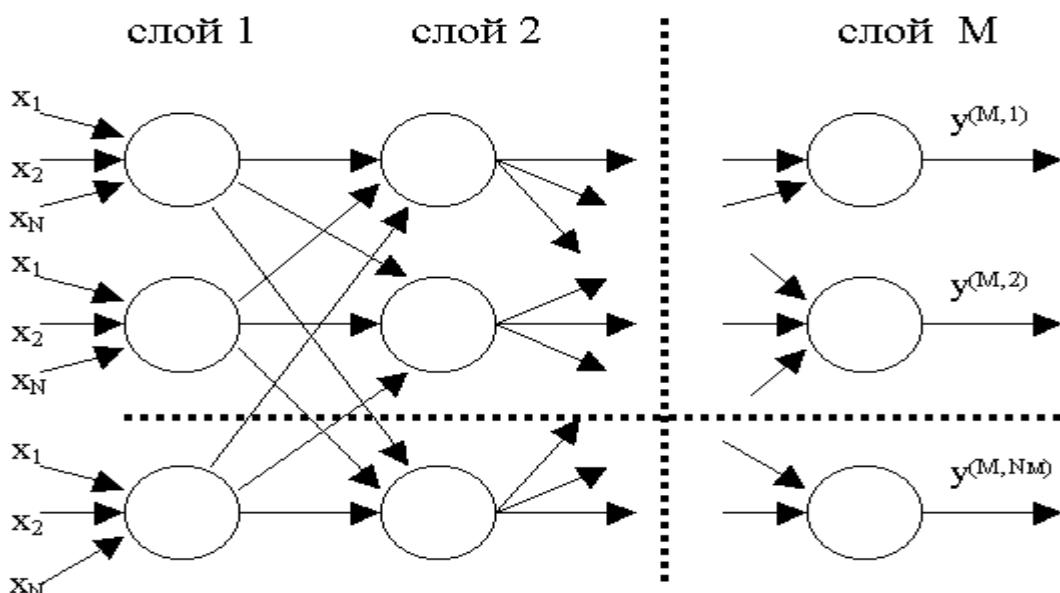


Рис. 2. Многослойный персептрон

Отличительные признаки многослойного персептрана:

- Все нейроны обладают нелинейной функцией активации, которая является дифференцируемой.
- Сеть достигает высокой степени связности при помощи синаптических соединений.
- Сеть имеет один или несколько скрытых слоев.

Такие многослойные сети еще называют **глубокими**. Эта сеть нужна для решения задач, с которыми не может справиться персептрон – в частности, линейно неразделимых задач. Многослойный персептрон является

единственной универсальной нейронной сетью, универсальным аппроксиматором, способным решить любую задачу машинного обучения.

Именно поэтому, если исследователь не уверен, нейросеть какого вида подходит для решения стоящей перед ним задачи, он выбирает сначала многослойный персептрон.

3. Сверточная нейронная сеть (Convolutional neural network, CNN) - сеть, которая обрабатывает передаваемые данные не целиком, а фрагментами.

Данные последовательно обрабатываются, а после передаются дальше по слоям. Сверточные нейронные сети состоят из нескольких типов слоев: сверточный слой, субдискретизирующий слой, слой полносвязной сети (когда каждый нейрон одного слоя связан с каждым нейроном следующего). Слои свертки и подвыборки (субдискретизации) чередуются и их набор может повторяться несколько раз. К конечным слоям часто добавляют персептроны, которые служат для последующей обработки данных (рис. 3).

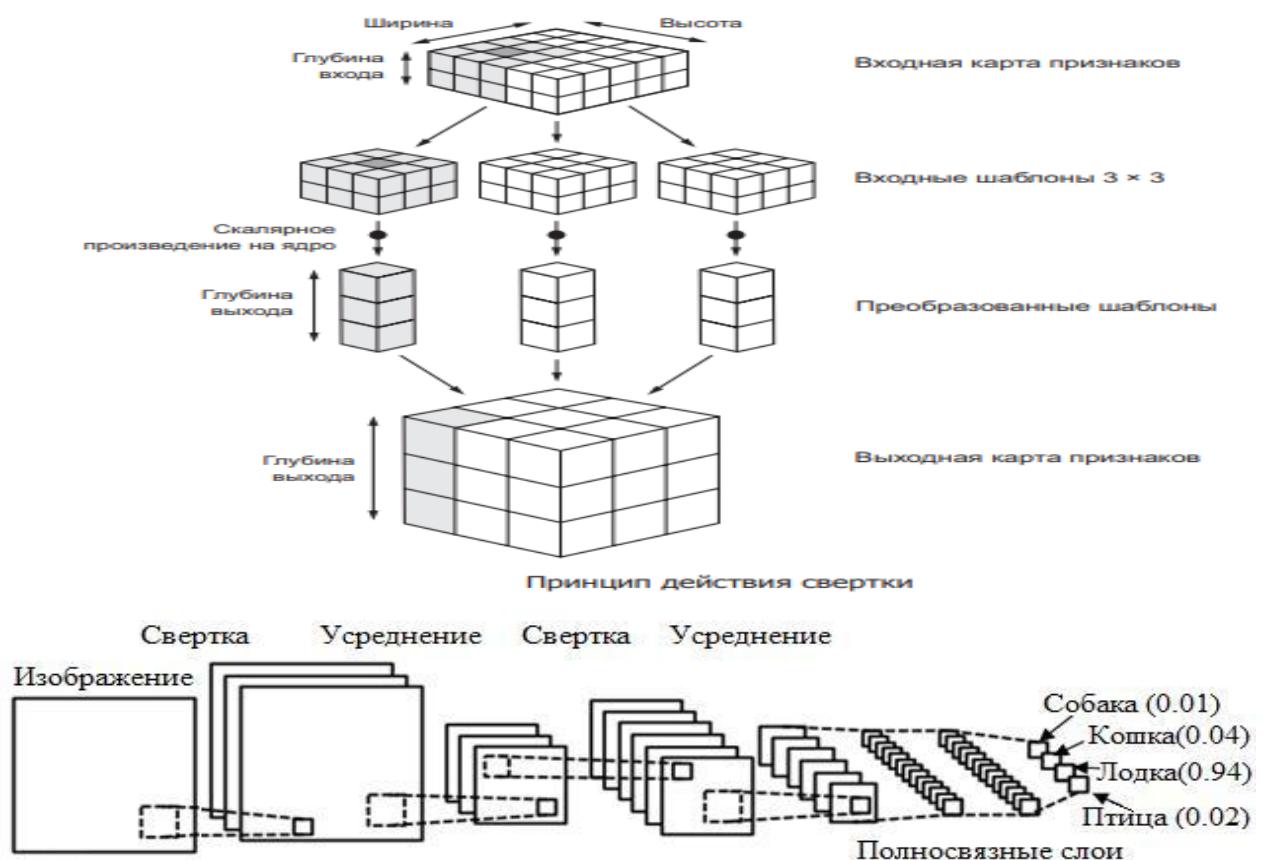


Рис.3. Сверточная нейронная сеть

Суть операции свертки в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно. Затем результат суммируется и записывается в аналогичную позицию выходного изображения для перехода от конкретных особенностей изображения к более

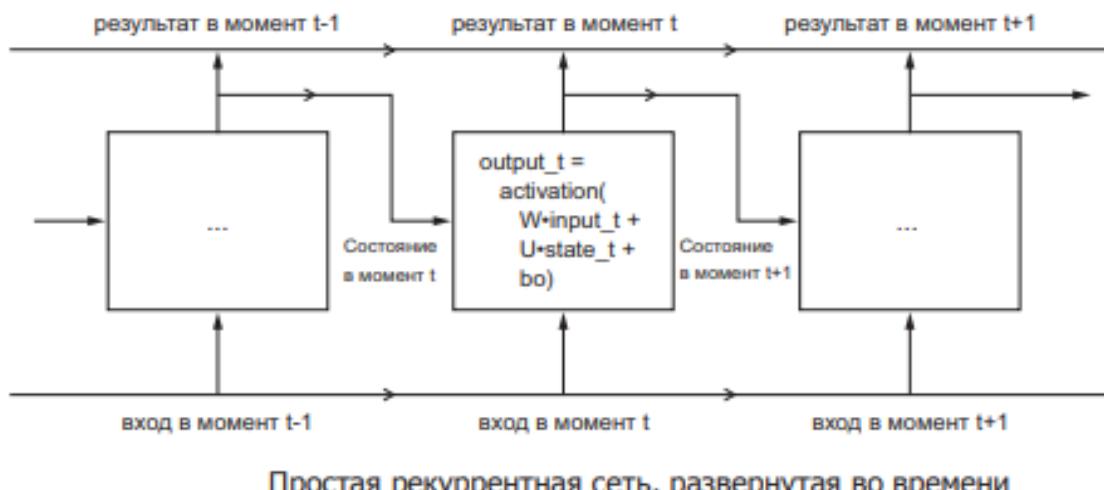
абстрактным деталям, и далее – к ещё более абстрактным, вплоть до выделения понятий высокого уровня (присутствует ли что-либо искомое на изображении).

Описанные сети относятся к **сетям прямого распространения**.

Сверточные нейронные сети решают следующие задачи: классификация, детекция (поиск и определение объектов) и сегментирование.

4. Рекуррентная нейронная сеть (Recurrent neural network, RNN)

RNN – сеть, в которой соединения между нейронами образуют ориентированный цикл, т.е. в сети имеются обратные связи. При этом информация к нейронам может передаваться как с предыдущих слоев, так и от самих себя с предыдущей итерации (задержка) (рис. 4).



Простая рекуррентная сеть, развернутая во времени

Рис. 4. Рекуррентная нейронная сеть

Характеристики сети:

- Каждое соединение имеет свой вес, который также является приоритетом.
- Узлы подразделяются на два типа: вводные и скрытые.
- Информация, находящаяся в нейронной сети, может передаваться как по прямой – слой за слоем, так и между нейронами.

Особенность рекуррентной нейронной сети состоит в том, что она имеет «области внимания». Данная область позволяет задавать фрагменты передаваемых данных, которым требуется усиленная обработка.

Информация в рекуррентных сетях со временем теряется со скоростью, зависящей от активационных функций. Данные сети применяются в распознавании и обработке текстовых данных.

5. Сеть долговременной краткосрочной памяти (Long short-term memory, LSTM) – особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям. Поэтому LSTM подходит для прогнозирования различных изменений при помощи экстраполяции (выявление тенденции на основе данных), а также в любых задачах, где важно умение «держать контекст» (рис. 5).

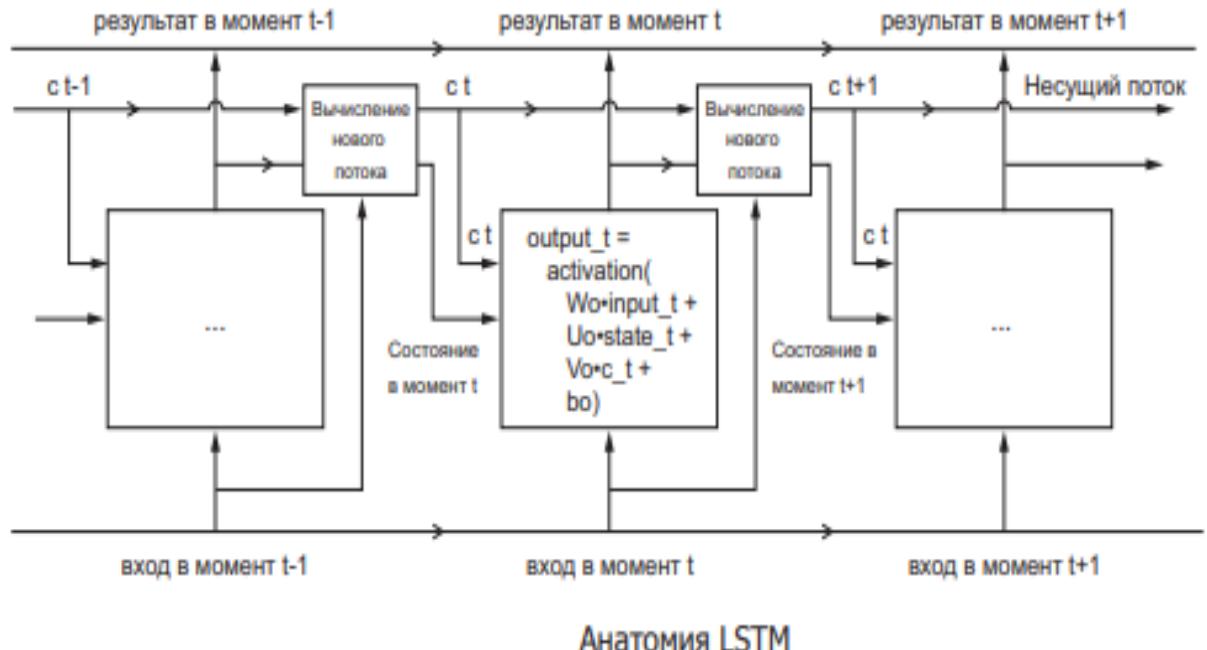


Рис. 5. Сеть долговременной краткосрочной памяти

Любая рекуррентная нейронная сеть имеет форму цепочки повторяющихся модулей нейронной сети. В RNN структура одного такого модуля очень проста, например, он может представлять собой один слой с функцией активации \tanh (гиперболический тангенс). Структура LSTM также напоминает цепочку, но вместо одного слоя нейронной сети она содержит четыре, и эти слои взаимодействуют особым образом.

- 1. Первый шаг в LSTM** – определить, какую информацию можно выбросить из состояния ячейки. Это решение принимает сигмоидальный слой, называемый «слоем фильтра забывания» (forget gate layer).
- 2. Следующий шаг** – решить, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей.
 - Сначала сигмоидальный слой под названием «слой входного фильтра» (input layer gate) определяет, какие значения следует обновить.
 - Затем \tanh -слой строит вектор новых значений-кандидатов, которые можно добавить в состояние ячейки.

- Затем происходит замена старых значений на новые, после чего происходит расчет выходных данных.

6. Самоорганизующаяся нейронная сеть (например, сеть Кохоннена, рис. 6 и 7) – нейронная сеть, структура которой имеет один слой нейронов без коэффициентов смещения (тождественно единичных входов). Процесс обучения сети происходит при помощи метода последовательных приближений. Нейронная сеть подстраивается под закономерности входных данных, а не под лучшее значение на выходе. В результате обучения сеть находит область, в которой находится лучший нейрон. Он на выходе будет иметь значение равное 1. Остальные нейроны, сигнал на которых получился меньше, получит значение равное 0.

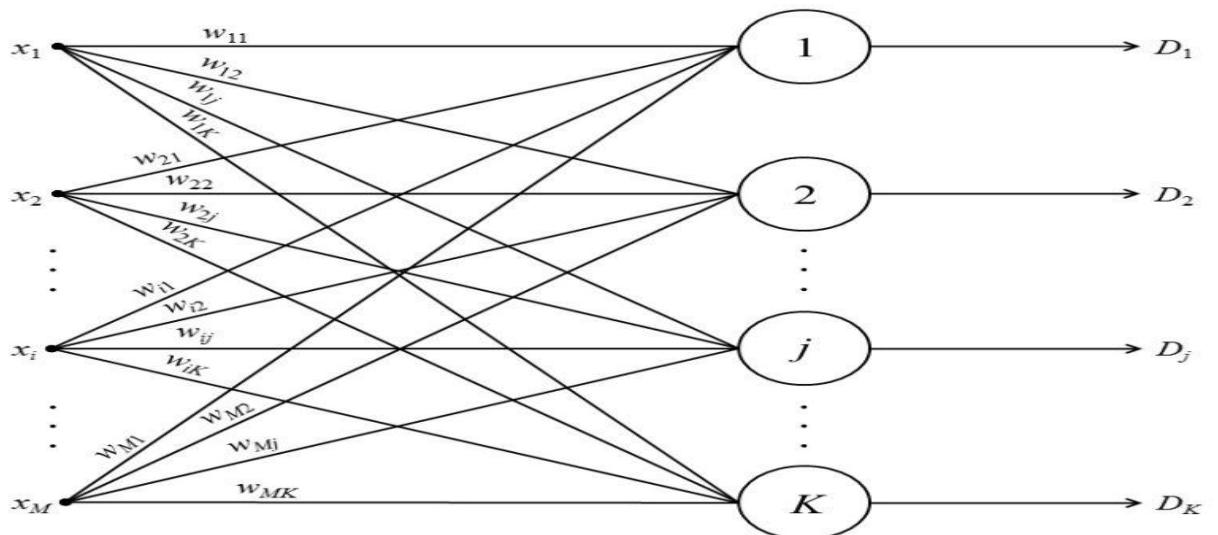


Рис. 6. Сеть Кохоннена

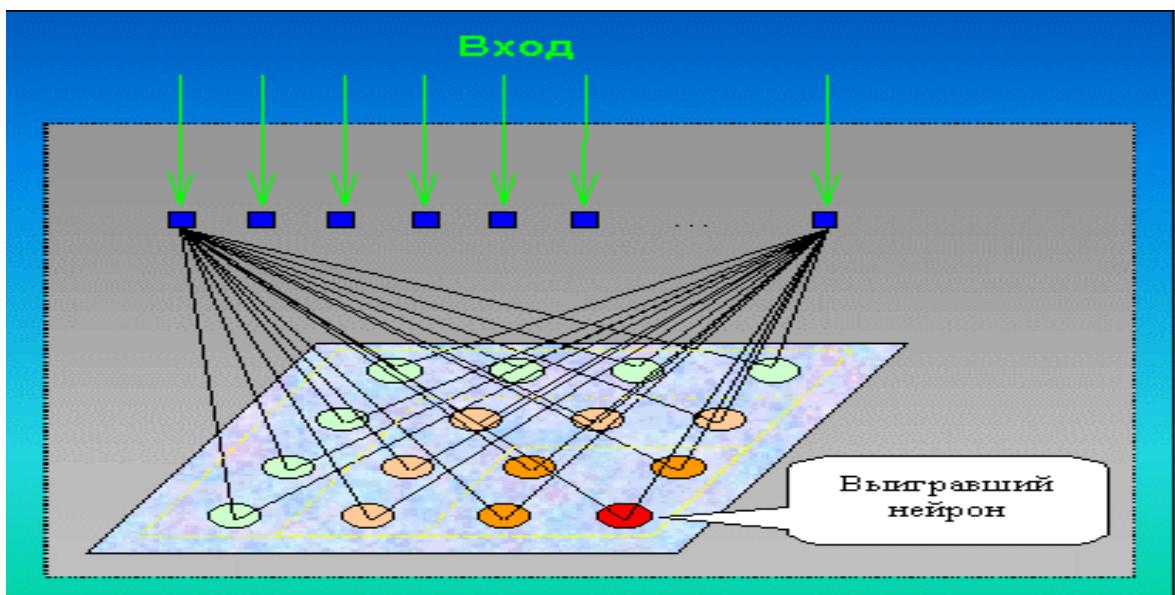
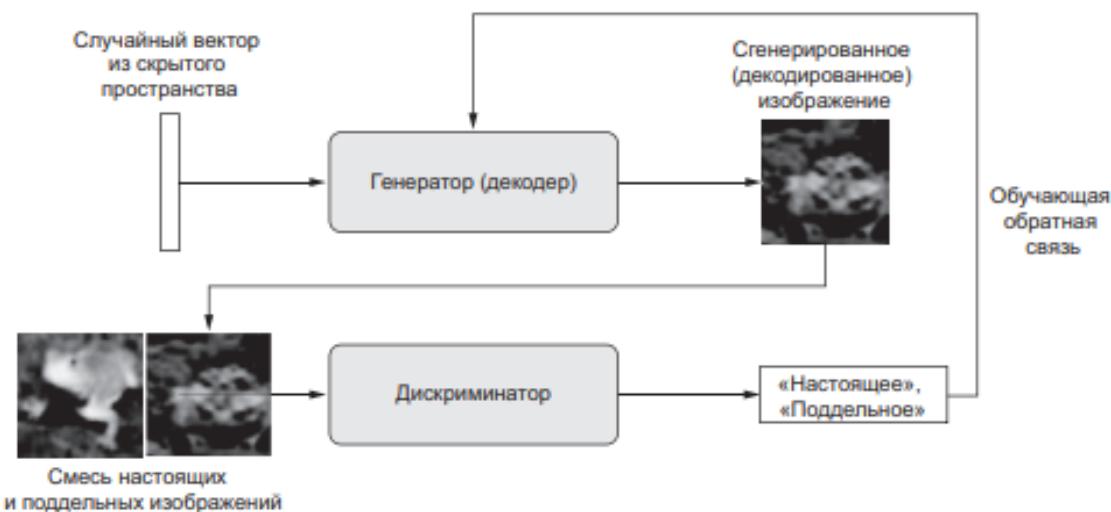


Рис. 7. Схема активации нейронов сетей Кохоннена

Такие сети часто используются в задачах кластеризации и классификации.

7. Генеративно-состязательная нейросеть (Generative adversarial network, GAN) — архитектура, состоящая из генератора и дискриминатора. Генератор и дискриминатор осуществляют работу друг против друга. GAN по своей сути имитируют любое распределение данных. GAN обучаются создавать структуры, похожие на сущности из реального мира в области изображений, музыки, речи, прозы (рис. 8).



Генератор преобразует случайные скрытые векторы в изображения, а дискриминатор стремится отличить настоящие изображения от сгенерированных искусственно. Генератор обучается обманывать дискриминатор

Рис. 8. Генеративно-состязательная нейросеть

Дискриминационные алгоритмы пытаются классифицировать входные данные. Учитывая особенности полученных данных, сети стараются определить категорию, к которой данные относятся.

Генеративные алгоритмы пытаются подобрать образы к данной категории.

Шаги, которые проходит GAN:

- Генератор получает случайное число и возвращает изображение.
- Это сгенерированное изображение подается в дискриминатор наряду с потоком изображений, взятых из фактического набора данных.
- Дискриминатор принимает как реальные, так и поддельные изображения и возвращает вероятности, числа от 0 до 1, причем 1 представляет собой подлинное изображение и 0 представляет фальшивое.

Дискриминатор представляет собой стандартную сверточную сеть, которая может классифицировать изображения, подаваемые на нее с помощью

биномиального классификатора, распознающего изображения как реальные или как поддельные.

8. Объяснимый искусственный интеллект (Explainable AI, XAI)

Объяснимый искусственный интеллект (Explainable AI, XAI) – модель, которая могла бы в перспективе объяснять механизмы, лежащие за алгоритмами машинного обучения.

Множество решений, применяющих алгоритмы ИИ, представляют собой подобие «черного ящика», – зачастую не только конечные пользователи, но и сами разработчики не могут точно определить, как именно модель машинного обучения пришла к тем или иным выводам в ходе обработки исходных данных.

Понимание алгоритмов работы искусственного интеллекта позволит разработчикам точно оценивать влияние входных признаков на выходной результат модели, выявлять необъективности и недостатки, связанные с работой модели, а также проводить тонкую настройку и оптимизацию ИИ.

Для пользователей объяснимость результата работы ИИ важна в части понимания причин выводов, сделанных моделью, а для экспертов – для объяснения тех выводов, которые на первый взгляд не имеют под собой оснований. Потребность в объяснимом искусственном интеллекте со стороны общества подтверждается такими документами, как:

- ***статья 22 Общего регламента по защите данных (ЕС)*** дает человеку право требовать объяснения того, как автоматизированная система приняла решение, которое его затрагивает;
- ***Закон об ответственности за работу алгоритмов (США)*** прямо требует от компаний предоставить оценку рисков для конфиденциальности или безопасности личности, создаваемых автоматизированной системой принятия решений, а также рисков, которые способствуют принятию неточных, несправедливых, предвзятых или дискриминационных решений;
- ***Закон о равных возможностях получения кредитов (США)*** устанавливает возможность объяснения причин получения или отказа в кредите, что представляется затруднительным при использовании искусственного интеллекта на основе «черного ящика».

Агентство перспективных оборонных исследовательских проектов (DARPA) Министерства обороны США, ответственное за разработку новых технологий, проводит программу Explainable AI (XAI).

В национальных программах развития технологий таких стран, как Франция, Норвегия, Индия, Южная Корея и других также говорится о необходимости развития XAI как неотъемлемой части государственной политики. Объяснимость модели может быть рассмотрена на всех этапах разработки искусственного интеллекта, как для изначально интерпретируемых моделей ИИ (линейная и логистическая регрессия, деревья решений и другие), так и для моделей на основе «черного ящика» (персептрон, сверточная и рекуррентная нейронные сети, сеть долгосрочной кратковременной памяти и другие).

Компании IBM, Microsoft, Google и opensource-сообщество предоставляют пользователям наборы инструментов и программные пакеты для объяснимого искусственного интеллекта.

Развитие технологий XAI находится на раннем этапе – большинство методов все еще находится в стадии разработки, появляются новые комбинированные методы, в ходе экспериментов часть теорий подвергаются критике. Количество и уровень вовлеченных участников исследований свидетельствуют о важности рассматриваемой тематики.

Методы XAI могут быть использованы для прогнозирования рисков и угроз при распространении информации, но наибольший эффект от технологии может быть получен при развитии методов XAI, выявляющих различные виды разнородных неструктурированных медиаматериалов. Это необходимо учитывать при разработке концептуальной архитектуры информационных систем ведомства.

9. Обоснование необходимости использования объяснимого искусственного интеллекта (XAI)

Модели глубокого обучения и нейронные сети работают на основе скрытых слоев (т.е. между входными данными и выходным результатом существует несколько уровней математической обработки и принятия решений), но демонстрируют результаты лучше, чем базовые алгоритмы машинного обучения. Использование таких моделей в сферах с повышенным уровнем ответственности за принятые решения привело к созданию концепции объяснимого искусственного интеллекта, в рамках которой исследуются методы анализа или дополнения моделей ИИ, позволяющие сделать внутреннюю логику и выходные данные алгоритмов прозрачными и интерпретируемыми для человека (рис. 9).

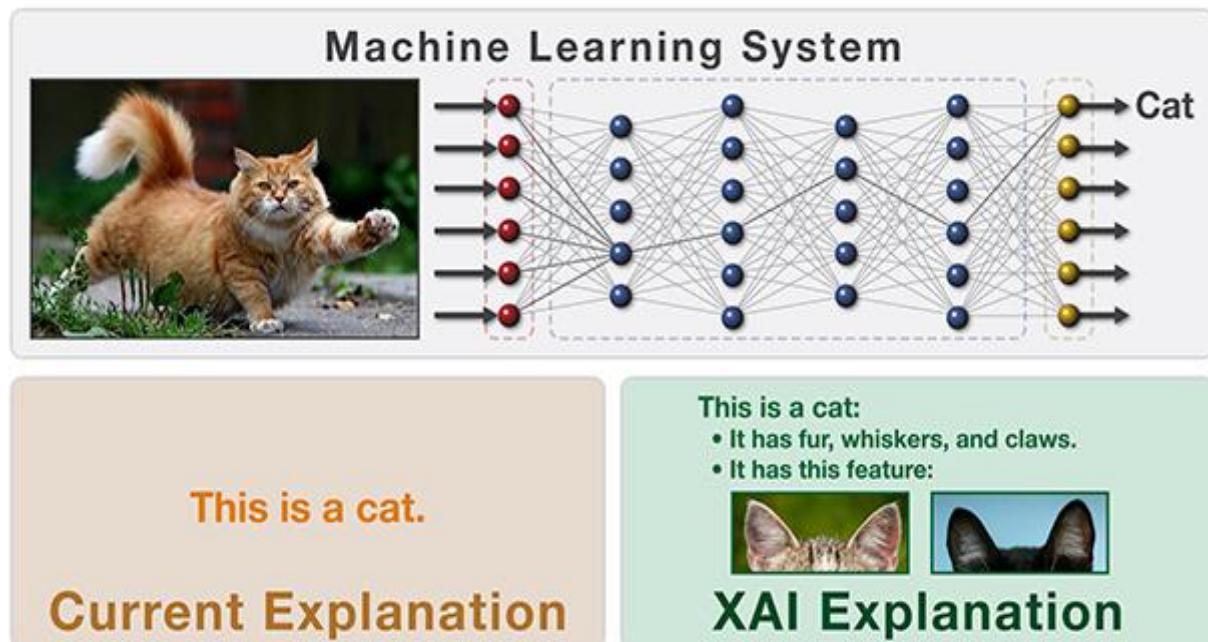


Рис. 9. Представление объяснения работы модели

Объяснимость ИИ будет включать в себя три составляющие: симулируемость, разложимость, алгоритмическую прозрачность.

Симулируемость означает возможность анализа модели человеком. Наиболее важным критерием для симулируемости является сложность модели. Простые, но обширные (со слишком большим количеством правил) системы, основанные на правилах, не соответствуют этой характеристике, тогда как одиночная нейронная сеть персептрана попадает в нее.

Разложимость означает способность объяснить каждую из частей модели (входные данные, параметры и выходные данные). Громоздкие функции не соответствуют данному критерию.

Алгоритмическая прозрачность означает способность пользователя понять процесс, которому следует модель ИИ, чтобы произвести любой заданный вывод из ее входных данных.

Линейная модель ИИ считается прозрачной, потому что ее поверхность ошибок (математическая интерпретация механизма обучения ИИ) понятна и может быть рассмотрена, что дает пользователю достаточно знаний о том, как модель будет действовать в каждой ситуации, с которой он может столкнуться.

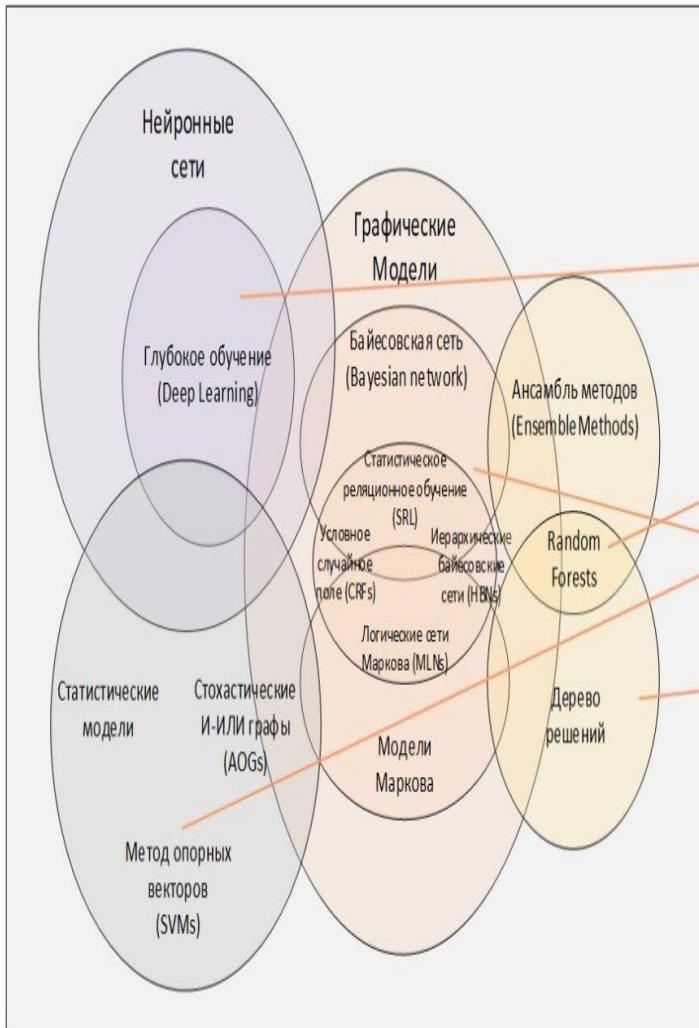
В глубоких архитектурах ИИ этого не происходит, поскольку поверхность ошибок может быть непрозрачной и ее нельзя полностью наблюдать, соответственно, решение необходимо аппроксимировать с

помощью эвристической оптимизации (например, с помощью стохастического градиентного спуска).

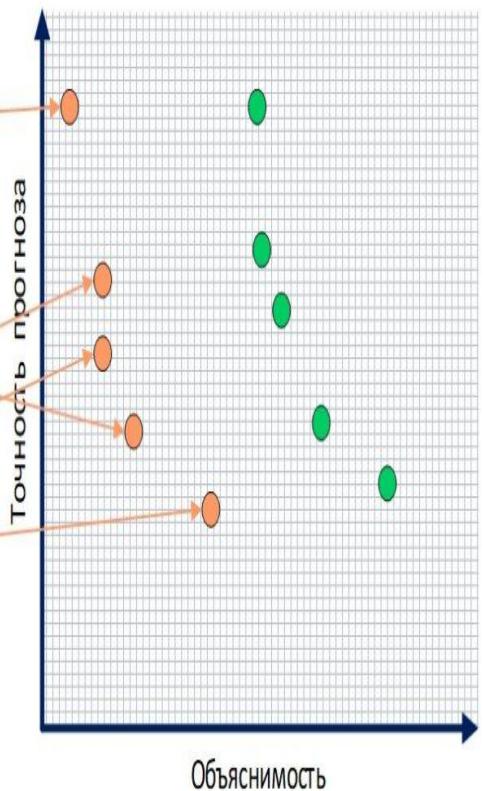
Объяснения алгоритмов работы ИИ могут быть представлены в текстовой или визуальной форме.

- **Текстовые объяснения** представляют собой метод создания символов, отображающих логику алгоритма посредством семантического отображения.
- Многие из методов визуализации сопровождаются методами уменьшения размерности для упрощения понимания работы модели человеком. Методы визуализации могут сочетаться с другими методами для улучшения их понимания и считаются наиболее подходящим способом представить сложные взаимодействия между переменными, участвующими в модели.

Текущие методы обучения ИИ



Ожидание по развитию
объяснимого интеллекта



Существует несколько различных подходов для решения проблемы объяснимости ИИ.

1. **Локальные объяснения** сегментируют пространство решений и дают объяснения менее сложным подпространствам решений, которые актуальны для всей модели. Эти объяснения могут быть сформированы с помощью методов, которые объясняют часть функционирования всей системы.
2. **Объяснения на примерах** предполагают извлечение репрезентативных примеров, которые улавливают внутренние отношения и корреляции, обнаруживаемые анализируемой моделью данных, и относятся к результату, сгенерированному определенной моделью.
3. **Объяснения посредством упрощения** в совокупности обозначают те методы, в которых целая новая система ИИ перестраивается на основе

обученной модели ИИ, которую необходимо объяснить. Новый ИИ обычно пытается оптимизировать свое сходство с изначальной моделью ИИ, уменьшая при этом ее сложность, но сохраняя тот же уровень производительности.

4. *Методы апостериорного объяснения релевантности признаков* проясняют внутреннее функционирование модели, вычисляя оценку релевантности для ее управляемых переменных. Эти оценки количественно определяют влияние (чувствительность) на выходные данные модели. Сравнение оценок различных переменных показывает вес, которую модель присваивает каждой из таких переменных при получении результатов.
5. *Отсутствие объяснимости может приводить к ситуациям, когда модель присваивает более высокий вес тем входным переменным, которые объективно не должны иметь такого веса.*
6. **Пример:** разбор решений модели, которая должна была классифицировать волков и хаски, показал, что она основывала свои выводы на факте наличия снега на фоне (рис. 10).

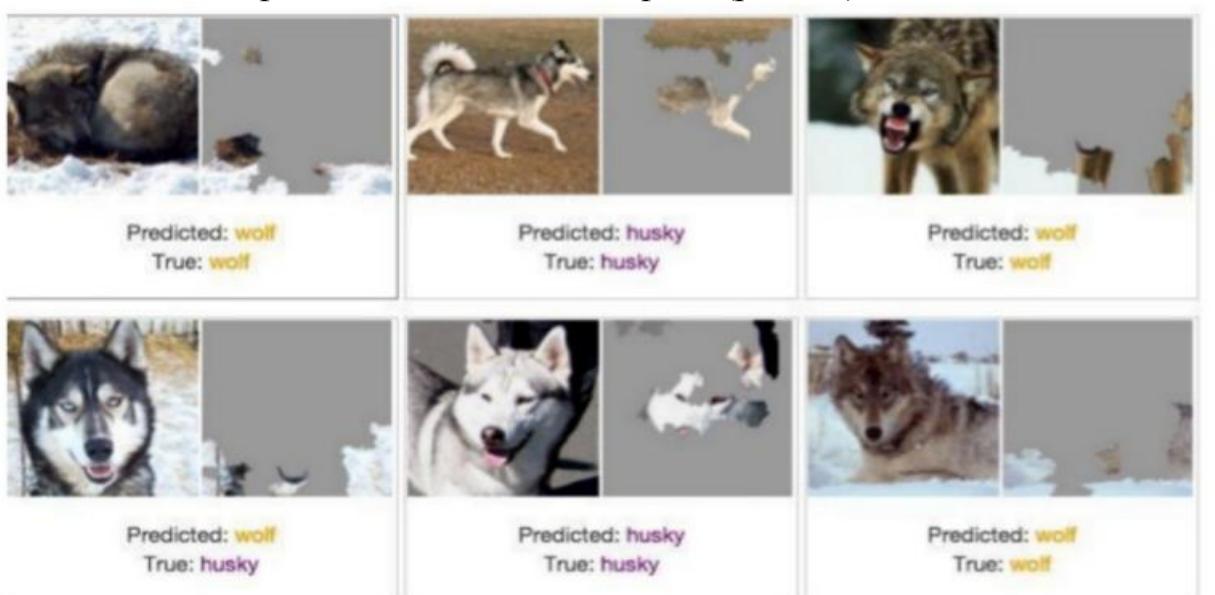
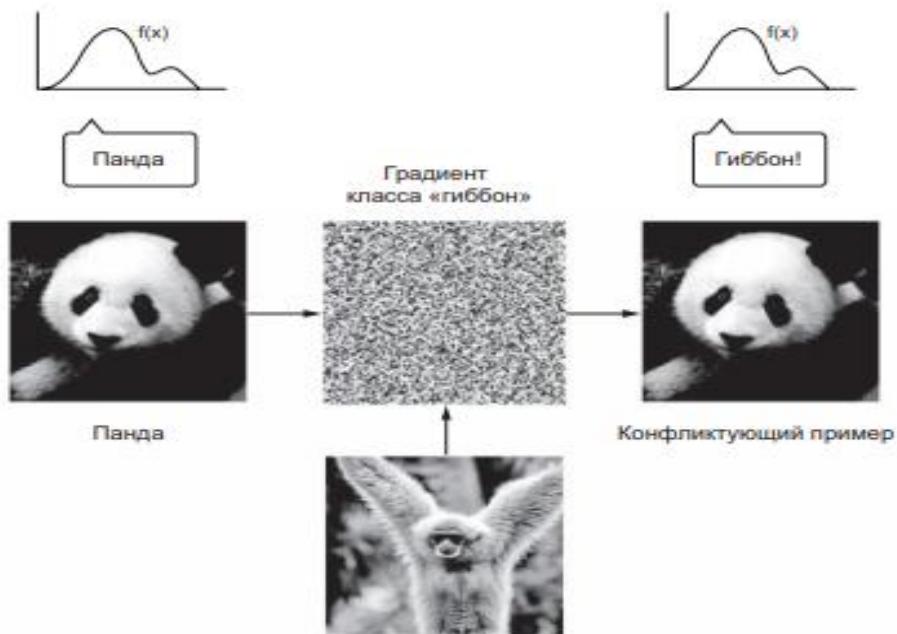


Рис. 10. Классификация волков и хаски на основе незначимого признака (наличия снега)

Злоумышленники могут добавлять небольшие искажения в изображения, которые не повлияют на решение человека, но могут сбить с толку алгоритм, ранее доказавший свою эффективность на не измененных искусственно изображениях (рис. 11).



Незаметные изменения в изображении могут мешать модели правильно его классифицировать

Рис. 11. Намеренное незначительное искажение изображения

Этические проблемы, связанные с работой моделей черного ящика, возникают из-за их склонности непреднамеренно принимать несправедливые решения с учетом чувствительных факторов, таких как раса, возраст или пол человека.

Пример: система оценки риска повторного совершения преступных деяний COMPAS принимала расу как одну из важных характеристик для принятия решения о высоком риске рецидива.

10. Интерпретируемые модели

Объяснимость модели ИИ может быть рассмотрена на всех этапах разработки ИИ, а именно: перед моделированием, в процессе разработки модели и после моделирования.

Самый простой способ добиться интерпретируемости – использовать только подмножество алгоритмов, создающих интерпретируемые модели ИИ. Линейная регрессия, логистическая регрессия и дерево решений являются обычно используемыми интерпретируемыми моделями ИИ.

10.1. Модель линейной регрессии

Модель линейной регрессии прогнозирует цель как взвешенную сумму входных параметров. Линейность установленных отношений облегчает интерпретацию. Модели линейной регрессии давно используются

статистиками, компьютерными специалистами и другими людьми, занимающимися количественными проблемами.

Линейные уравнения имеют легкую для понимания интерпретацию на модульном уровне (на уровне весов). Линейные модели широко распространены в медицине, социологии, психологии и др.

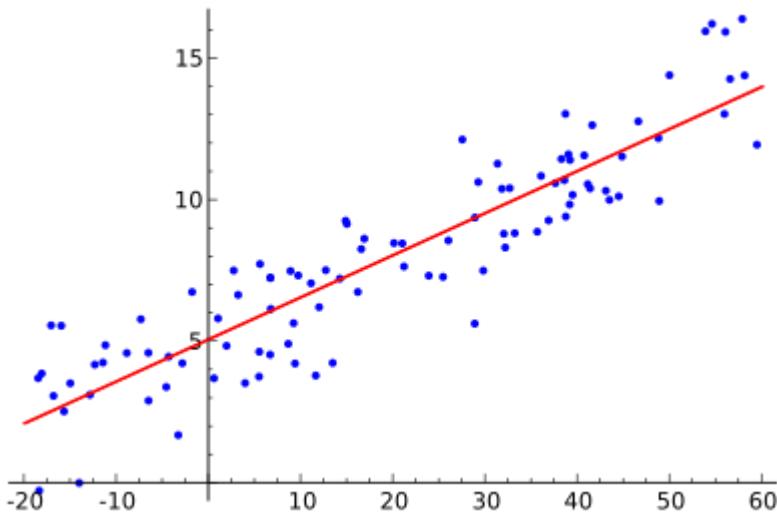


Рис. 12. Линейная регрессия

Отношения модели ИИ должны соответствовать определенным допущениям, а именно: линейность, нормальность, гомоскедастичность (имеют постоянную среднюю дисперсию), независимость, фиксированные характеристики и отсутствие мультиколлинеарности.

Однако линейные модели имеют следующие недостатки:

- ✚ Каждая нелинейность или взаимодействие должны создаваться вручную и явно передаваться модели в качестве входной характеристики;
- ✚ в части взаимосвязей линейные модели сильно ограничены и обычно сильно упрощают реальность, что сказывается на их возможностях прогнозирования;
- ✚ интерпретация весов может быть сложна для понимания, поскольку зависимость прослеживается на всей модели.
- ✚ Признак с высокой положительной корреляцией с результатом u и другой признак могут получить отрицательный вес в линейной модели, потому что, учитывая другой коррелированный признак, он отрицательно коррелирует с u в многомерном пространстве.
- ✚ Полностью коррелированные функции делают невозможным поиск однозначного решения линейного уравнения.

10.2. Логистическая регрессия

Логистическая регрессия моделирует вероятности проблем классификации с двумя возможными исходами и является расширением модели линейной регрессии для задач классификации. Интерпретация более сложна по сравнению с линейной регрессией, поскольку интерпретация весов является мультипликативной, а не аддитивной.

Логистическая регрессия может пострадать от полного разделения. Если есть функция, которая идеально разделяет два класса, модель логистической регрессии больше не может быть обучена, поскольку вес для этой функции не будет сходиться – оптимальный вес будет бесконечным.

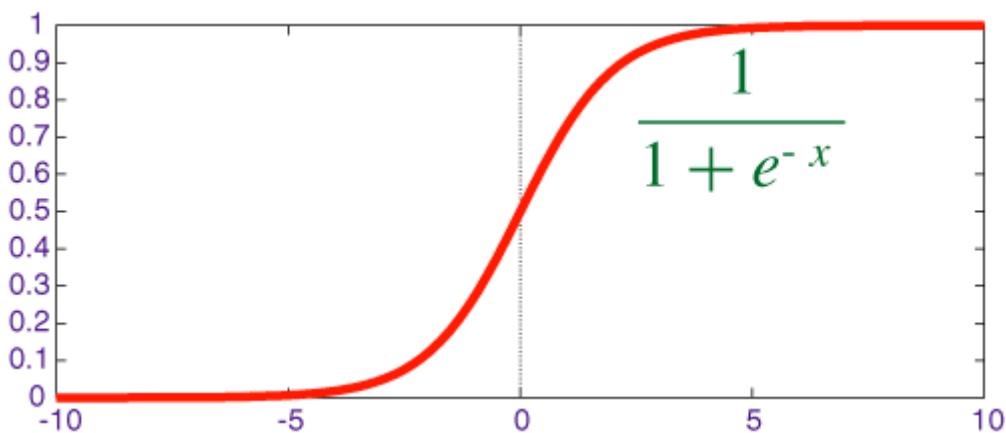


Рис. 13. Логистическая регрессия

Преимущество логистической регрессии в задачах классификации заключается в том, что она дает вероятности получения результатов для класса. Отсюда непосредственно вытекает ее практическое значение – это мощный статистический метод предсказания событий, которые являются результатом одной или нескольких независимых переменных.

Модель может быть расширена для предсказания нескольких классов, тогда она называется **полиномиальная регрессия**.

10.3. Обобщенные линейные модели (General linear model, GLM)

GLM расширяют возможности модели линейной регрессии. Эти модели дают возможность использовать переменные отклика с распределением ошибок, отличным от нормального.

- GLM – это общая математическая структура для выражения отношений между переменными, которые могут выражать или проверять линейные отношения между числовой зависимой переменной и любой комбинацией категориальных или непрерывных независимых переменных.
- Большинство методов машинного обучения с учителем каким-то образом расширяют GLM (через методы штрафов, метод ансамблей, объединение прогнозов из нескольких моделей машинного обучения в одном наборе данных и другие).

10.4. Обобщенная аддитивная модель (Generalized additive model, GAM)

GAM – это обобщенная линейная модель, в которой линейный предиктор линейно зависит от неизвестных гладких функций некоторых переменных-предикторов, и интерес сосредоточен на выводе об этих гладких функциях. В своей работе GAM использует сплайн-функции – функции, которые можно комбинировать для аппроксимации произвольных функций. GAM вводят штрафы для весов, чтобы они оставались близкими к нулю, что эффективно снижает гибкость сплайнов и снижает возможность переобучения. Параметр гладкости, который обычно используется для управления гибкостью кривой, затем настраивается с помощью перекрестной проверки.

Большинство модификаций линейной модели делают модель менее интерпретируемой. Любая функция связи, которая не является функцией идентичности, усложняет интерпретацию; взаимодействия также усложняют интерпретацию; эффекты нелинейных функций либо менее интуитивно понятны, либо больше не могут быть суммированы в одно число. GLM, GAM и другие расширения полагаются на предположения о процессе генерации данных. Если предположения не работают, интерпретация весов больше не действует.

Производительность древовидных ансамблей, таких как случайный лес или градиентное дерево, во многих случаях выше, чем у самых сложных линейных моделей.

10.5. Метод ансамблей базируется на алгоритмах машинного обучения, генерирующих множество классификаторов и разделяющих все объекты изновь поступающих данных на основе их усреднения или итогов голосования. Изначально метод ансамблей был частным случаем байесовского усреднения, но затем усложнился дополнительными алгоритмами:

- **бустинг (boosting)** – преобразует слабые модели в сильные посредством формирования ансамбля классификаторов (улучшающее пересечение);
- **бэггинг (bagging)** – собирает усложнённые классификаторы, при этом параллельно обучая базовые (улучшающее объединение);
- корректирование ошибок выходного кодирования.

Метод ансамблей – более мощный инструмент по сравнению с отдельно стоящими моделями прогнозирования:

- сводит к минимуму влияние случайностей, усредняя ошибки каждого базового классификатора;
- уменьшает дисперсию;
- исключает выход за рамки множества: если агрегированная гипотеза оказывается вне множества базовых гипотез, то на этапе формирования комбинированной гипотезы оно расширяется при помощи того или иного способа, и гипотеза уже входит в него.

10.6. Дерево принятия решений – это метод поддержки принятия решений, основанный на использовании древовидного графа: модели принятия решений, которая учитывает их потенциальные последствия (с расчётом вероятности наступления того или иного события), эффективность, ресурсозатратность.

Преимущества метода заключаются в том, что он структурирует и систематизирует проблему; итоговое решение принимается на основе логических выводов.

Деревья решений

Деревья решений – это способ представления правил в иерархической, последовательной структуре, где каждому объекту соответствует единственный узел, дающий решение

Деревья решений – это логический алгоритм классификации, основанный на поиске конъюнктивных закономерностей.



Рис. 14. Дерево решений

10.7. Наивные байесовские классификаторы относятся к семейству простых вероятностных классификаторов, основанных на теореме Байеса, которая рассматривает функции как независимые (это называется строгим, или наивным, предположением).

Используется в следующих областях машинного обучения:

- *определение спама, приходящего на электронную почту;*
- *автоматическая привязка новостных статей к тематическим рубрикам;*
- *выявление эмоциональной окраски текста;*
- *распознавание лиц и других паттернов на изображениях.*

10.8. Алгоритм RuleFit изучает разреженную линейную модель с исходными функциями, а также ряд новых функций, которые являются правилами принятия решений. Новые функции отражают взаимодействие между исходными функциями. RuleFit автоматически генерирует эти функции из деревьев решений. Каждый путь через дерево может быть

преобразован в правило принятия решения путем объединения разделенных решений в правило. Деревья решений в данном случае обучены предсказывать интересующий результат, что гарантирует необходимость разбиения для задачи прогнозирования. Для RuleFit можно использовать любой алгоритм, который генерирует множество деревьев решений, например, *случайный лес* (рис. 15).

Ансамбли. Случайный лес

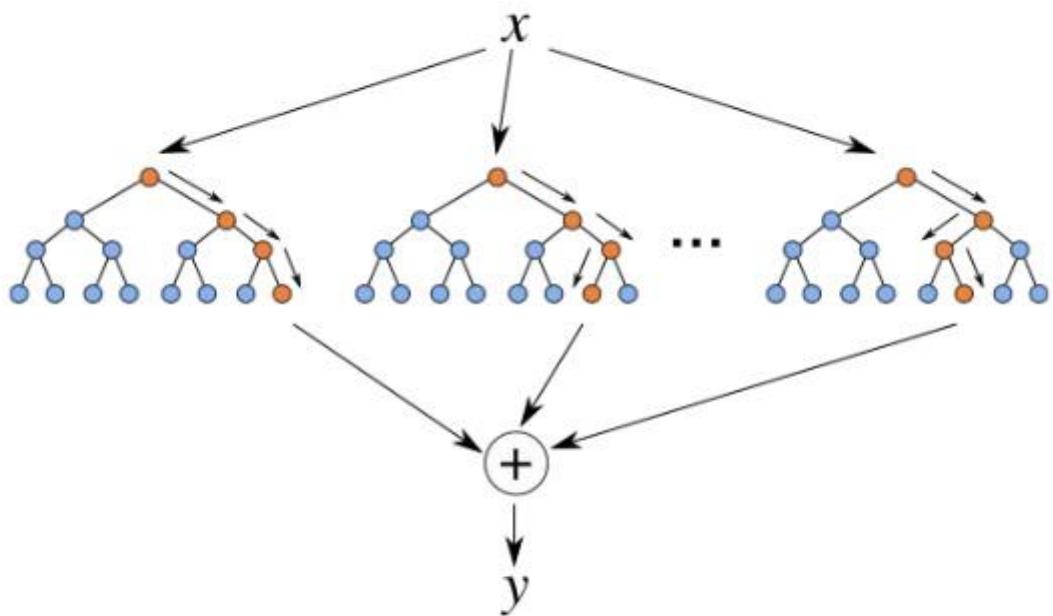


Рис. 15. Случайный лес

Каждое дерево разбивается на правила принятия решений, которые используются в качестве дополнительных функций в модели разреженной линейной регрессии. RuleFit имеет показатель важности функции, который помогает определить линейные условия и правила, которые важны для прогнозов. Важность функции рассчитывается на основе весов регрессионной модели. Показатель важности может быть агрегирован для исходных функций. RuleFit также представляет графики частичной зависимости, чтобы показать среднее изменение прогноза при изменении характеристики.

10.9. Метод k-ближайших соседей используется для задач классификации (иногда в задачах регрессии), объект относится к тому классу, которому принадлежит большинство из его соседей (рис. 16).

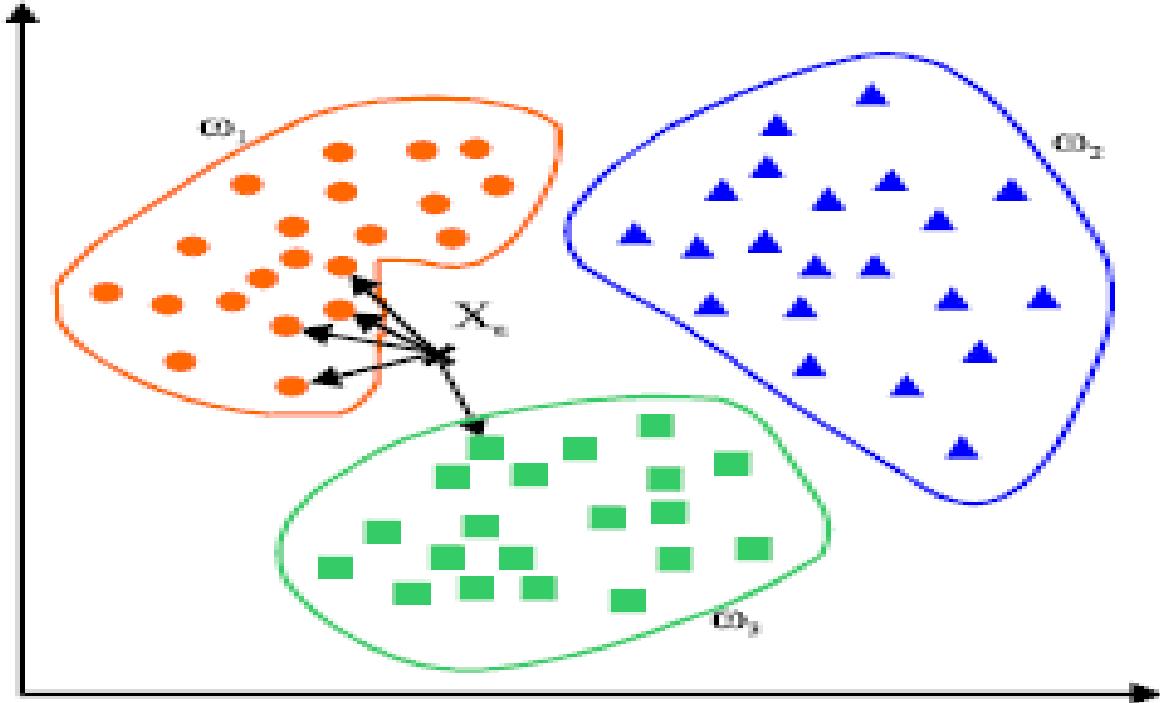


Рис. 16. К-ближайших соседей

В процессе обучения алгоритм просто запоминает все векторы признаков и соответствующие им метки классов. При работе с реальными данными, т.е. наблюдениями, метки класса которых неизвестны, вычисляется расстояние между вектором нового наблюдения и ранее запомненными. Затем выбирается k ближайших к нему векторов, и новый объект относится к классу, которому принадлежит большинство из них.

Увеличение значение параметра k повышает достоверность классификации, но при этом границы между классами становятся менее четкими. На практике хорошие результаты дают эвристические методы выбора параметра k , например, перекрестная проверка.

Несмотря на свою относительную алгоритмическую простоту, метод показывает хорошие результаты.

Главный недостаток: высокая вычислительная трудоемкость, которая увеличивается квадратично с ростом числа обучающих примеров.

10.10. Апостериорные методы объяснений

Когда модели машинного обучения не соответствуют ни одному из критериев, налагаемых для объявления их прозрачными, необходимо разработать и применить к модели отдельный метод для объяснения ее решений. Это цель методов апостериорной объяснимости, которые нацелены на передачу понятной информации о том, как уже разработанная модель производит свои прогнозы для любого заданного входа.

Интерпретация в данных методах основана на ограниченном доступе к внутренней работе модели. Например, объяснения могут быть получены из градиентов, «прогона» модели в обратном направлении или с помощью запросов путем оценки более простых суррогатных моделей для фиксации наблюдаемого локального поведения входа-выхода.

Глобальные апостериорные объяснения полезны для принимающих решения лиц, которых поддерживает модель машинного обучения. Врачи, судьи и кредитные специалисты получают общее представление о том, как работает модель, но обязательно существует разрыв между моделью черного ящика и объяснением. Местные апостериорные объяснения актуальны для частных лиц, таких как пациенты, ответчики, на которых влияет результат модели и которым необходимо понимать ее интерпретацию с их конкретной точки зрения. Большим преимуществом методов интерпретации, не зависящих от модели, по сравнению с методами интерпретации для конкретных моделей является их гибкость.

Желательными аспектами системы объяснения, не зависящей от модели, являются:

- **гибкость модели** (метод интерпретации может работать с любой моделью машинного обучения, такой как случайные леса и глубокие нейронные сети);
- **гибкость объяснений** (различные формы объяснений, например, линейная формула для одних случаев, график с важностью функций – для других);
- **гибкость представления** (система объяснения должна иметь возможность использовать другое представление объекта в качестве объясняемой модели).

10.11. Локально интерпретируемое не зависящее от модели объяснение (Local interpretable model-agnostic explanations, LIME)

LIME объясняет классификатор для конкретного единичного выхода функции и поэтому подходит для локального рассмотрения. Объясняет отдельные прогнозы нейронной сети, аппроксимируя ее локально с помощью интерпретируемых моделей, таких как линейные модели и мелкие деревья.

Для локального объяснения решения алгоритма для определенного входа, линейная модель обучается имитировать алгоритм только для небольшой области вокруг входа. Эта линейная модель по своей природе может быть интерпретирована и сообщает, как изменится выход при изменении какой-либо входной функции.

LIME работает с табличными данными, текстом и изображениями.

Объяснения, созданные с помощью локальных суррогатных моделей, могут использовать другие (интерпретируемые) функции, которым была обучена исходная модель. Эти интерпретируемые функции должны быть получены из экземпляров данных.

Классификатор текста может полагаться на встраивание абстрактных слов как на признаки, но объяснение может основываться на наличии или отсутствии слов в предложении.

Модель регрессии может полагаться на неинтерпретируемое преобразование некоторых атрибутов, но объяснения могут быть созданы с использованием исходных атрибутов.

Методы, не зависящие от модели, основанные на возмущениях (такие как LIME) более склонны к нестабильности, чем их аналоги, основанные на градиентах.

В случае повторного процесса выборки, могут измениться объяснения. Нестабильность в данном случае означает, что объяснениям трудно доверять, и нужно быть очень критичным.

Локальные суррогатные модели с LIME очень многообещающие. Но этот метод все еще находится в стадии разработки, и необходимо решить многие проблемы, прежде чем его можно будет безопасно применять.

10.12. SHAP (Аддитивное объяснение Шепли)

SHAP это метод объяснения индивидуальных прогнозов. SHAP основан на игре теоретически оптимальных значений Шепли. SHAP определяет

пределный вклад каждой функции в достижение выходного значения, начиная с базового значения.

Значения Шепли рассматривают все возможные прогнозы с использованием всех возможных комбинаций входных данных. Благодаря такому подходу SHAP может гарантировать согласованность и локальную точность.

SHAP хорошо работает для задач классификации и регрессии, но плохо применим для обучения с подкреплением.

SHAP имеет прочную теоретическую основу в теории игр. Прогноз справедливо распределен между значениями признаков. На выходе можно получить контрастные объяснения, которые сравнивают полученный прогноз со средним прогнозом.

Быстрое вычисление позволяет вычислить множество значений Шепли, необходимых для интерпретации глобальной модели. Методы глобальной интерпретации включают важность характеристик, зависимость характеристик, взаимодействия, кластеризацию и сводные графики. С SHAP глобальные интерпретации согласуются с локальными объяснениями, поскольку значения Шепли являются «атомарной единицей» глобальных интерпретаций.

Недостатки метода:

- Проблема медленных решений (методы SHAP требуют вычисления значений Шепли для множества значений) остается актуальной для глобальных объяснений модели.
- SHAP может игнорировать зависимость функций при замене значений признаков случайными значениями, что, в свою очередь, может привести к приданию слишком большого значения маловероятным точкам данных.

LIME и SHAP были признаны научным сообществом как самые многообещающие модели апостериорных объяснений, не зависящих от модели, однако группой ученых было проведено исследование, доказавшее, *что данные методы плохо определяют необъективность модели.*

В эксперименте со специально созданным классификатором, который был явно необъективен, использовали в качестве значимых признаков только данные о расе человека. Данные методы игнорировали необъективность, находя вполне безобидные объяснения для полученных в ходе работы модели выходных данных.

Объяснения этих классификаторов, сгенерированные с использованием готовых реализаций LIME и SHAP, не помечают какие-либо важные чувствительные атрибуты (например, расу) как важные особенности классификатора для любого из тестовых примеров, демонстрируя, что необъективные классификаторы успешно обманули эти методы объяснения.

10.13. Layer-wise relevance backpropagation (LRP)

LRP – это метод, который определяет важные пиксели путем выполнения обратного прохода в нейронной сети. Обратный проход – это консервативная процедура перераспределения релевантности, при которой нейроны, которые вносят наибольший вклад в более высокий уровень, получают от него наибольшую релевантность.

Метод может быть легко реализован на большинстве языков программирования и интегрирован в существующие нейронные сети. Правила распространения, используемые LRP, могут пониматься как глубокая декомпозиция Тейлора для многих архитектур, в том числе Deep Sparse Rectifier Neural Networks (сетей глубокого выпрямления) или LSTM.

LRP также отлично работает для сверточных нейронных сетей (CNN) и может использоваться для сетей долгой краткосрочной памяти (LSTM).

10.14. Метод интегрированных градиентов (Integrated Gradients, IG)

В методе **IG** градиент выходных данных прогноза вычисляется с учетом характеристик входных данных по интегральной траектории.

Целью метода является объяснение взаимосвязи между предсказаниями модели с точки зрения ее характеристик. Использование: понимание важности функций, определение перекоса данных и отладка производительности модели.

IG стал популярным методом интерпретируемости из-за его широкой применимости к любой дифференцируемой модели, простоты реализации, теоретического обоснования и вычислительной эффективности по сравнению с альтернативными подходами, что позволяет масштабировать его до крупных сетей и функций.

Данный метод не требует модификации исходной сети, прост в реализации и применим к множеству глубоких моделей (разреженных и плотных, текстовых и визуальных).

Интегрированные градиенты определяют важность функций на отдельных примерах, но не предоставляют визуализацию важности функций для всего

набора данных; определяют индивидуальные значения функций, но не объясняют взаимодействия и комбинации функций.

10.15. XRAI. На основе метода интегрированных градиентов метод XRAI оценивает перекрывающиеся области изображения для создания карты значимости, которая выделяет соответствующие области изображения, а не пиксели (рис. 17).

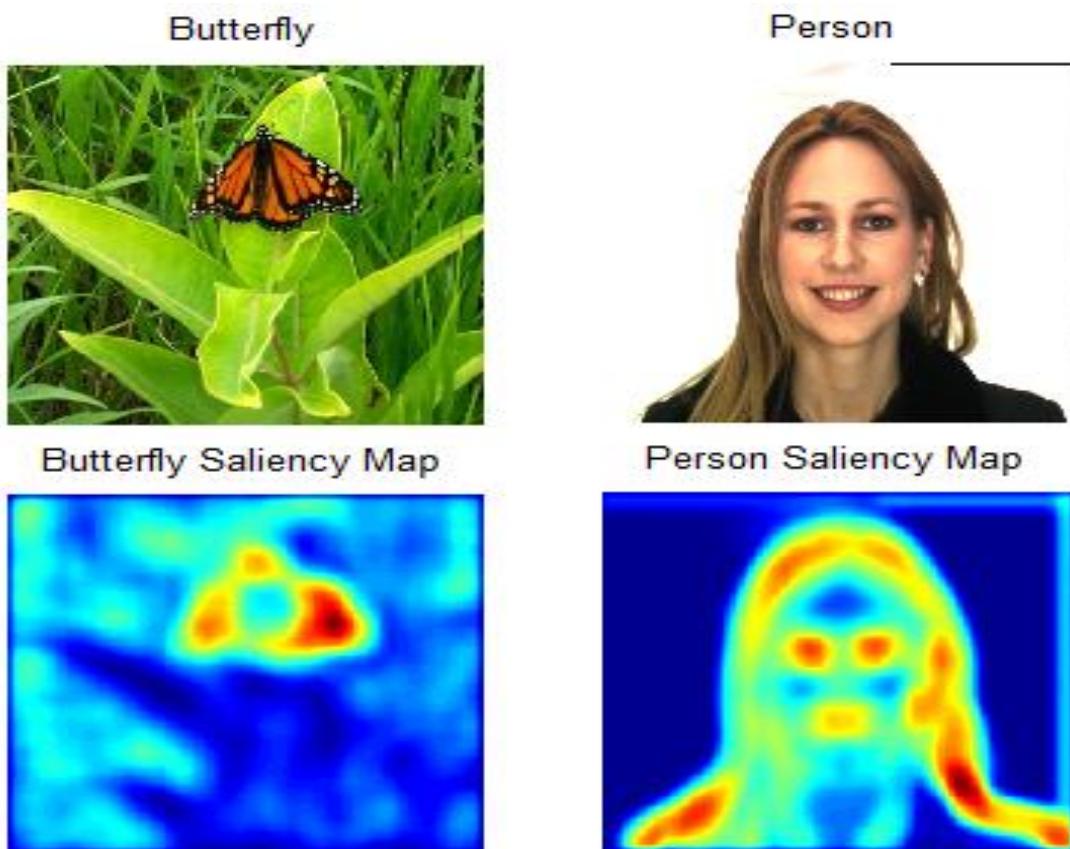


Рис. 17. Карта значимости признаков

Независимо от атрибуции на уровне пикселей, XRAI сверхсегментирует изображение, чтобы создать лоскутное одеяло из небольших областей. XRAI использует графовый метод для создания сегментов изображения. Затем происходит агрегация атрибуции на уровне пикселей в каждом сегменте, чтобы определить его плотность атрибуции. Используя эти значения, XRAI ранжирует каждый сегмент, а затем упорядочивает сегменты от наиболее положительного до наименее положительного. Это определяет, какие области изображения наиболее заметны или наиболее сильно влияют на предсказание класса.

Метод рекомендован для естественных изображений, которые представляют собой любые сцены реального мира, содержащие несколько объектов.

10.16. DeepLIFT – это подход, основанный на обратном распространении, который приписывает изменение входным данным на основе различий между входными данными и соответствующими ссылками (или базовыми показателями) для нелинейных активаций. Таким образом, DeepLIFT пытается объяснить разницу между исходными данными и эталонными данными с точки зрения разницы между входными данными и эталонными данными.

10.17. DeepLIFT SHAP – это метод, расширяющий DeepLIFT для аппроксимации значений SHAP. DeepLIFT SHAP берет распределение базовых показателей и вычисляет атрибуцию DeepLIFT для каждой пары исходных данных и базовых показателей и усредняет полученные атрибуции для каждого примера входных данных.

Правила DeepLIFT для нелинейностей служат для линеаризации нелинейных функций сети, метод аппроксимирует значения SHAP для линеаризованной версии сети. Метод также предполагает, что входные функции независимы.

11. Существующие решения для XAI

AIX360 от IBM — расширяемый набор инструментов, который предлагает ряд возможностей для улучшения объяснимости модели. Соответствующий пакет Python AI Explainability 360 включает в себя алгоритмы, охватывающие различные метрики объяснений наряду с посредническими метриками объяснимости, предоставляет инструменты для визуального исследования поведения обученных моделей с помощью минимального количества кода.

Набор инструментов включает в себя серию алгоритмов интерпретируемости, которые отражают современные исследования по этой теме, а также интуитивно понятный пользовательский интерфейс, который помогает понять модели машинного обучения с разных точек зрения. Один из основных вкладов AI Explainability 360 заключается в том, что он не полагается на единственную форму интерпретации модели машинного обучения. AI Explainability 360 дает разные объяснения для разных ролей, таких как специалисты по обработке данных или заинтересованные стороны в бизнесе. Объяснения, генерируемые AI Explainability 360, могут быть основаны на данных или на моделях.

Разработчики могут начать использовать AI Explainability 360, включив интерпретируемые компоненты с помощью API, включенных в набор инструментов. AI Explainability 360 включает в себя серию презентаций и

руководств, которые могут помочь разработчикам относительно быстро начать работу.

Интерпретируемость модели в **Microsoft Azure** обеспечивается пакетом SDK, используя который можно объяснять предсказания модели через генерацию важности значения функций для всей модели или отдельных точек данных и с помощью интерактивной визуализации для обнаружения связей в данных и объяснений во время обучения модели.

Azureml-interpret использует методы интерпретируемости, разработанные в Interpret-Community, пакете Python с открытым исходным кодом для обучения интерпретируемых моделей и помощи в объяснении систем искусственного интеллекта с «черным ящиком». Interpret-Community служит базой для поддерживаемых моделей объяснения этого SDK и в настоящее время поддерживает следующие методы интерпретируемости: вариации SHAP (TreeSHAP, SHAP deep Explainer, SHAP Kernel explainer и др.), Global Surrogate₁, Permutation Feature Importance Explainer₂. Для объяснения работы глубоких нейронных сетей можно использовать TabularExplainer, который использует методы SHAP.

Google Cloud's AI Explanations ставит целью использовать объяснения ИИ для упрощения разработки модели, а также объяснить поведение модели ключевым заинтересованным сторонам. AI Explanations работает с моделями, решающими задачи классификации и регрессии, демонстрируя, как та или иная функция данных повлияла на результат. В AI Explanations используются следующие методы: метод интегрированных градиентов, XRAI и Sampled Shapley. Для визуализаций используется What-If Tool.

В 2018 году КомандаPAIR (People + AI Reserach, часть Google AI) представила **What-If Tool** — инструмент для обнаружения предвзятости в моделях искусственного интеллекта. Он поставляется как часть веб-приложения TensorBoard. What-If Tool визуализирует влияние определенных данных на предсказание модели, при этом доступен для тех, кто не разбирается в программировании.

What-If Tool позволяет:

- Автоматически визуализировать набор данных с использованием Facets. Facets Overview дает возможность получить представление о форме каждой функции набора данных; Facets Dive с помощью визуализации позволяет изучить отдельные наблюдения.
- Редактировать отдельные примеры из набора и отслеживать, как это отражается на результатах.

- Автоматически создавать графики частичных зависимостей, отражающие перемены в предсказании модели при изменении какого-либо одного свойства.
- Сравнивать наиболее похожие примеры из датасета, для которых модель дала разные предсказания.

Проверить работу инструмента можно на заранее обученных алгоритмах: модели многоклассовой классификации; модели классификации изображений; регрессионной модели. What-if-tool может работать с текстовыми, визуальным и табличными данными.

Платформа **Thales XAI** обеспечивает различные уровни объяснения, например, на основе примеров, на основе функций, контрафактов с использованием текстовых и визуальных представлений, однако стоит отметить, что упор делается на объяснение на основе семантики с помощью графов знаний. Графы знаний используются для кодирования лучшего представления данных, структурирования модели машинного обучения более интерпретируемым образом и принятия семантического сходства для локального (на основе экземпляров) и глобального (на основе модели) объяснения.

Объяснение любого предсказания связи получается путем определения репрезентативных горячих точек в графе знаний, то есть связанных частей графиков, которые при удалении отрицательно влияют на точность прогнозирования.

ELI5 – это пакет Python, который помогает отлаживать классификаторы машинного обучения и объяснять их прогнозы. Он обеспечивает поддержку нескольких платформ и пакетов машинного обучения:

scikit-learn. В настоящее время ELI5 позволяет объяснять веса и прогнозы линейных классификаторов и регрессоров scikit-learn, печатать деревья решений в виде текста или графическом формате SVG, показывать важность функций и объяснять прогнозы деревьев решений и ансамблей на основе деревьев.

Также поддерживаются Pipeline и FeatureUnion.

Функции некоторых инструментов в Eli5:

XGBoost – показать важность функций и объяснить прогнозы XGBClassifier, XGBRegressor и xgboost.Booster.

LightGBM – показать важность функций и объяснить прогнозы LGBMClassifier и LGBMRegressor.

CatBoost – показать важность функций CatBoostClassifier и CatBoostRegressor.

Keras – объяснить предсказания классификаторов изображений с помощью визуализаций GradCAM.

ELI5 также реализует несколько алгоритмов проверки моделей черного ящика:

TextExplainer позволяет объяснить предсказания любого текстового классификатора с использованием алгоритма LIME. Существуют также утилиты для использования LIME с нетекстовыми данными и произвольными классификаторами черного ящика, но эта функция в настоящее время является экспериментальной.

Метод перераспределения важности можно использовать для вычисления значимости признаков для оценщиков черного ящика.

Есть возможность получить текстовое объяснение для отображения на консоли, HTML-версию, встраиваемую в блокнот IPython или на веб-панели, версию JSON, которая позволяет реализовать настраиваемый рендеринг и форматирование на клиенте, а также преобразовать объяснения в объекты DataFrame pandas.

Пакеты с открытым исходным кодом значительно улучшили воспроизводимость исследований и внесли значительный вклад в недавние исследования в области глубокого обучения и XAI.

Некоторые программные пакеты XAI, доступные в GitHub:

Interpret от InterpretML может использоваться для объяснения моделей черного ящика и в настоящее время поддерживает объяснимый бустинг, деревья решений, список правил принятия решений, линейную логистическую регрессию, SHAP kernel explainer, TreeSHAP, LIME, анализ чувствительности Морриса и частичную зависимость.

Пакет **IML** охватывает такие методы как важность функций, графики частичной зависимости, графики индивидуальных условных ожиданий, накопленные локальные эффекты, суррогаты деревьев, LIME и SHAP.

Пакет **DeepExplain** включает различные методы на основе градиента, такие как карты значимости, интегрированные градиенты, DeepLIFT, LRP и др., а также методы на основе возмущений, такие как окклюзия, SHAP и др.

12. Проблемы, связанные с развитием и использованием XAI

- В научных исследованиях, посвященных объяснимому искусственному интеллекту, одной из главных проблем развития XAI считается тот факт, что создание более понятной модели машинного обучения может в конечном итоге ухудшить качество принимаемых ею решений.
- Не всегда верно, что более сложные модели по своей сути более точны. Это утверждение неверно в тех случаях, когда данные хорошо структурированы, а функции, имеющиеся в распоряжении разработчика, имеют высокое качество и ценность. Этот случай распространен в некоторых отраслевых средах, поскольку анализируемые функции ограничиваются контролируемыми физическими проблемами, в которых все функции сильно коррелированы, и в данных можно исследовать не так уж много возможного ландшафта значений.
- Более сложные модели обладают гораздо большей гибкостью, чем их более простые аналоги, что позволяет аппроксимировать более сложные функции.
- Дilemma аппроксимации тесно связана с интерпретируемостью: объяснения, сделанные для модели машинного обучения, должны соответствовать требованиям аудитории, для которой они формируются, обеспечивая репрезентативность исследуемой модели без излишнего упрощения ее основных черт.

В науке в отношении XAI еще не существует единых объективных показателей того, что конкретно является хорошим объяснением. Для решения данной проблемы исследователи предлагают провести ряд экспериментов в области человеческой психологии, социологии или когнитивных наук для создания объективно убедительных объяснений.

Объяснения более убедительны, когда они являются ограничительными, что означает, что предварительным условием для хорошего объяснения является то, что оно не только указывает, почему модель приняла решение X, но и почему она приняла решение X, а не решение Y. Для объяснений более важны причинно-следственные связи, чем определение вероятности.

Необходимо преобразовать вероятностные результаты в качественные понятия, содержащие причинно-следственные связи. Возможно, достаточно сосредоточиться исключительно на основных причинах процесса принятия решений. В ряде исследований было доказано, что использование контрафактических объяснений может помочь пользователю понять решение модели.

Хотя методы объяснимости все чаще используются в качестве проверок работоспособности в процессе разработки, все еще существуют значительные ограничения на существующие методы, которые не позволяют их использовать для непосредственного информирования конечных пользователей. Эти ограничения включают в себя необходимость оценки объяснений экспертами в предметной области, риск ложных корреляций, отраженных в объяснениях модели, отсутствие причинно-следственной интуиции и задержку при вычислении и отображении объяснений в реальном времени. В будущих исследованиях следует стремиться устранить эти ограничения.



Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н. Э. Баумана)

Цикл лекций по дисциплине **«Методология программной инженерии»**

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук.
rtn@bmstu.ru

Москва - 2024



*Лекция 15

Искусственный интеллект

1. Мультиагентные системы (МАС). Структура, принципы построения и применение МАС.
2. Искусственные иммунные системы (ИИС)

- › Цель мультиагентных систем (МАС) - координация независимых процессов.
- › Агент представляет собой компьютерную сущность в виде программы или робота.
- › Агента можно считать автономным, поскольку он способен адаптироваться при изменении своей среды.
- › МАС состоит из набора компьютерных процессов, которые происходят одновременно и существуют в одно и то же время, обмениваются общими ресурсами и общаются друг с другом.
- › Ключевой проблемой в МАС является формализация координации между агентами.

МАС представляет собой перспективный подход к разработке ПО для приложений в сложных областях:

- › где взаимодействующие компоненты приложения автономны и распределены;
- › в динамичных и неопределенных средах;
- › где должны соблюдаться некоторые организационные правила и законы;
- › где агенты могут присоединиться и покинуть мультиагентную систему во время выполнения.

Примерами таких приложений являются:

- * системы, которые управляют производством и оптимизируют его;
- * распределяют электроэнергию между потребителями;
- * оптимально планируют грузы в транспортных системах.

Разработка мультиагентных систем требует создание отдельных агентов, организаций и сред.

*Мультиагентные организации с точки зрения социальных и организационных концепций обладают ролями, наделены нормами, протоколами связи, ресурсы которых подвержены мониторингу.

* Разработанные языки программирования и фреймворки применяются для создания симуляций на основе агентов для многих отраслей непрерывного производства: электроэнергия, металлургия, здравоохранение, интернет, транспорт, управления трафиком и создания серьезных игр.

* МАС отличаются от одноагентных систем тем, что в них существует несколько агентов, которые моделируют цели и действия друг друга. В общем сценарии может быть прямое взаимодействие между агентами.

* С точки зрения отдельного агента МАК отличаются от систем с одним агентом тем, что динамика окружающей среды может быть определена другими агентами.

* В дополнение к неопределенности, которая может быть присуща домену, другие агенты намеренно воздействуют на окружающую среду непредсказуемыми способами.

*Классификация агентов установлена по двум критериям: *когнитивные агенты или реагенты.*

**Когнитивные агенты* проявляют «умное» поведение при взаимодействии с окружающей средой.

Реагенты (реактивные агенты)

ограничиваются реакциями на воздействие внешней среды.

Современные МАС имеют динамические среды, могут иметь любое количество агентов с возможностью прямого общения или без него и с разной степенью неоднородности.

Это познавательное и реактивное поведение соответствует двум теоретическим школам многоагентных систем.

1. Первая поддерживает подход «умных» агентов для сотрудничества, то есть стремление агентов к явным целям.
2. Во второй изучается возможность возникновения «умного» поведения набора *неинтеллектуальных агентов*, то есть поведение агентов, исходя из ситуаций окружающей среды.

Таблица, группирующая различные типы агентов:

1. Когнитивные агенты.
2. Агенты "модули".
3. Реактивные агенты:
- 4.(рефлексивные и тропические агенты).
5. Агенты с телеономическим поведением.
6. Интеллектуальные агенты.

1. Когнитивные агенты в основном преднамеренные, то есть у них есть фиксированные цели, которые они пытаются выполнить.

2. Модулями называют агентов, которые имеют представление о своей «вселенной» без конкретных целей. Они могли бы служить, например, для ответа на вопросы других агентов во «вселенной».

3. Реагентов можно разделить на рефлексных и тропических агентов.

3.1. У инстинктивного агента будет фиксированная миссия, то есть на определенное воздействие вызовет фиксированное поведение.

3.2. **Тропический агент** реагирует только на местное состояние окружающей среды, например если свет, то он бежит.

4. Телеономические агенты работают в рамках заранее определенных законов или правил.

Телеономия - это качество кажущейся целенаправленности структур и функций в живых организмах, обусловленное естественными процессами, такими как естественный отбор.

Термин происходит от греческого "τελεονομία". Это соединения двух греческих слов: τέλος (конец, цель) и νόμος (номос-закон).

5. Интеллектуальный агент (ИА)

способен принимать решения на основе своего опыта и может выбирать действия в различной ситуации.

ИА способен к действиям на основе информации, которую он сам воспринимает, на основе своих собственных переживаний, решений и действий.

Платформы для разработки нескольких агентов

AnyLogic - многоагентное и многокомпонентное ПО для моделирования с открытым исходным кодом, основанная на объектно-ориентированном языке программирования SmallTalk.

DomIS - инструмент для проектирования многоагентных систем, ориентированный на «оперативный контроль сложных систем».

JACK - язык программирования и среда разработки для когнитивных агентов, разработанная в качестве ориентированного на агентов расширения языка Java.

GAMA - платформа моделирования с открытым исходным кодом (LGPL), предлагающая пространственно явную среду моделирования на основе агентов с использованием данных ГИС для описания агентов и их среды.

JADE (Java Agent DEVELOPMENT) является основой разработки многоагентных систем с открытым исходным кодом, и на основе языка Java.

Семь моделей стандарта

Методологический стандарт позволяет понятным и простым способом, создать МАС, не только используя естественный язык, но и используя шаблоны описания, которые помогают в спецификации системы.

1. Сценарная модель, описывающая компанию или организацию.

2. Модель целей и задач определяет и описывает органическую структуру.

3. Модель агента определяет людей и автономные системы.

4. Организационная модель описывает среду, с которой связан отдельный агент.

5. Ролевая модель связывает цели и задачи с определенным агентом.

6. Модель взаимодействия описывает связь, подчеркивая координацию между агентами.

7. Модель проектирования определяет агента и сетевую архитектуру

Примеры взаимодействия между агентами:

1. *MAC в социологии* позволяет параметризовать различных агентов, составляющих сообщество. Добавляя ограничения, можно попытаться понять, какой будет наиболее эффективный компонент для достижения ожидаемого результата.

Они должны экспериментировать со сценариями, которые могут быть недостижимы реальными людьми, либо по техническим или этическим причинам.

2. Распределенный искусственный агент (РИА)

Решает проблемы сложности больших монолитных программ неприродного интеллекта - выполнение, распределение и централизованный контроль.

Для решения сложной проблемы иногда легче создавать относительно небольшие программы (агенты) во взаимодействии, чем одна большая монолитная программа.

Автономия позволяет системе динамически адаптироваться к непредвиденным изменениям в окружающей среде.

Преимущества подхода с несколькими агентами MAS:

- Распределяет вычислительные ресурсы и возможности в сети взаимосвязанных агентов.
- Позволяет осуществлять соединение и взаимодействие нескольких существующих унаследованных систем.
- Обслуживание воздушных судов.
- Военное разминирование территорий.
- Беспроводное взаимодействие и связь.

- Планирование военной логистики.
- Система управления цепями поставок.
- Совместное планирование миссий.
- Финансовое управление портфолио.
- Электронные кошельки на покупку книг.

Таким образом, MAC - это слабосвязанная сеть программных агентов, которые взаимодействуют для решения проблем, выходящих за рамки индивидуальных возможностей или знаний каждого агента.

В МАС агенты имеют несколько важных характеристик:

- **Автономность:** агенты, хотя бы частично, независимы.
- **Ограниченностъ представления:** ни у одного из агентов нет представления обо всей системе, или система слишком сложна, чтобы знание о ней имело практическое применение для агента.
- **Децентрализация:** нет агентов, управляющих всей системой.

Обычно в МАС исследуются программные агенты, но могут также быть роботы, люди или команды людей и смешанные команды. В МАС может проявляться самоорганизация и сложное поведение, даже если стратегия поведения каждого агента достаточно проста.
(Модель так называемого роевого интеллекта).

Агенты могут обмениваться полученными знаниями, используя некоторый специальный язык и подчиняясь установленным правилам «общения» (протоколам) в системе.

Примерами таких языков являются Knowledge Query Manipulation Language (KQML) и FIPA's Agent Communication Language (ACL).

Примеры мультиагентных систем в игровой индустрии многочисленны и разнообразны.

Они используются в видеоиграх и в фильмах, например, чтобы симулировать движение толпы. Они также могут использоваться компаниями, например, для отслеживания поведения клиентов, просматривающих веб-сайты.

МАС также используются в мире финансов.

Например, платформа MetaTrader 4 позволяет использовать экспертные агенты в автоматической торговле, которые следуют курсам Forex

Свойства МАС

МАС также относятся к самоорганизующимся системам, так как в них ищется оптимальное решение задачи без внешнего вмешательства. Под оптимальным решением понимается решение, на которое потрачено наименьшее количество энергии в условиях ограниченных ресурсов.

- *Главное достоинство МАС – это гибкость.* Многоагентная система может быть дополнена и модифицирована без переписывания значительной части программы.
- Также эти системы обладают способностью к самовосстановлению и обладают устойчивостью к сбоям, благодаря достаточному запасу компонентов и самоорганизации.

Роевой интеллект

Люди давно стали интересоваться так называемым “роевым поведением”:

- каким образом птицы летят на юг огромными косяками, не сбиваясь с курса;
- как огромные колонии муравьёв работают так слаженно и возводят сложные структуры;
- как пчёлы могут так точно определять и добывать в необходимое для всей колонии питание.

Все эти большие группы животных/насекомых можно объединить одним общим словом — рой.

Инженеры стали моделировать “роевой интеллект” (РИ) — попытки сделать роботизированные, автоматические и автоматизированные рои, заложив к его основанию некоторые фундаментальные правила.

Такой термин как “Роевой интеллект” был введён Ван Цином и Херардо Бени в 1989 году. Такая модель подразумевает наличие так называемой “многоагентной системы”, которая определяется как система, состоящая из множества интеллектуальных агентов, способных самостоятельно на протяжении некоторого, достаточно длительного промежутка времени, выполнять поставленную задачу.

Искусственная иммунная система (ИИС)

Это такая вычислительная система, которая способна адаптироваться и использовать схожие с реальной иммунной системой принципы.

У данного метода есть три основные теории, которые описывают его функционирование и взаимодействие между элементами:

- теория отрицательного отбора;
- теория иммунной *сети*(*модели, основанные на принципах функционирования иммунной системы*).
- теория клonalной селекции.

На основе теорий функционирования ИИС было создано несколько классов алгоритмов, которые успешно решают эти задачи с помощью нейронных сетей и машинного обучения.

Например, искусственные иммунные сети используются для решения задач визуализации данных и кластеризации.

В целом ИИС можно использовать для задач оптимизации, классификации, моделирования системы поиска и распознавания образов (аномалий), а также в области информационной безопасности.

теория отрицательного отбора

В таких системах чаще всего функционирование базируется на двух «столпах»: *антиген и антитело*.

Антигенами в этом случае будут являться сетевые пакеты или системные вызовы.

Антитела будут вырабатываться ИИС в качестве реакции на специфические антигены.

В зависимости от типа антигена, антитела могут быть *пропускающими, блокирующими или уничтожающими*.

Соответственно, пакет, поступивший на устройство, может быть признан ИИС вредоносным, в таком случае он будет удален, а прием подобных пакетов – заблокирован, или же наоборот – безопасным, и тогда пакет будет спокойно пропущен в систему.

Теория иммунной сети

- Ерне предложил гипотезу, согласно которой иммунная система представляет собой регулируемую сеть молекул и клеток, распознающих друг друга даже при отсутствии антигена.
- Такие структуры часто называют *идиотипическим сетями*, они служат математической основой для изучения поведения иммунной системы.
- Теория Ерне интерпретируется в виде системы дифференциальных уравнений, описывающей динамику концентрации клонов лимфоцитов и соответствующих молекул иммуноглобулинов.

Клонально-селективная теория.

Теория разработана Фрэнком Бёрнетом (1899—1985) для объяснения функционирования иммунной системы.

Согласно этой теории в организме возникают клоны клеток, иммунокомпетентных в отношении различных антигенов.

Антиген избирательно контактирует с соответствующим клоном, стимулируя выработку им антител.

Другие менее распространенные алгоритмы:

- **метод капель воды**, который находит наиболее оптимальные “пути для воды”, подобно рекам;
- **алгоритм кукушки** — основан на принципе паразитирования, подобно тому, как некоторые виды кукушек откладывали яйца в чужие гнёзда, со временем научившись имитировать цвета чужих яиц;
- **метод альтруизма**, основанный на том, что каждый агент “заботится” об окружающих, не обращая внимания на себя;
- **метод гравитационного поиска** — заключён в соблюдении закона всемирного тяготения (все тела притягиваются друг к другу), а именно в поиске наиболее качественных, “тяжёлых”, агентов.

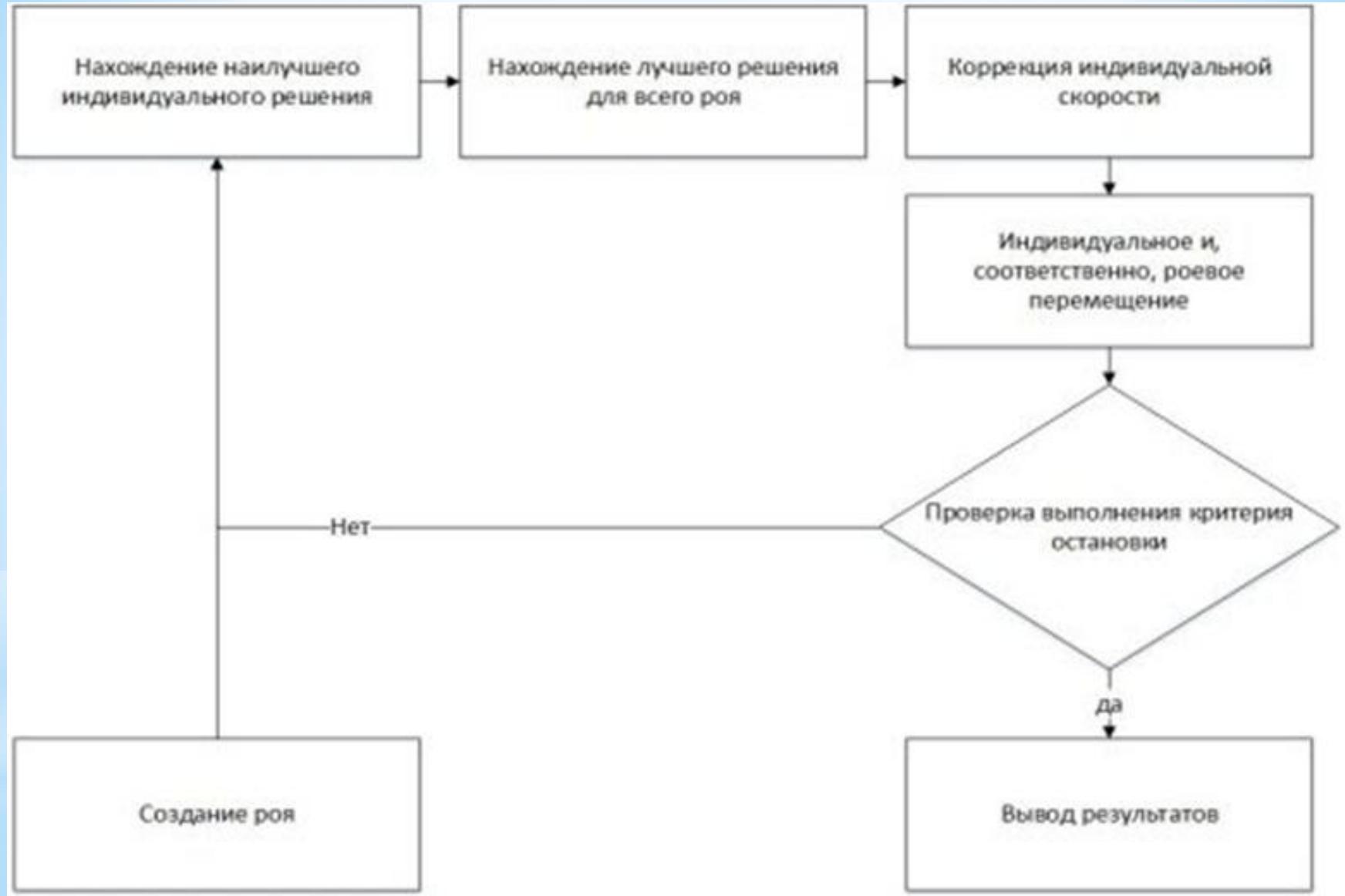
Метод роя частиц (МРЧ)

Данный метод является методом *численной оптимизации*. Он поддерживает общее количество возможных решений, которые называются *частицами или агентами*, и перемещает их в пространстве к наилучшему найденному в этом пространстве решению. Это оптимальное решение всё время находится в изменении из-за нахождения агентами более выгодных решений. Самая первая компьютерная модель МРЧ была придумана ещё в далёком 1986 Крейгом Рейнольдсом. Он придумал довольно простые правила поведения для частиц роя, действуя по которым, рой выглядел крайне похожим на реальный аналог птичьего роя. Но классическая модель МРЧ была создана лишь в 1995 году Расселом Эберхартом и Джеймсом Кеннеди.

Модель Эберхарта и Кеннеди отличается тем, что частицы-агенты роя, помимо подчинения неким правилам обмениваются информацией друг с другом, а текущее состояние каждой частицы характеризуется местоположением частицы в пространстве решений и скоростью перемещения.

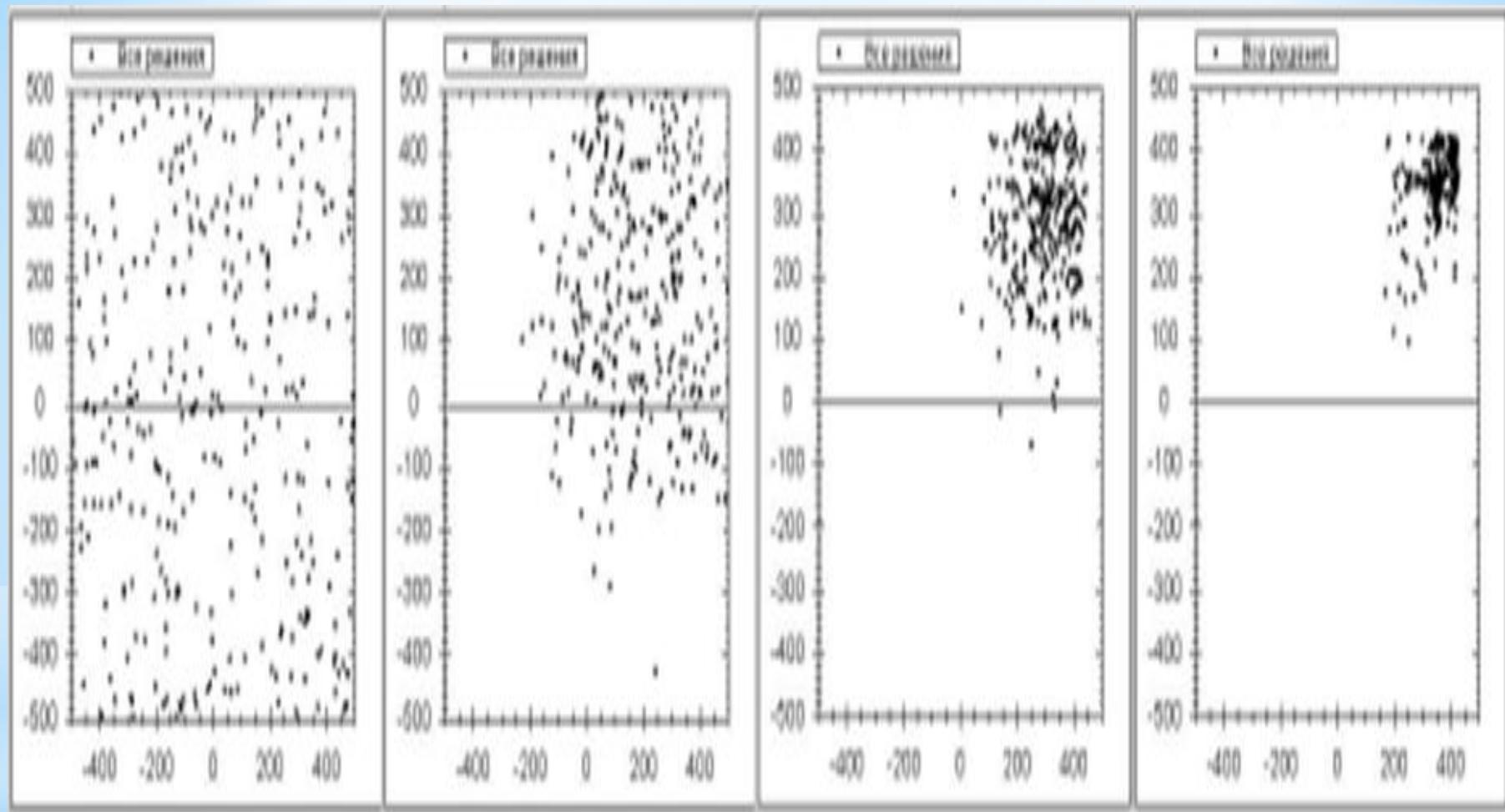
Если проводить аналогию со стаей, то можно сказать, что все агенты алгоритма (частицы), в стае они могут быть птицами или рыбами, ставят для себя три довольно простых задачи:

- *Все агенты должны избегать пересечения с окружающими их агентами;*
- *Каждая частица должна корректировать свою скорость в соответствии со скоростями окружающих её частиц;*
- *Каждый агент должен стараться сохранять достаточно малое расстояние между собой и окружающими его агентами.*



Алгоритм роя частиц — итеративный процесс, постоянно находящийся в изменении.

- Рассматривается область поиска в виде многомерного пространства с агентами нашего алгоритма.
- Изначально все агенты находятся в случайных местах пространства и со случайным вектором скорости.
- В каждой из точек, которую частица посещает, она рассчитывает заданную функцию и фиксирует наилучшее значение искомой функции.
- Все частицы знают местоположение наилучшего результата поиска во всём рое и с каждой итерацией агенты корректируют вектора своих скоростей и их направления, стараясь приблизиться к наилучшей точке роя и при этом быть поближе к своему индивидуальному максимуму.
При этом постоянно происходит расчёт искомой функции и поиск наилучшего значения.



Пример работы роя методом МРЧ

Концепцию данного алгоритма описывает формула, согласно которой корректируется модуль и направление скорости агентов:

$$v \leftarrow v + \text{rnd}()(\text{Pbest}-x)c_1 + \text{rnd}()(\text{gbest}-x)c_2,$$

где: ω — коэффициент инерции, определяющий баланс между тем, насколько широко будет “заходить” в исследовании агент и тем, насколько сильно агент будет желать остаться рядом с найденными ранее оптимальными решениями;

Pbest — координаты наилучшей найденной агентом точкой;

gbest — координаты наилучшей роевой точки;

x — текущие координаты точки;

$\text{rnd}()$ — случайный коэффициент, принимающий значение от 0 до 1; c_1, c_2 — постоянные ускорения.

Изначально этот алгоритм применялся для исследований социального психолога.

Однако самое большое распространение этот алгоритм получил при решении задач оптимизации различных многомерных нелинейных уравнений.

Этот алгоритм в современном мире применяется в машинном обучении, для решений задач оптимизации в биоинженерии и в других экспериментальных науках.

Муравьиный алгоритм

- Оптимизационный алгоритм с подражанием колонии муравьев один из самых эффективных алгоритмов для решения задач по поиску маршрутов в графах и по нахождению приблизительных решений для задачи коммивояжёра.
- Суть алгоритма заключается в применении модели функционирования колонии муравьев к решению различных задач.
- В этом алгоритме муравьиная колония рассматривается как мультиагентная система, в которой все агенты действуют самостоятельно по очень простым алгоритмам, но вся система в целом ведет себя крайне разумно.
- Поведение колонии муравьев основывается на самоорганизации, достигаемой за счет взаимодействия агентов на низком уровне ради общей цели.

Особи могут взаимодействовать как с помощью прямого обмена информацией (химический, визуальный контакт), так и с помощью непрямого обмена (стигмержи).

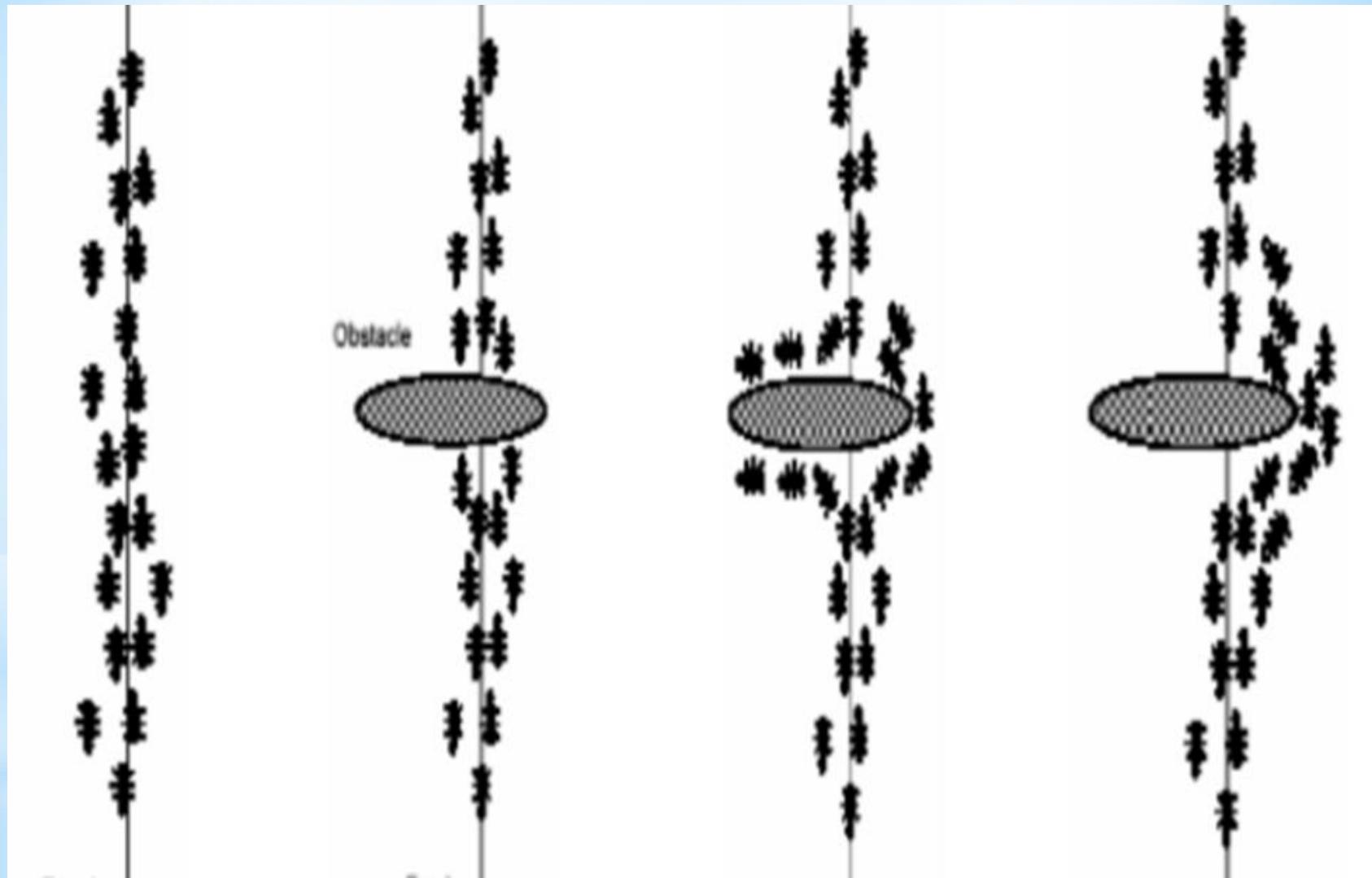
Он заключается в том, что некий агент может изменять область пространства с помощью некоторого вещества (феромона), после чего другие агенты могут использовать эту информацию для определения собственного маршрута. В результате концентрация феромонов на маршруте определяет приоритет его выбора.

Кроме того, «феромон» может испаряться, что создает динамичность алгоритму.

Первым, кто сумел применить поведение муравьев для решения задачи о кратчайших путях, стал Марко Дориго в начале 90-х годов XX века. В настоящее время эти алгоритмы показывают лучшие результаты в некоторых задачах.

Концепция алгоритма заключается в способности муравьев находить кратчайший путь крайне быстро и адаптироваться к различным внешним условиям.

При движении каждый муравей помечает свой путь феромоном, что в дальнейшем используется другими муравьями. Это и есть простой алгоритм одного агента, который в сумме всех агентов колонии позволяет находить кратчайший путь или изменять его при обнаружении препятствия.



**Пример нахождения муравьями нового пути
при появлении препятствия**

Муравьиный алгоритм представим в виде следующего набора команд:

Пока (не выполнены условия выхода):

1. Создание агентов;
2. Поиск подходящего решения;
3. Изменение феромона;
4. Вспомогательные действия (не обязательно).

1. Создание агентов:

- ❖ Начальное расположение, где размещается агент, зависит от начальных условий и ограничений задачи.
- ❖ Агенты могут или быть в одной точке, или в разных с повторениями, или в разных без повторений.
- ❖ Также указывается первоначальное значение феромона, чтобы значения не были нулевыми.

2. Поиск подходящего решения:

Вероятность того, что произойдет переход из вершины i в j, можно определить по формуле:

$$P_{ij}(\alpha) = \frac{\tau_{ij}(\alpha)^{\alpha} \left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_{j \in \text{allowed nodes}} \tau_{ij}(\alpha)^{\alpha} \left(\frac{1}{d_{ij}}\right)^{\beta}}$$

где $\tau_{ij}(t)$ — уровень феромона,

d_{ij} — эвристическое расстояние,

α β — константные параметры.

Если $\alpha = 0$, с большей вероятностью выберется ближайший город;

Если $\beta = 0$, выбор будет основываться лишь на феромоне;

Необходим найденный экспериментальным способом компромисс между этими двумя величинами.

3. Изменение феромона

– Уровень феромона изменяется по формуле:

$$\tau_{ij}(t+1) = (1 - \rho)\tau_{ij}(t) + \sum_{k \in Colony \ that \\ used \ edge \ (i,j)} \frac{Q}{L_k}$$

где ρ — интенсивность испарения,

$L_k(t)$ — цена текущего решения k -го муравья,

Q — параметр, который имеет значение порядка цены оптимального решения,

$\frac{Q}{L_k(t)}$ - феромон, который откладывается k -м муравьем, который использует ребро.

4. Вспомогательные действия: в основном используют алгоритмы локального поиска.

Алгоритм искусственной пчелиной колонии

Пчелиный алгоритм – алгоритм роевого интеллекта, основан на имитации поведения колонии пчел, может использоваться в задачах оптимизации.

Необходимым условием для его применения является наличие некоторого топологического расстояния ли его аналога на области решений.

Для сбора нектара в пчелиной колонии применяется два вида пчел: пчелы-разведчики и пчелы-рабочие.

Пчелы-разведчики проводят исследование территории, окружающей улей, на предмет наличия нектара. По возвращении в улей они сообщают информацию о количестве нектара, направлении его расположения и расстоянии до него. Далее, в наиболее подходящие области вылетают *пчёлы-рабочие*, причем, чем больше нектара в данной области, тем больше пчел вылетает в нее. Кроме сбора меда, в их задачу входит обновление информации о данной и близлежащих областях.

Вместо поля с цветами рассмотрим область решений. Вместо нектара используем критерии задачи оптимизации, целевую функцию.

Можно задать определенное минимальное расстояние между двумя соседними областями. В этом случае, при возникновении наложения, область с худшим значением целевой функции отсекается. Вместо нее выбирается другая область. Данные области запоминаются и при следующей итерации в них посыпается определенное количество пчел.

Работу алгоритма можно разбить на два этапа:

1. Инициализация

- При инициализации для n разведчиков генерируются начальные положения.
- В простейшем случае используется метод случайного перебора.

$$X_i = \text{rand}(G(X)), i = 1, \dots, n^s$$

2. Локальный поиск

- После формирования списков лучших и перспективных областей, в их окрестности отправляются рабочие.
- В некоторых вариантах алгоритма число отправляемых пчел зависит от качества области, с точки зрения целевой функции. Эта зависимость может быть линейной или определяться по более сложным правилам.
- В данном случае, в каждую область высыпается фиксированное количество пчел, в зависимости от класса, которому принадлежит данная область.

- Каждую итерацию разведчики отправляются на новые области.

$$X_{(i-1)c^b+k} = N_{ij-1}^b + \text{Rnd} \cdot rad, i = 1, \dots, n^b, k = 1, \dots, c^b$$

$$X_{n^b c^b + (i-1)c^b + k} = N_{ij-1}^g + \text{Rnd} \cdot rad, i = 1, \dots, n^g, k = 1, \dots, c^g$$

В данном алгоритме используется несколько параметров:

- количество разведчиков,
- количество лучших и перспективных,
- радиус локальной разведки,
- количество пчел для каждого класса области,
- минимально возможное расстояние между соседними областями.

Качество получаемых решений и скорость работы алгоритма значительно зависит от выбора данных параметров.

Существует множество модификаций данного алгоритма. Они улучшают качество результата и скорость его работы.

В основном это происходит благодаря уменьшению зависимости от подбираемых параметров.

Сравнение методов

	Метод роя частиц	Муравьиный алгоритм	Алгоритм пчелиной колонии
Преимущества	<ul style="list-style-type: none"> Крайне низкая алгоритмическая сложность в реализации; Достаточно эффективен для глобальной оптимизации. 	<ul style="list-style-type: none"> Достаточно эффективен для TSP (Traveling Salesman Problem) с небольшим количеством узлов; Используется приложениях, которые могут адаптироваться к изменениям; Благодаря памяти всей колонии и случайному выбору пути не так сильно подвержен неудачным первоначальным решениям. 	<ul style="list-style-type: none"> Возможность эффективного разделения на параллельные процессы; Высокая скорость работы.
Применение	<ul style="list-style-type: none"> Задачи машинного обучения; Задачи оптимизации функций многих параметров, форм, размеров и топологий; Область проектирования Биоинженерия, биомеханика, биохимия. 	<ul style="list-style-type: none"> Расчеты компьютерных и телекоммуникационных сетей; Задача коммивояжёра; Задача раскраски графа; Задача оптимизации сетевых трафиков. 	<ul style="list-style-type: none"> Оптимизация управления; Оптимизация классификаторов.
Развитие	<ul style="list-style-type: none"> Представление МРЧ как многоагентной вычислительной системы; Возможности включения других, более сложных методов РИ. 	<ul style="list-style-type: none"> Гибридизация с генетическими алгоритмами; Использование базы нечётких правил. 	<ul style="list-style-type: none"> Снижение зависимости от устанавливаемых параметров; Объединение с генетическими алгоритмами.

Литература:

1. Sami Thampi SWARM INTELLIGENCE // arxiv.org - Алгоритм роя частиц // habrahabr.ru - Григорьев И. В., Мустафина С. А.
2. РЕАЛИЗАЦИЯ МЕТОДА РОЯ ЧАСТИЦ НА NVIDIA CUDA // Науч.Форум, 2014.
3. Алгоритм роя частиц. Описание и реализации на языках Python и C# // jenyay.net. 2011. Аноп М. Ф., Катуева Е. В., Михаличук В. И.
4. Алгоритмы роя пчел и частиц в задаче обеспечения надежности по постепенным отказам // Наука и Образование. — 2015. — № 1.
5. Лебедев Б. К., Лебедев В. Б. Размещение на основе метода пчелиной колонии // Известия ЮФУ. Технические науки. Чураков М., Якушев А.
6. Муравьиные алгоритмы // 2006. Матренин П.В.
7. Методы стохастической оптимизации: учебное пособие / П.В.
8. Матренин, М.Г. Гриф, В.Г. Секаев // Новосибирск: Изд-во НГТУ, 2016. – 67 с.
9. Водолазский, И. А. Ройвой интеллект и его наиболее распространённые методы реализации / И. А. Водолазский, А. С. Егоров, А. В. Краснов. — Текст: непосредственный // Молодой ученый. — 2017. — № 4 (138). — С. 147-153. — URL: <https://moluch.ru/archive/138/38900/> (дата обращения: 15.05.2022).



**Федеральное государственное бюджетное образовательное учреждение высшего
образования «Московский государственный технический университет
имени Н. Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)**

Цикл лекций по дисциплине **«Методология программной инженерии»**

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук.

rtn@bmstu.ru

Москва - 2024

Лекция 16



Документирование программных средств

Организация документирования ПС

Документация является органической, составной частью программного продукта для ЭВМ и требуются значительные ресурсы для ее создания и применения.

Тексты и объектный код программ для ЭВМ могут стать программным продуктом только в совокупности с комплексом документов, полностью соответствующих их содержанию и достаточных для его освоения, применения и изменения.

Документы должны быть:

- корректными,
- строго адекватными текстам программ и содержанию БД,
- систематически, структурировано и понятно изложены

**Оценивание масштаба проекта и его возможного влияния
на комплекс документации программного средства**



**Формирование концепции и спецификации требований
к проекту программного средства**



**Формирование предварительной спецификации требований
к номенклатуре комплекса документов**



**Оценивание влияния — приоритета каждого выбранного
документа на функциональную пригодность программного
средства**



**Оценивание относительных затрат на создание
каждого документа проекта программного средства**



**Оценивание уровня приоритета каждого документа с учетом
затрат и исключение нерентабельных документов из разработки**



**Выделение документов с высоким приоритетом
и утверждение состава комплекса документации для реализации
в проекте программного средства**

Процессы документирования программ и данных входят в весь ЖЦ сложных систем и ПС. Поэтому организация и реализация *работ по созданию документов должны распределяться между специалистами, ведущими непосредственное и преимущественное создание проектов комплексов .*

- ❖ При создании особо сложных систем целесообразно выделение специального коллектива, обеспечивающего организацию и реализацию основных системных работ по документообороту ПС.
- ❖ Совокупные затраты на документирование крупных программных продуктов могут достигать 20 – 30% от общей трудоемкости проекта и необходимого числа (десятки) специалистов в жизненном цикле проекта ПС.

В простых случаях, организация работ может быть упрощена, затраты на документирование снижаются до 10%, однако всегда целесообразно выделять специалистов, непосредственно ответственных за создание и контроль комплекта документов.

По своему назначению и ориентации на определенные задачи и группы пользователей, документацию ПС можно разделить:

1. **Технологическую документацию** процессов разработки и обеспечения всего ЖЦ, включающую подробные технические описания.

Подготавливается для специалистов, ведущих проектирование, разработку и сопровождение комплексов программ.

Обеспечивает возможность отчуждения, детального освоения, развития и корректировки ими программ и данных на всем жизненном цикле ПС;

Технологическая документация в наибольшей степени должна отражать процессы ЖЦ комплексов программ и данных и требования к этим документам.

Стандарты и нормативные документы, входящие в ЖЦ проекта ПС, должны регламентировать структуру, состав этапов, работ и документов ЖЦ ПС.

Стандарты в Технологической документации должны:

- формализовать выполнение и документирование конкретных работ при проектировании, разработке и сопровождении ПС;
- обеспечивать адаптацию документов к характеристикам среды разработки, внешней и ОС;
- регламентировать процессы обеспечения качества ПС и его компонентов, методы и средства их достижения, реальные значения достигнутых показателей качества.

2. Эксплуатационную документацию объекта и результатов разработки, создаваемую для конечных пользователей ПС и позволяющую им осваивать и квалифицированно применять эти средства для решения конкретных функциональных задач систем.

Эксплуатационная документация должна обеспечивать *отчуждаемость программного продукта* от первичных разработчиков и возможность освоения и эффективного применения комплексов программ достаточно квалифицированными специалистами – пользователями *ПС и системы*.

Эксплуатационные документы должны исключать возможность некорректного использования ПС за пределами условий эксплуатации, при которых документами гарантируются требуемые показатели качества функционирования ПС.

Основная ее задача состоит в фиксировании, полноценном использовании и обобщении результатов функционирования объектов и процессов всего жизненного цикла.

Базой эффективного управления проектом ПС и его документированием должен быть *План*.

- План проекта должен отражать рациональное сочетание целей, стратегий действий, конкретных процедур, доступных ресурсов и других компонентов, необходимых для достижения основной цели с заданным качеством.
- Контроль является органической функцией управления и должен иметь ряд средств регулирования поведения отдельных специалистов и коллектива разработчиков документов в целом.
- При планировании и разработке комплекс документации должен проверяться и аттестовываться на полноту в условиях ограниченных ресурсов, на корректность, адекватность и непротиворечивость отдельных документов.

Формирование требований к документации сложных программных средств

Масштаб проектов ПС является одним из важнейших факторов, влияющим на *формирование, структуры и содержания документации.*

Оценки масштаба проекта ПС должны быть проанализированы и скорректированы для установления в договоре между заказчиками и разработчиками исходного компромиссного масштаба, допустимого для разработки первичных требований к документации.

Каждый из разработчиков в той или иной степени должен привлекаться к управлению требованиями, как к проекту, так и его документации.

Разработчикам необходимо выработать профессиональные приемы для ***понимания и изложения в документах потребностей заказчиков и пользователей.***

- ❖ Команда разработчиков ПС получает представление о сложности и размере создаваемого продукта и составе его документации;
- ❖ Менеджеры проекта - базу для расчета содержания спецификаций, графиков, затрат и ресурсов;
- ❖ Тестировщики - планы тестирования, варианты испытаний и процедуры проверок;

- ❖ Специалисты по сопровождению и поддержке – получают представление о функциональности каждой составной части продукта;
- ❖ Клиенты отдела маркетинга и специалистов по продажам – должны иметь представление о конечном программном продукте;
- ❖ Составители документации, создающих шаблоны документов, руководства для пользователей и справки на основании спецификации требований к ПС – получат проект пользовательского интерфейса;
- ❖ Специалисты, ответственные за обучение персонала – получат спецификации требований к ПС и документацию для пользователей, а также для разработки обучающих материалов;
- ❖ Персонал, занимающийся юридической стороной проекта – проверяет, соответствуют ли требования к продукту существующим законам и постановлениям.

- Атрибуты качества ПС и полезность документов имеют различные меры, вследствие чего они в большинстве своем *не сопоставимы между собой*.
- Для обобщенного оценивания качества ПС необходим учет относительного влияния каждой конструктивной характеристики и документа, на функциональную пригодность ПС.
- При этом не всегда учитываются ресурсы для их реализации в конкретном ПС. Это часто приводит к не рациональным требованиям и документам, которые значительно отличаются:
 - либо по степени влияния на функциональную пригодность,
 - либо по величине ресурсов, необходимых для их реализации как полноценных документов.

Для управления и сопоставительного оценивания выбранных характеристик качества документов целесообразно каждому из них присваивать *коэффициент или приоритет* влияния на функциональную пригодность.

Аналогично, экспертами целесообразно оценивать относительные ресурсы, которые следует затрачивать на реализацию каждого документа.

Для каждого вида документов отношение коэффициента влияния на функциональную пригодность к относительным затратам на его достижение можно рассматривать как *обобщенный уровень приоритета реализации этого документа*.

**Оценивание масштаба проекта и его возможного влияния
на комплекс документации программного средства**



**Формирование концепции и спецификации требований
к проекту программного средства**



**Формирование предварительной спецификации требований
к номенклатуре комплекса документов**



**Оценивание влияния — приоритета каждого выбранного
документа на функциональную пригодность программного
средства**



**Оценивание относительных затрат на создание
каждого документа проекта программного средства**



**Оценивание уровня приоритета каждого документа с учетом
затрат и исключение нерентабельных документов из разработки**



**Выделение документов с высоким приоритетом
и утверждение состава комплекса документации для реализации
в проекте программного средства**

Планирование документирования проектов сложных ПС

Общее руководство процессом документирования комплексов программ можно разделить на *два уровня*:

- 1) адаптация состава и содержания документов к данной деловой, проблемно-ориентированной области, например, авиационной, медицинской, военной, финансовой или административной;
- 2) адаптация номенклатуры, структуры и содержания документов для каждого специфического проекта, контракта или предприятия.

Может представлять интерес оценка *ориентировочного физического объема документации* (например, в стандартных страницах А4 или эквивалентных объемов файлов).

В качестве *гипотетического примера* выделим *два масштаба проектов*: малый – 50 тысяч строк и крупный – один миллион строк, и выделим оценки на *технологическую и на эксплуатационную документацию*.

В эксплуатационной документации обычно не оформляются и не приводятся спецификации компонентов, тексты программ с комментариями, тесты и результаты тестирования, что резко сокращает номенклатуру документов до трех – семи видов

**Документы предварительных требований, спецификаций
и ресурсов для разработки программного средства**

№	Документы	Масштаб проекта		
		Крупный	Средний	Малый
1.1	Интервью заказчиков и пользователей о проблемах и целях создания программного продукта	+	+	
1.2	Результаты обследования и описание системы и целей разработки комплекса программ	+	+	+ -
1.3	Технико-экономическое обоснование проекта программного средства	+	+	+ -
1.4	Концепция и основные предложения по созданию программного средства	+	+	
1.5	Предварительный укрупненный план проектирования и разработки версии программного средства	+	+ -	
1.6	Системный проект, общее описание программного средства и среды разработки для согласования между заказчиком и разработчиком	+	+	
1.7	Техническое задание на предварительное (детальное) проектирование программного средства	+	+	+

Документы проектирования и выбора характеристик качества программного средства

№	Документы	Масштаб проекта		
		Крупный	Средний	Малый
2.1	Стандарты и ограничения на процессы проектирования программного средства	+	+	
2.2	Спецификация требований к системе и к комплексу программ	+	+	
2.3	Предварительное описание и контроль согласованности требований компонентов программного средства	+	+ -	+ -
2.4	Описание функционирования программного средства, взаимодействия с объектами внешней среды и человеко-машинного диалога	+	+ -	+ -
2.5	Описания алгоритмов компонентов (модулей) программного средства	+	+ -	
2.6	Описание информационного обеспечения программного средства и системы управления базами данных	+	+	+ -
2.7	Требования к характеристикам качества программного средства	+	+ -	+ -
2.8	Пояснительная записка к предварительному или детальному проекту программного средства	+	+	+
2.9	Описание концепции технологии автоматизированного проектирования программного средства	+	+ -	
2.10	План и поддерживающее его Руководство по документированию жизненного цикла программного средства	+	+ -	+ -
2.11	Ведомость предварительного или детального проекта программного средства	+	+ -	

Документы процессов разработки и программирования компонентов программных средств

№	Документы	Масштаб проекта		
		Крупный	Средний	Малый
3.1	План разработки компонентов программного средства	+	+	+
3.2	План обеспечения качества компонентов программного средства	+	+ -	
3.3	Стандарты кодирования компонентов программного средства	+	+	+ -
3.4	Руководство по программированию компонентов комплекса программ	+	+	+
3.5	Документация на разработанный функциональный программный компонент или модуль программного средства	+	+	+

**Документы верификации и тестирования компонентов
программных средств**

№	Документы	Масштаб проекта		
		Крупный	Средний	Малый
4.1	Состав базовых документов, регламентирующих верификацию и тестирование программных компонентов	+	+ -	
4.2	Исходные данные для верификации программных компонентов	+	+ -	
4.3	Результаты верификации корректности взаимодействия компонентов в составе программного средства	+	+ -	+ -
4.4	Исходные данные для тестирования компонентов	+	+	+
4.5	Организация, подготовка тестирования и обеспечение качества компонентов	+	+ -	+ -
4.6	Сценарии тестирования и спецификации тестов для каждого компонента	+	+	+
4.7	План тестирования программного компонента	+	+	+
4.8	Отчет о результатах верификации и тестирования компонентов	+	+	+
4.9	Методика комплексирования функциональных компонентов	+	+ -	
4.10	Оценка реализации комплексирования функциональных компонентов комплексов программ	+	+ -	

**Документы квалификационного тестирования, испытаний
и оценивания качества программных продуктов**

№	Документы	Масштаб проекта		
		Крупный	Средний	Малый
5.1	Методика генерации тестов, имитирующих внешнюю среду и обработку результатов квалификационного тестирования	+	+ -	+ -
5.2	Методика применения проблемно-ориентированной системы квалификационного тестирования и испытаний комплексов программ	+ -	+ -	
5.3	Методика, содержание и сценарии квалификационного тестирования и испытаний программных средств	+	+	+ -
5.4	Программа испытаний комплекса программ	+	+	+
5.5	Методики проведения испытаний комплекса программ по отдельным характеристикам качества	+	+	+
5.6	Протоколы по результатам испытаний функциональных компонентов и/или комплекса программ	+	+	+ -
5.7	Итоговый отчет результатов разработки программного продукта	+	+	+
5.8	Акт завершения работ по проекту программного продукта	+	+	+ -
5.9	Акт приемки программного продукта в промышленную эксплуатацию	+	+ -	+ -

**Документы сопровождения и конфигурационного управления
версиями программного продукта**

№	Документы	Масштаб проекта		
		Крупный	Средний	Малый
6.1	Описание среды жизненного цикла и конфигурации программного средства	+	+	+ -
6.2	План сопровождения версий программного продукта	+	+	+ -
6.3	План управления конфигурацией программного продукта	+	+	+ -
6.4	Программа управления конфигурацией программного продукта	+	+	+ -
6.5	Отчеты пользователей о выявленных дефектах и предложениях по корректировке комплекса программ	+	+	+
6.6	Описание выявленных дефектов и предложений по совершенствованию функций версии программного продукта	+	+	+
6.7	Описание подготовленных и утвержденных корректировок и обобщенных характеристик новой базовой версии программного продукта	+	+	+
6.8	Извещение пользователям о выпуске новой версии программного продукта и/или о прекращении сопровождения предшествующей версии	+	+ -	+ -
6.9	Описание новой базовой версии программного продукта	+	+	+
6.10	План передачи и внедрения новой базовой версии программного продукта пользователям	+	+ -	
6.11	Отчет о результатах эксплуатации снятой с сопровождения версии программного продукта и ее архивации	+	+ -	
6.12	Отчет о результатах тиражирования базовых версий, конфигурациях и параметрах пользовательских версий программного продукта	+	+ -	

Документы процессов эксплуатации программных продуктов

№	Документы	Масштаб проекта		
		Крупный	Средний	Малый
7.1	Общее описание системы, в которой используется программный продукт	+	+	+ -
7.2	Общие требования к формированию пользовательской документации программных продуктов	+	+	+ -
7.3	Описание административного управления программными продуктами системы	+	+ -	+ -
7.4	Руководство системного администратора программного продукта	+	+ -	+ -
7.5	Общее описание руководства пользователей программного продукта	+	+ -	+ -
7.6	Руководство оперативного пользователя программного продукта	+	+	+
7.7	Инструкция по формированию и ведению информации базы данных	+	+ -	+ -
7.8	Паспорт на программный продукт	+	+	+
7.9	Пользовательская документация на коммерческие пакеты — закрытые коробки программных средств			
7.10	Руководство по подготовке документов и обучению специалистов применению программного продукта	+	+ -	+ -

Понятие Онтологии. Новая концепция Интернета Web 3.0

Понятие Онтологии

Онтологи́я – учение о сущем; учение о бытии как таковом; раздел философии, изучающий фундаментальные принципы бытия, его наиболее общие сущности и категории, структуру и закономерности.

Позже этот термин стал применяться в информатике:

Онтологи́я – попытка всеобъемлющей и подробной формализации некоторой области знаний с помощью концептуальной схемы.

Современные онтологии строятся по большей части одинаково, независимо от языка написания. Обычно они состоят из понятий (или классов), экземпляров (или объектов), атрибутов (или свойств) и отношений (или связей). С помощью этих четырех типов элементов формируются концептуальные схемы, предназначенные для моделирования предметных областей и процессов.

Онтологии используются в процессе программирования как форма представления знаний о реальном мире или его части. Основные сферы применения — моделирование бизнес-процессов, семантическая паутина (англ. Semantic Web), искусственный интеллект.

Хотя термин «онтология» изначально философский, в информатике он принял самостоятельное значение.

Здесь есть два существенных отличия:

- 1.** Онтология в информатике должна иметь формат, который компьютер сможет легко обработать;
- 2.** Информационные онтологии создаются всегда с конкретными целями — решения конструкторских задач; они оцениваются больше с точки зрения применимости, чем полноты.

ОНТОЛОГИИ В ИНТЕРНЕТ

Зачем нужно описывать содержимое Web-страницы?

- Онтологии содержимого Web-страниц необходимы поисковым программам для улучшения качества поиска по Web.
- Идея построения спецификаций концептуализаций содержания Web-страниц находится в основании концепции так называемого Умного Web или Semantic Web.
- Semantic Web представляет собой следующее поколение World Wide Web, в котором кроме гипертекстовых документов содержатся описания семантики этих документов, а также описания семантики различных сервисов, предоставляющих эти документы конечным пользователям.

Новая концепция Интернета

Web1 - это первый интернет, в котором люди могли только просматривать веб-сайты и ничего больше не делать.

Web2 - это интернет, которым мы пользуемся сейчас, где мы можем создавать и обмениваться такими вещами, как фотографии, видео и сообщения.

Web3 - это новая идея для интернета, где люди могут не только создавать и делиться вещами, но и владеть ими. Он использует специальные технологии, такие как блокчейн, виртуальная реальность и интернет вещей, чтобы сделать интернет более справедливым и безопасным для всех.

Web3 — это концепция развития интернета следующего поколения, которая базируется на идее децентрализации.

Потенциальные преимущества Web3

- Повышенная безопасность вашей информации: Web3 будет использовать новые способы хранения и защиты информации от хакеров.
- Настоящая собственность на вашу информацию: В Web3 люди будут иметь больше контроля над своей информацией и даже смогут зарабатывать на ней деньги.
- Отсутствие цензуры
- В Web3 не будет центрального органа, который может несправедливо цензурировать людей. Крупным компаниям будет сложнее контролировать то, что люди видят и чем делятся.

- * Идеи умного Web давно были восприняты сообществом W3, в результате чего уже на протяжении более десяти лет ведутся работы по воплощению этих идей в жизнь.
- * Первой задачей, которую необходимо решить для этого, является разработка стандартного языка, который был бы понятен всем поисковым программам.

Наиболее используемые сейчас два языка:

- * Язык Resource Description Framework (RDF) – система описания ресурсов Web.
- * Web Ontology Language (OWL) – язык онтологии Web. OWL можно рассматривать как расширение языка RDF.

Язык RDF

- * Язык RDF разработан для того, чтобы описывать содержимое Web. В Semantic Web, когда говорят о каких-то сущностях Web, называют эти сущности ресурсами.
- * RDF представляет собой язык для описания таких ресурсов. Ввиду того что описания семантики документов должны быть понятны компьютерам, необходимо разработать специальные программы-агенты, которые производили бы такое чтение.
- * Также необходимо обеспечить возможность обмена информацией между различными программными агентами.
- * Таким образом, под RDF подразумевается не только сам язык, но также и различные дополнительные программные модули, необходимые для обеспечения полноценного чтения и обмена информацией, записанной на этом языке.
- * Этот факт подчеркивается в названии языка RDF (Resource Description Framework).

Главный элемент языка RDF—это тройка, или триплет. Тройка представляет собой совокупность трех сущностей:

- * Субъект.
- * Объект.
- * Предикат.

Предикаты еще часто называют свойствами. Тройка имеет также представление в виде графа вида субъект–предикат–объект, где субъект и объект представлены как узлы, а предикат выступает в роли ребра, которое эти узлы соединяет.

Язык OWL

- * OWL (Web Ontology Language) представляет собой язык, предназначенный для описания онтологий и разработанный консорциумом W3 специально для этих целей.
- * OWL построен как расширение RDF, позволяющий не только описывать классы и свойства, но также задавать ограничения на их использование.
- * На языке дескрипционной логики это означает, что логика, лежащая в основе OWL, содержит кроме описания отношений также и аксиомы, задающие соотношения между данными отношениями и различного рода ограничения последних.



Московский государственный технический
университет им. Н.Э. Баумана

Цикл лекций по дисциплине «Методология программной инженерии»

Автор: Романова Татьяна Николаевна,
доцент кафедры «Программное обеспечение
ЭВМ и информационные технологии»,
кандидат физико-математических наук,

rtn@bmstu.ru
rtn.51@mail.ru

Москва - 2024



Лекция 1

*Математика делает то, что можно,
и только так, как нужно.
Информатика делает то, что нужно,
но только так, как можно!*

Программистский фольклор

Введение в дисциплину

«Методология программной инженерии».

Основные понятия программной инженерии.

Понятие сложной системы

Введение в дисциплину

«Методология программной инженерии»

Накопление в мире знаний, опыта разработки и применения огромного количества различных сложных программ для ЭВМ, способствовал систематизации и обобщению методов и технологий их разработки. В результате сформировалась современная методология и инженерная дисциплина обеспечения процессов ЖЦ ПО – *программная инженерия* для различных областей применения.

Программная инженерия - это область компьютерной науки и технологии, которая занимается построением программных систем, настолько больших и сложных, что для этого требуется участие слаженных команд разработчиков различных специальностей и квалификаций.

Суть методологии программной инженерии состоит в применении систематизированного, научного и предсказуемого процесса проектирования, разработки и сопровождения программных средств.

Программная инженерия изучает различные методы и инструментальные средства разработки ПС с точки зрения достижения определенных целей.

Методы и средства, изучаемые в данной дисциплине, могут использоваться в разных технологических процессах и в разных технологиях программирования.

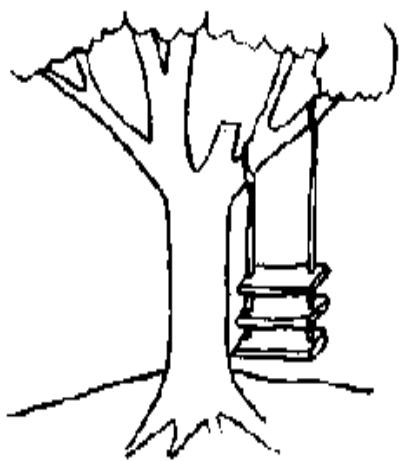
В феврале 2004 года была завершена работа над *Руководством к своду знаний по программной инженерии* (The Guide to the Software Engineering Body of Knowledge, **SWEBOK**).

SWEBOK – всестороннее описание знаний, необходимых для практической деятельности в области программной инженерии. Software Engineering (SE2004) рассчитан для подготовки профессиональных программистов в рамках бакалавриата. Программа дисциплины МПИ создана для магистров с учетом современных требований к разработке ПО.

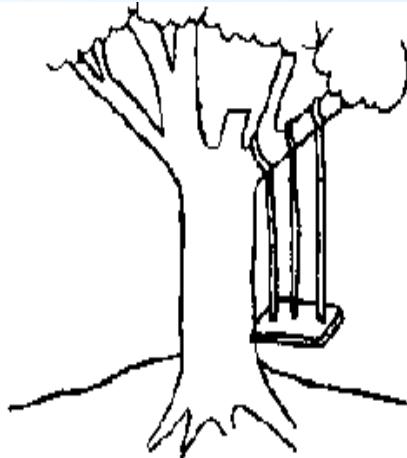
- ❖ *Основная цель курса лекций* - представить студентам современный комплекс задач, методов и стандартов *программной инженерии* для создания и развития сложных, многоверсионных, тиражируемых программных средств (ПС) и баз данных (БД) требуемого высокого качества.
- ❖ Изложение ориентировано на коллективную работу специалистов над крупными программными проектами.
- ❖ Внимание акцентировано на комплексе методов и процессов, которые способны непосредственно обеспечить эффективный ЖЦ сложных высококачественных программных продуктов и БД.

Сложность анализируемых объектов – комплексов программ и психологическая самоуверенность ряда программистов в собственной “непогрешимости”, часто приводят к тому, что реальные характеристики качества функционирования программных продуктов остаются неизвестными не только для заказчиков и пользователей, но также для самих разработчиков.

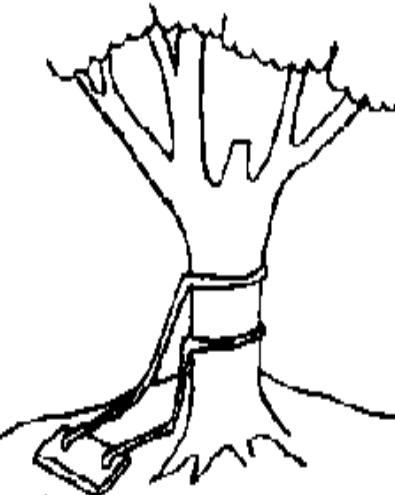
Отсутствие четкого декларирования в документах понятий и требуемых значений характеристик качества ПС вызывает конфликты между заказчиками-пользователями и разработчиками-поставщиками из-за разной трактовки одних и тех же характеристик.



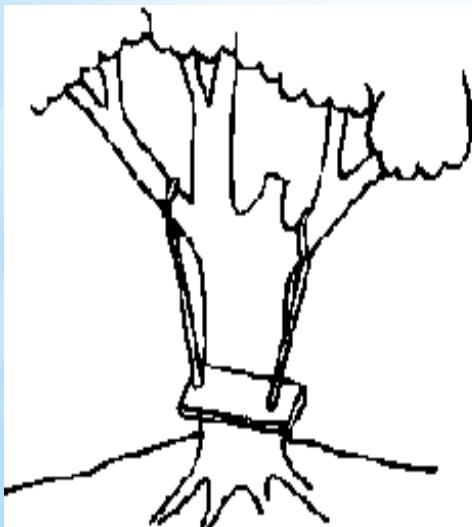
1. Как было предложено организатором разработки



2. Как было описано в техническом задании



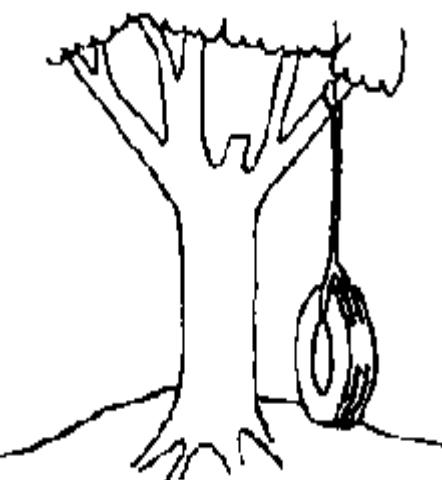
3. Как было спроектировано ведущим системным специалистом



4. Как было реализовано программистами



5. Как было внедрено



6. Чего хотел пользователь

Главное различие между *технологией программирования и программной инженерией* как дисциплинами для изучения заключается в способе рассмотрения и систематизации материала.

В *технологии программирования* акцент делается на изучении процессов разработки ПС и порядке их прохождения. Методы и инструментальные средства, используемые для разработки ПС, образуют технологические процессы, которые рассматриваются в дисциплине *технология программирования*.

Методы программной инженерии поддерживают и конкретизируют технологический процесс, а также отслеживают значения качества компонентов на всех этапах жизненного цикла ПС.

Для каждого проекта, выполняющего ответственные функции, должны разрабатываться и применяться система качества, специальные планы, методология и инструментальные средства разработки и испытаний, обеспечивающие требуемые качество, надежность и безопасность функционирования ПС.

*Методология программной инженерии и стандарты
регламентируют современные процессы управления
проектами сложных систем и программных средств.*

Они обеспечивают организацию, освоение и применение
апробированных, высококачественных процессов
проектирования, программирования, верификации,
тестирования и сопровождения программных средств и их
компонентов.

Тем самым эти проекты и процессы позволяют получать
стабильные, предсказуемые результаты и программные
продукты требуемого качества

Практическое применение *профилей стандартов*, сосредоточивших мировой опыт создания различных типов крупных комплексов программ, способствует значительному повышению производительности труда специалистов и качества создаваемых программных продуктов.

Методология – совокупность механизмов, применяемых при разработке программных систем и объединённых единым философским подходом.

Метод – концептуальное описание правил построение моделей системы, представляющих разные взгляды на проект с использованием специальных графических нотаций, которые определяет изобразительные средства и состав документации по проекту.

Технология – сложный комплекс, в основе которого лежит применение различных орудий, инструментов и аппаратов, использующий наработанные человечеством знания и умения [1].

Под **технологией программирования** будем понимать технологию разработки программных средств, включая в нее все процессы ЖЦ ПО и все процессы, связанные с созданием необходимой программной документации.

Информационная технология –
система методов и способов сбора,
получения, накопления, хранения,
обработки, анализа и передачи
информации с использованием средств
ЭВМ с целью повышения
эффективности, защищенности и
оперативной актуализации
производственных процессов.

- В *технологии программирования* методы рассматриваются с точки зрения организации технологических процессов.
- В *методологии программирования* методы рассматриваются с точки зрения основ их построения.
- *Методология программирования* определяется как совокупность механизмов, применяемых в процессе разработки программного обеспечения и объединенных одним общим философским подходом.(Г. Буч. Объектно-ориентированное проектирование с примерами применения. - М.: Конкорд, 1992).

Основные требования к методикам и методам проектирования ПО

- ❖ Метод должен отражать специфику подхода (парадигму программирования).
- ❖ Метод должен быть освоен всеми участниками проекта и должен быть наглядным с точки зрения полученных результатов.
- ❖ Должны существовать формальные переходы от этапов анализа к этапу проектирования и обратно.
- ❖ Должны существовать инструментальные средства, поддерживающие все эти методы

Понятие сложной системы

Международная организация по стандартизации (МОС) в области науки о компьютерах и ИТ определила понятие "система" следующим образом:

"Система - это множество элементов и отношений между ними, рассматриваемых, как единое целое".

Такими элементами могут быть как материальные, так и логические объекты, а также результаты деятельности людей (например, организационные формы предприятий, математические методы и языки программирования).

Классификация ИС

Классификации всегда относительны. Так в детерминированной системе можно найти элементы стохастических систем.

Цель любой классификации ограничить выбор подходов к отображению системы и дать рекомендации по выбору методов.

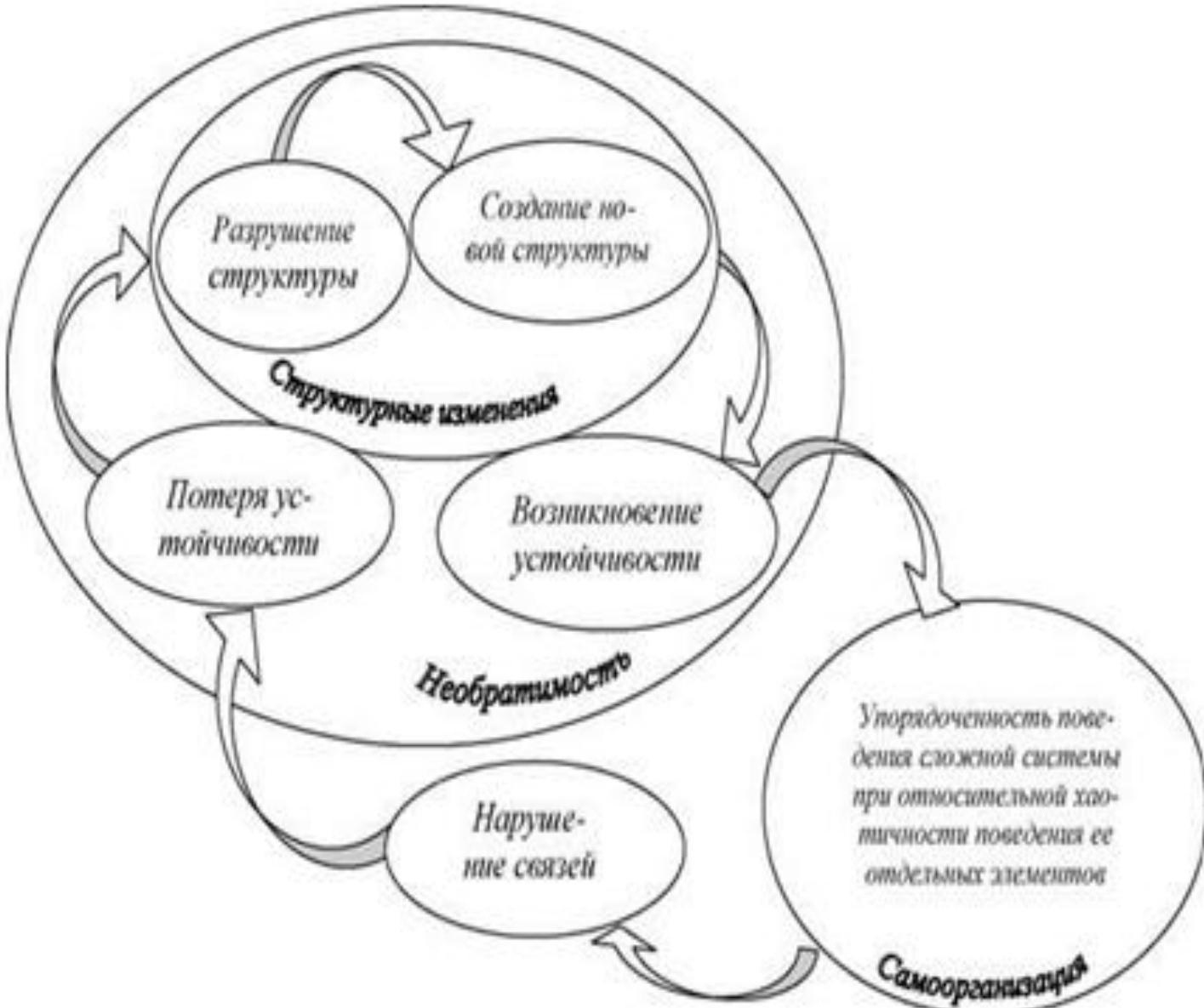
Системы классифицируются следующим образом:

- ❖ по виду отображаемого объекта — технические, биологические и др.;
- ❖ по виду научного направления — математические, физические, химические и т. п.;
- ❖ по виду формализованного аппарата представления системы — детерминированные и стохастические;
- ❖ по типу целеустремленности — открытые и закрытые;
- ❖ по сложности структуры и поведения — простые и сложные;
- ❖ по степени организованности — хорошо организованные, плохо организованные (диффузные), самоорганизующиеся системы.

Классификация систем по сложности

В зависимости от числа элементов, входящих в систему, выделяет четыре класса систем:

- малые системы ($10\ldots 10^3$ элементов),
- сложные ($10^4\ldots 10^7$ элементов),
- ультрасложные ($10^7\ldots 10^{30}$ элементов),
- суперсистемы ($10^{30}\ldots 10^{200}$ элементов).



Понятие сложной информационной системы

1. Система, которая разрабатывается не одним человеком, а группой разработчиков (более 5 человек).
2. Если количество строк исходного кода исчисляется сотнями тысяч или даже миллионами.
3. Если сложную задачу можно декомпозировать на более простые задачи (которые будут реализованы более мелкими подсистемами).
4. Если в требованиях встречаются *взаимоисключающие требования*. Например, нужно обработать огромные информационные потоки, и время отклика должно быть минимальным. Следует найти компромисс.

*Сложная программа обладает
следующими свойствами:*

- Она решает одну или несколько связанных задач, зачастую сначала не имеющих четкой постановки, настолько важных для каких-либо лиц или организаций, что те приобретают значимые выгоды от ее использования.
- Она должна быть удобной в использовании.
- Должна включать достаточно полную и понятную пользователям документацию
- Должна включать набор документов для обучения пользователей работе с программой.

- Ее низкая производительность на реальных данных приводит к значимым потерям для пользователей.
- Ее неправильная работа наносит ощутимый ущерб пользователям и другим организациям и лицам, даже если сбои происходят не слишком часто.
- Для выполнения своих задач она должна взаимодействовать с другими программами и программно-аппаратными системами, работать на разных платформах

- Пользователи, работающие с ней, приобретают дополнительные выгоды от того, что программа развивается, в нее вносятся новые функции и устраняются ошибки.
- Необходимо наличие проектной документации, позволяющей развивать ее, возможно, вовсе не тем разработчикам, которые ее создавали, без больших затрат на обратную разработку (реинжиниринг).

- В ее разработку вовлечено значительное количество людей (более 5-ти человек).
«Большую» программу практически невозможно написать с первой попытки, с небольшими усилиями и в одиночку.
- Велико количество ее возможных пользователей.

Основная задача, которую пытаются решить с помощью распределенных систем — обеспечение как можно большему числу пользователей максимально простого доступа к возможно большему количеству ресурсов.

Наиболее важными свойствами такой системы являются прозрачность, открытость, масштабируемость и безопасность.

Прозрачность (*transparency*).

Прозрачностью называется способность системы скрыть от пользователя физическое распределение ресурсов, а также аспекты их перераспределения и перемещения между различными машинами в ходе работы, репликацию (т.е. дублирование) ресурсов, трудности, возникающие при одновременной работе нескольких пользователей с одним ресурсом, ошибки при доступе к ресурсам и в работе самих ресурсов.

Степень прозрачности может быть различной, поскольку скрывать все эффекты, возникающие при работе распределенной системы, неразумно. Кроме того, *прозрачность системы и ее производительность* обычно находятся в обратной зависимости.

Например, при попытке преодолеть отказы в соединении с сервером большинство Web-браузеров пытается установить это соединение несколько раз, а для пользователя это выглядит как сильно замедленная реакция системы на его действия.

Открытость системы (openness) определяется как полнота и ясность описания интерфейсов работы с ней и служб, которые она предоставляет через эти интерфейсы.

Такое описание должно включать в себя все, что необходимо знать для того, чтобы пользоваться этими службами, независимо от реализации данной системы и платформы, на которой она развернута.

Открытость системы важна:

- ★ для обеспечения ее переносимости,
- ★ для облегчения использования системы и возможности построения других систем на ее основе.
- ★ Распределенные системы обычно строятся с использованием служб, предоставляемых другими системами.

Именно поэтому использование компонентных технологий при разработке практически полезного распределенного ПО неизбежно.

Масштабируемость системы (scalability). — это зависимость изменения ее характеристик от числа ее пользователей, числа подключенных ресурсов, а также от степени географической распределенности системы.

В число значимых характеристик при этом попадают функциональность, производительность, стоимость, трудозатраты на разработку, на внесение изменений, на сопровождение, на администрирование, удобство работы с системой.

Большую роль играет **административная масштабируемость системы** — зависимость удобства работы с ней от числа административно независимых организаций, вовлеченных в ее обслуживание.

При реализации очень больших систем (поддерживающих работу тысяч и более пользователей, включающих сотни и более машин) хорошая масштабируемость может быть достигнута только с помощью децентрализации основных служб системы и управляющих ею алгоритмов.

Вариантами такого подхода являются следующие:

- 1.Децентрализация обработки запросов за счет использования нескольких машин для этого.
- 2.Децентрализация данных за счет использования нескольких хранилищ данных или нескольких копий одного хранилища.
- 3.Использование, где это возможно,
асинхронной связи — передачи сообщений без приостановки работы до прихода ответа.

4. Децентрализация алгоритмов работы за счет использования «уникальных» алгоритмов,

- не требующих полной информации о состоянии системы,
- способных продолжать работу при сбое одного или нескольких ресурсов системы,
- не предполагающих единого хода времени на всех машинах, входящих в систему.

5. Использование комбинированных систем организации взаимодействия, основанных на следующих схемах:

- Иерархическая организация систем, хорошо масштабирует задачи поиска информации и ресурсов.
- Репликация* — построение копий данных и их распределении по системе для балансировки нагрузки на разные ее элементы — и ее частном случае, *кэшировании*, организующем хранение результатов наиболее часто используемых запросов как можно ближе к клиенту.
- Взаимодействие точка-точка (peer-to-peer, P2P), обеспечивающем независимость взаимодействующих машин от других машин в системе.

Безопасность (safety).

Так как распределенные системы вовлекают в свою работу множество пользователей, машин и географически разделенных элементов, вопросы их безопасности получают гораздо большее значение, чем при работе обычных приложений, сосредоточенных на одной физической машине. Это связано как с невозможностью надежно контролировать доступ к различным элементам такой системы, так и с доступом к ней гораздо более широкого и разнообразного по своему поведению сообщества пользователей.

Понятие безопасности включает следующие характеристики:

Сохранность и целостность данных.

При обеспечении групповой работы многих пользователей с одними и теми же данными нужно обеспечивать их сохранность, т.е. предотвращать исчезновение данных, введенных одним из пользователей, и в тоже время целостность, т.е. непротиворечивость, выполнение всех присущих данным ограничений.

- Это непростая задача, не имеющая решения, удовлетворяющего все стороны во всех ситуациях. При одновременном изменении одного и того же элемента данных разными пользователями итоговый результат должен быть непротиворечив, и поэтому часто может совпадать только с вводом одного из них.
- Как будет обработана такая ситуация и возможно ли ее возникновение вообще, зависит
 - ★ от дополнительных требований к системе,
 - ★ от принятых протоколов работы,
 - ★ от того, какие риски — потерять данные одного из пользователей или значительно усложнить работу пользователей с системой — будут сочтены более важными.

Защищенность данных и коммуникаций.

❖ При работе с коммерческими системами, с системами, содержащими большие объемы персональной и бизнес-информации, с системами обслуживания пользователей государственных ведомств очень важна защищенность, как информации, постоянно хранящейся в системе, так и информации одного сеанса работы.

Для распределенных систем обеспечить защищенность гораздо сложнее, поскольку нельзя физически изолировать все элементы системы и разрешить доступ к ней только людям.

Отказоустойчивость и способность к восстановлению после ошибок.

- ❖ Одним из достоинств распределенных систем является возможность построения более надежно работающей системы из
 - ❖ не вполне надежных компонентов.
 - ❖ Однако для того, чтобы это достоинство стало реальным, необходимо тщательное проектирование систем с тем, чтобы избежать зависимости работоспособности системы в целом от ее отдельных элементов.

Перед разработчиками систем, удовлетворяющих перечисленным свойствам, встает огромное количество проблем.

Решать их все сразу просто невозможно в силу ограниченности человеческих способностей. Чтобы хоть как-то структурировать эти проблемы, их разделяют по следующим аспектам:

- ❑ Организация связи и передачи данных между элементами системы.
- ❑ Поддержка идентификации и поиска отдельных ресурсов внутри системы.

- ❑ Организация работ в рамках процессов и потоков.
- ❑ Синхронизация параллельно выполняемых потоков работ.
- ❑ Поддержка целостности данных и непротиворечивости вносимых изменений.
- ❑ Организация отказоустойчивой работы.
- ❑ Организация защищенности данных и коммуникаций.