Самостоятельная работа 1

# Схемы ISO 5807:1985 (ГОСТ 19.701-90)

## Применение

Схемы ISO 5807:1985, также известные как ГОСТ 19.701-90 в России, являются стандартом, который определяет общие требования к созданию схем в технической документации. Эти схемы широко используются в инженерии, архитектуре, строительстве и других отраслях для визуализации и передачи информации о различных процессах, системах и устройствах.

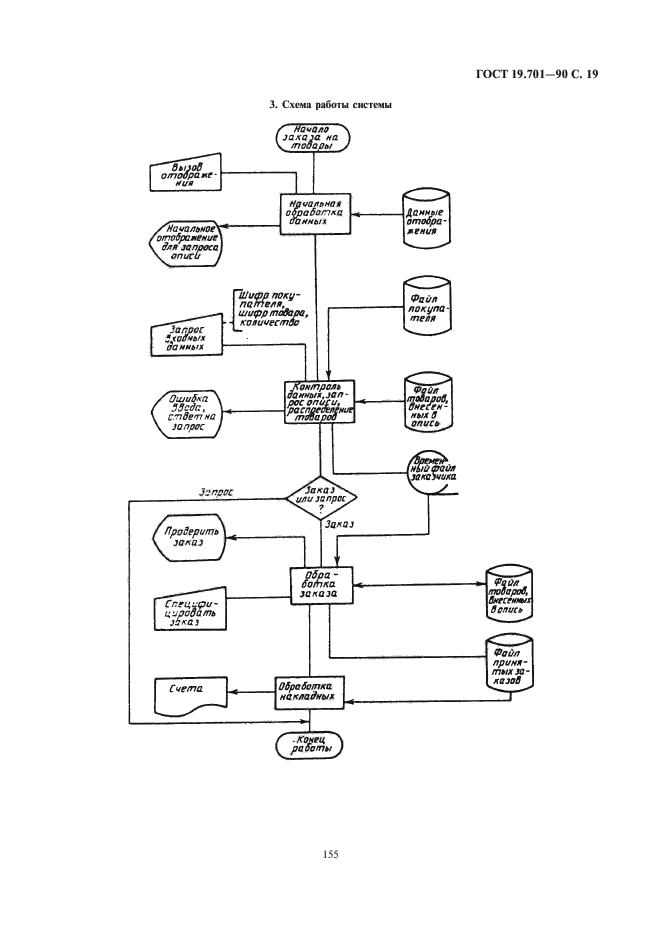


Рис 1. Пример Схемы по ГОСТу

## Преимущества

* Единый стандарт

Стандарт указывает один способ создания схем, благодаря чему все схемы, даже написанные разными специалистами, будут понятны всем

* Эффективная коммуникация

С данным стандартом знакомо множество специалистов из разных отраслей, что позволяет быстро и эффективно обмениваться информацией между различными участниками проекта.

## Недостатки

* Ограничения в представлении информации

Некоторые алгоритмы и процессы могут быть сложно представить на схемах, особенно если они очень сложны или абстрактны. Схемы получаются запутанными и трудными к пониманию даже для специалистов

* Возможность непонимания

Некоторые люди могут иметь трудности с пониманием схем, особенно если у них не было предварительной технической подготовки.

# ДРАКОН-схемы

## Применение

ДРАКОН-схемы - это метод визуализации алгоритмов, который представляет собой нотацию, основанную на графическом представлении алгоритмов программ, которая помогает упростить понимание структуры и логики алгоритмов.

ДРАКОН-схемы похожи на ISO 5807:1985, но обладают более строгими правилами и более современным представлением алгоритмов.

Такие схемы применяются в информационных технологиях, особенно в области программирования, а также в обучении алгоритмическим навыкам. Они помогают визуализировать и разрабатывать сложные алгоритмы. Эти схемы также могут использоваться для документации программного обеспечения, обучающих материалов, и анализа алгоритмов. для начинающих разработчиков.

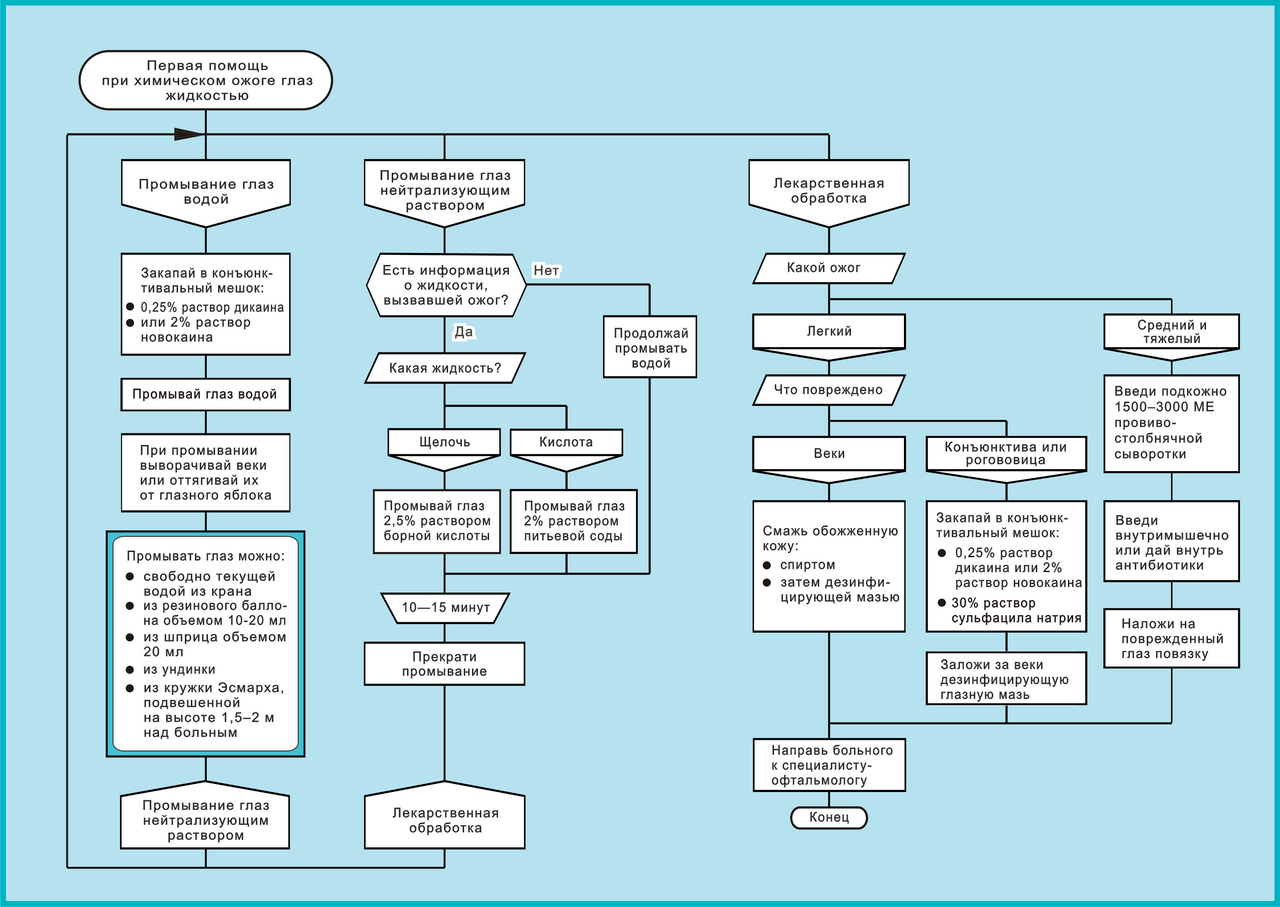


Рис 2. Пример Схемы ДРАКОН

## Преимущества

* Ясное визуальное представление

ДРАКОН-схемы обеспечивают ясное визуальное представление алгоритмов, что делает их легкими для понимания и интерпретации.

* Структурирование алгоритмов

С помощью ДРАКОН-схем можно структурировать алгоритмы, разделяя их на логические блоки и определяя последовательность выполнения операций.

* Удобство для обучения и обучения программированию:

Схемы облегчают процесс обучения программированию, поскольку они предоставляют интуитивно понятное визуальное представление алгоритмов.

* Строгий стандарт

ДРАКОН-схемы отличаются от ГОСТа более строгим и семантичным стандартом, что делает их более точными и удобными для интерпретации.

## Недостатки

* Ограничения в выражении сложных алгоритмов

Сложные алгоритмы трудно или неэффективно представлять с помощью ДРАКОН-схем, особенно если они содержат множество ветвлений или повторяющихся структур.

* Трудности в масштабировании

При разработке крупных и сложных программных проектов может возникнуть сложность в масштабировании ДРАКОН-схем и их поддержке на протяжении всего проекта.

* Затраты времени на создание

Создание крупных ДРАКОН-схем может быть трудоемким процессом, особенно если алгоритмы сложны и содержат множество шагов.

* Непопулярность

ДРАКОН-схемы имеют несколько узких применений в индустрии, но их практическое использование сильно снизилось в современной разработке программного обеспечения.

# Диаграммы деятельности UML

## Применение

UML (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур. Он предоставляет набор графических инструментов и концепций, которые позволяют разработчикам, аналитикам и другим участникам проекта ясно и эффективно описывать различные аспекты программного продукта.

Одной из основных областей применения UML является анализ и проектирование программного обеспечения. С помощью UML разработчики могут создавать модели, которые описывают функциональные требования к системе, ее структуру и архитектуру. Эти модели помогают разработчикам и аналитикам лучше понять требования к системе и принять информированные решения о ее дизайне и реализации.

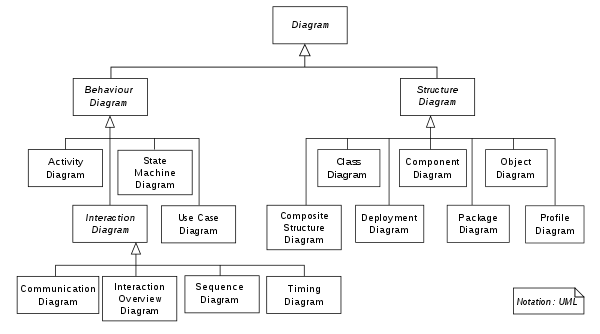


Рис 3. Пример Схемы UML

## Преимущества

* Ясное и структурированное представление

UML обеспечивает ясное и структурированное визуальное представление программных систем практически со всех возможных точек зрения.

* Стандартизация

UML является стандартизированным языком моделирования, что обеспечивает единообразие в создании и моделей в различных между различными специалистами.

* Гибкость и масштабируемость

UML предлагает широкий набор диаграмм и элементов, которые могут быть использованы для моделирования различных аспектов программных систем любого масштаба и сложности.

* Автоматическая генерация кода:

Инструменты UML предоставляют возможность автоматической генерации кода на основе моделей.

## Недостатки

* Сложность и избыточность

Некоторые аспекты UML могут быть слишком сложными или избыточными для конкретных проектов, что может привести к излишней сложности в создании моделей.

* Затраты времени и ресурсов

Создание и поддержка UML-моделей может потребовать значительных затрат времени и ресурсов, особенно для крупных и сложных проектов.

* Необходимость обучения и опыта

Использование UML требует от участников проекта соответствующего обучения и опыта, чтобы эффективно создавать и интерпретировать модели.

# Диаграммы потоков данных DFD

## Применение

Диаграммы потоков данных (DFD - Data Flow Diagrams) представляют собой графическое представление потоков данных в информационной системе. Они используются для моделирования и анализа процессов, включая потоки данных, внешние сущности, которые взаимодействуют с системой, и обработку данных внутри системы.

Самые распространенные нотации для построения DFD:

● нотация Йордона ― Де Марко

● нотация Гейна ― Сарсона.

Обе нотации состоят из различных элементов, таких как процессы (функции или операции), внешние сущности (источники и получатели данных), потоки данных (связи между процессами и внешними сущностями) и хранилища данных (места, где данные могут быть временно или постоянно храниться), но различаются в символах, которые используются для обозначения элементов.

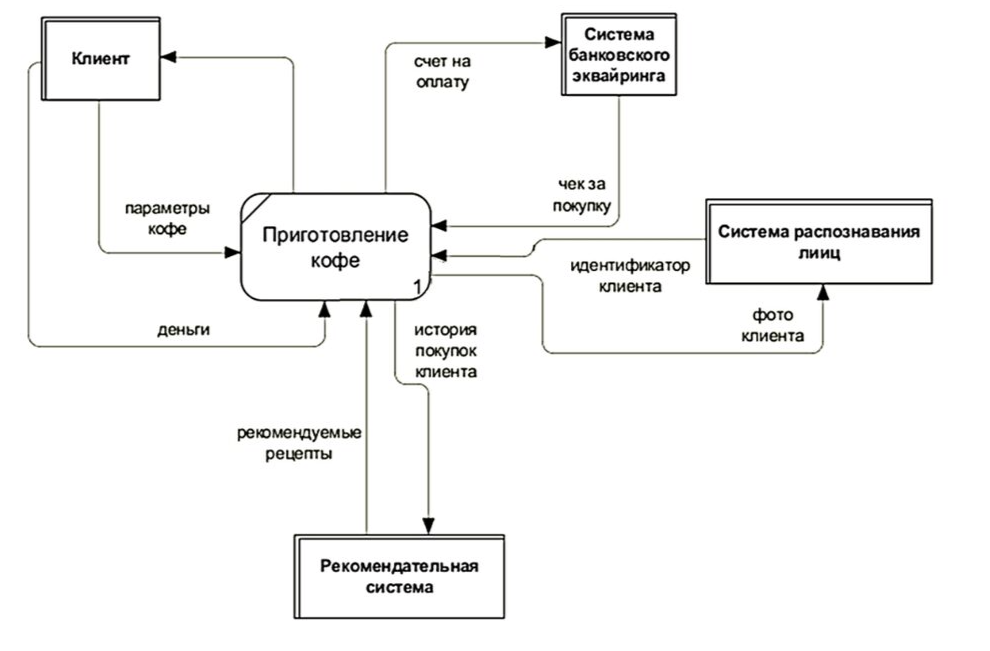


Рис 4. Пример Диаграммы DFD

## Преимущества

* Простота и ясность

DFD предоставляют простой и понятный способ визуализации потоков данных и процессов в информационной системе.

* Структурированное моделирование

DFD помогают структурировать процессы и данные в системе, что облегчает их анализ и проектирование.

* Удобство в использовании

Создание и интерпретация DFD не требует специальных навыков или инструментов, что делает их доступными для широкого круга пользователей.

## Недостатки

* Ограниченность в выражении сложных систем

DFD может быть недостаточно для моделирования очень сложных и абстрактных систем, так как они не предоставляют механизмов для описания более сложных структур и поведения.

* Недостаточная детализация

В некоторых случаях DFD могут быть слишком абстрактными и не содержать достаточной детализации для полного понимания и анализа процессов в системе.

# Р-схемы

## Применение

Р-схема - нагруженный по дугам ориентированный граф, изображаемый с помощью вертикальных и горизонтальных линий и состоящий из структур (подграфов), каждая из которых имеет только один вход и один выход. Графический язык Р-схем сконструирован специально, чтобы осуществить плавный переход от алгоритма, представленного в виде двумерной картинки, к двумерной программе.

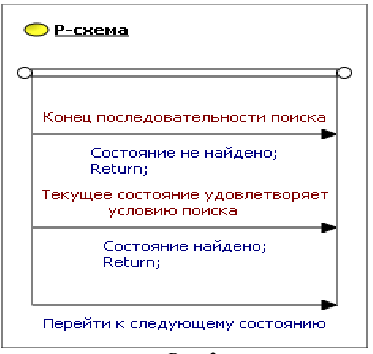


Рис 5. Пример P-схемы

## Преимущества

* Целенаправленность и простота

Используется для перехода от диаграммы алгоритма к программному коду.

## Недостатки

* Неизвестность

Очень мало информации и, как следствие, мало применений и специалистов, которые бы разбирались в данных схемах.

* Узкая направленность

Узкий круг выполняемых задач, которым трудно оправдать обучение и использование данных схем в дальнейшем.

# для программирования микроконтроллеров (SFC, FBD)

## Применение

Программируемые логические контроллеры (PLC) - это специализированные устройства, используемые для автоматизации и управления процессами в промышленности, производстве и других областях. Языки программирования для PLC предоставляют средства для создания программного обеспечения, которое управляет поведением и функциональностью PLC.

SFC (Sequential Function Chart) - графический язык, предназначенный для написания программ последовательного управления технологическим процессом.

Основными элементами языка являются:

* Состояния, в которых выполняются определенные действия;
* Переходы из состояния в состояние
* Альтернативное ветвление алгоритма,
* Параллельное ветвление
* Переход к заданному состоянию
* Остановка процесса

FBD (Function Block Diagram) — графический язык программирования стандарта, предназначенный для программирования программируемых логических контроллеров. Программа образуется из списка цепей, выполняемых последовательно сверху вниз. Цепи могут иметь метки. Инструкция перехода на метку позволяет изменять последовательность выполнения цепей для программирования условий и циклов.

Так как такие языки созданы специально для PLC, то их трудно сравнивать с другими языками общего назначения. Между собой же FBD и SFC крайне похожи.

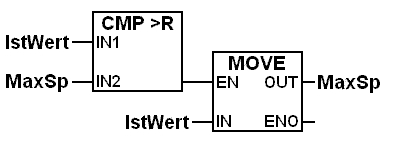


Рис 6. Пример FBD

# для описания бизнес-процессов (IDEF0, BPMN)

## Применение

Языки для описания бизнес-процессов представляют собой набор нотаций и синтаксических правил, используемых для моделирования, анализа и документирования бизнес-процессов в организации. Они позволяют представить последовательность шагов, действий и ресурсов, необходимых для выполнения конкретного бизнес-процесса. Два таких языка - IDEF0 и BPMN

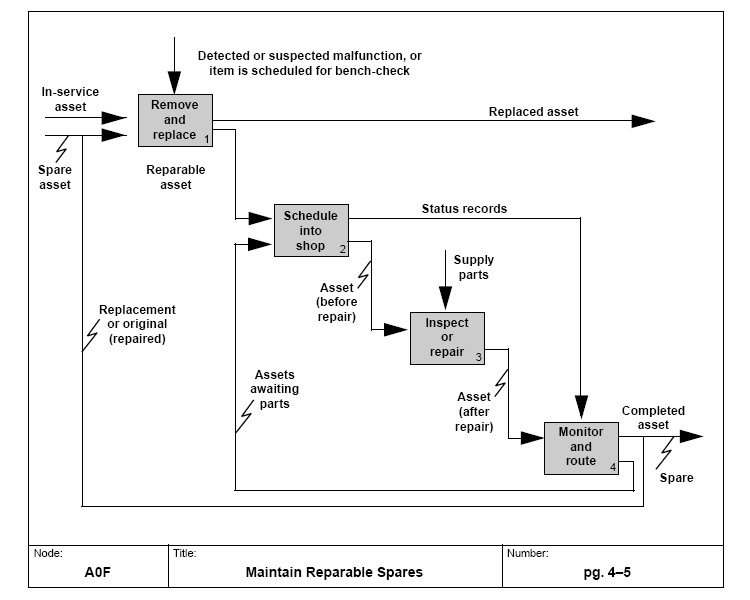


Рис 7. Пример IDEF0

## Преимущества

* Структурирование процессов

IDEF0 обеспечивает структурирование и иерархическое представление бизнес-процессов, что позволяет лучше понять их организацию и взаимосвязи между различными элементами.

* Удобство визуализации

Графическое представление бизнес-процессов с помощью IDEF0 делает их более понятными и доступными для анализа участникам проекта.

* Стандартизация

IDEF0 - это стандартизированный метод моделирования, который облегчает совместную работу и обмен информацией между участниками проекта.

## Недостатки

* Ограниченность возможностей

IDEF0 не так гибок и общирен, его может не хватить для некоторых сложных моделирований.

* Сложность моделирования

Для некоторых может стать слишком сложным моделирование бизнес-процессов из-за большого количества возможных элементов и правил их сочетания.

## Диаграммы Нэсси-Шнейдермана

Диаграммы Нэсси-Шнейдермана (Nassi-Shneiderman diagrams) - это графические средства, используемые для проектирования и визуализации алгоритмов. Они были разработаны в качестве альтернативы текстовому представлению алгоритмов программирования. Диаграммы Нэсси-Шнейдермана часто используются в учебных целях для обучения программированию, а также в процессе анализа и проектирования алгоритмов в различных областях, включая информационные технологии, инженерию и управление.

Основные элементы диаграмм Нэсси-Шнейдермана включают в себя:

* Блоки

Представляют собой базовые операции или действия, которые выполняются в рамках алгоритма. Эти блоки обычно содержат текстовые метки, описывающие действие, которое они представляют.

* Стрелки

Используются для указания порядка выполнения операций. Они соединяют блоки, показывая направление выполнения программы от одного действия к другому.

* Управляющие структуры

Такие как условные операторы, циклы и процедуры. Они позволяют управлять потоком выполнения программы, в зависимости от определенных условий.

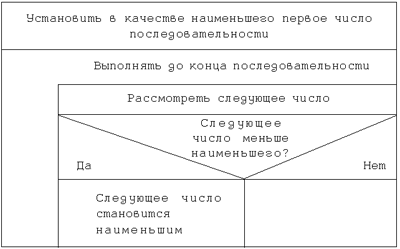


Рис 8. Пример диаграммы Нэсси-Шнейдермана