ИНТЕГРИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Лекция №7

Структурный анализ и проектирование ИСУ

Структурный анализ

Структурный анализ - это систематический пошаговый подход к анализу требований и проектированию спецификаций системы независимо от того, является ли она существующей или создается вновь.

Все методологии структурного анализа базируются на ряде общих принципов, часть из которых регламентирует организацию работ на начальных этапах ЖЦ, а часть используется при выработке рекомендаций по организации работ. В качестве двух базовых принципов используются следующие: принцип "разделяй и властвуй" и принцип иерархического упорядочивания. Первый является принципом решения трудных проблем путем разбиения их на множество меньших независимых задач, легких для понимания и решения. Второй принцип в дополнение к тому, что легче понимать проблему если она разбита на части, декларирует, что устройство этих частей также существенно для понимания. Понимание проблемы резко облегчается при организации ее частей в древовидные иерархические структуры, т.е. система может быть понята и построена по уровням, каждый из которых добавляет новые детали.

Структурный анализ

Выделение двух базовых принципов инженерии программного обеспечения вовсе не означает, что остальные принципы являются второстепенными, игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к неуспеху всего проекта). Отметим основные из таких принципов.

- Принцип абстрагирования - заключается в выделении существенных с некоторых позиций аспектов системы и отвлечение от несущественных с целью представления проблемы в простом общем виде.
- Принцип формализации - заключается в необходимости строгого методического подхода к решению проблемы.
- Принцип упрятывания - заключается в упрятывании несущественной на конкретном этапе информации: каждая часть "знает" только необходимую ей информацию.
- Принцип концептуальной общности - заключается в следовании единой философии на всех этапах ЖЦ (структурный анализ - структурное проектирование - структурное программирование - структурное тестирование).

Структурный анализ

- Принцип полноты - заключается в контроле на присутствие лишних элементов.
- Принцип непротиворечивости - заключается в обоснованности и согласованности элементов.
- Принцип логической независимости - заключается в концентрации внимания на логическом проектировании для обеспечения независимости от физического проектирования.
- Принцип независимости данных - заключается в том, что модели данных должны быть проанализированы и спроектированы независимо от процессов их логической обработки, а также от их физической структуры и распределения.
- Принцип структурирования данных - заключается в том, что данные должны быть структурированы и иерархически организованы.
- Принцип доступа конечного пользователя - заключается в том, что пользователь должен иметь средства доступа к базе данных, которые он может использовать непосредственно (без программирования).

DFD

Диаграммы потоков данных (DFD) являются основным средством моделирования функциональных требований проектируемой системы. С их помощью эти требования разбиваются на функциональные компоненты (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств - продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Для изображения DFD традиционно используются две различные нотации: Йодана (Yourdon) и Гейна-Сарсона (Gane-Sarson).

Целью рассматриваемых методологий является преобразование общих, неясных знаний о требованиях к системе в точные (насколько это возможно) определения. Обе методологии фокусируют внимание на потоках данных, их главное назначение - создание базированных на графике документов по функциональным требованиям. . Методологии поддерживаются традиционными нисходящими методами проектирования спецификаций и обеспечивают один из лучших способов связи между аналитиками, разработчиками и пользователями системы за счет интеграции множества следующих средств:

DFD

- ***DFD - диаграммы потоков данных.*** Являются графическими иерархическими спецификациями, описывающими систему с позиций потоков данных. В состав DFD могут входить четыре графических символа, представляющих потоки данных, процессы преобразования входных потоков данных в выходные, внешние источники и получатели данных, а также файлы и БД, требуемые процессами для своих операций.
- ***Словари данных.*** Являются каталогами всех элементов данных, присутствующих в DFD, включая групповые и индивидуальные потоки данных, хранилища и процессы, а также все их атрибуты.
- ***Миниспецификации обработки***, описывающие DFD-процессы нижнего уровня и являющиеся базой для кодогенерации. Фактически миниспецификации представляют собой алгоритмы описания задач, выполняемых процессами: множество всех миниспецификаций является полной спецификацией системы. Миниспецификации содержат номер и/или имя процесса, списки входных и выходных данных и тело (описание) процесса, собственно и являющееся спецификацией алгоритма или операции, трансформирующей входные потоки данных в выходные.

DFD

Компонента	Нотация Йодана	Нотация Гейна-Сарсона
поток данных	имя →	имя →
процесс	имя / номер	номер / имя
хранилище	имя	имя
внешняя сущность	имя	имя

Основные символы диаграммы потоков данных

DFD

DFD-диаграммы являются ключевой частью документа спецификации требований. Каждый узел - процесс в DFD может развертываться в диаграмму нижнего уровня, что позволяет на любом уровне абстрагироваться от деталей (отметим, что структурные методологии, ориентированные на потоки управления, не обладают этим свойством). Проектные спецификации строятся по DFD и их миниспецификациям автоматически. Наиболее часто для описания проектных спецификаций используется методика структурных карт Джексона, иллюстрирующая иерархию модулей, связи между ними и некоторую информацию об их исполнении (последовательность вызовов, итерацию).

Отметим, что DFD моделируют функции, которые система должна выполнять, но ничего (или почти ничего) не сообщают об отношениях между данными, а также о поведении системы в зависимости от времени - для этих целей методологии использует диаграммы "сущность-связь" и диаграммы переходов состояний, соответственно.

DFD

Главной отличительной чертой методологии Гейна-Сарсона является наличие этапа моделирования данных, определяющего содержимое хранилищ данных (БД и файлов) в DFD в Третьей Нормальной Форме. Этот этап включает построение списка элементов данных, располагающихся в каждом хранилище данных; анализ отношений между данными и построение соответствующей диаграммы связей между элементами данных; представление всей информации по модели в виде связанных нормализованных таблиц. Кроме того, методологии отличаются чисто синтаксическими аспектами, так, например, различны графические символы, представляющие компоненты DFD.

DFD

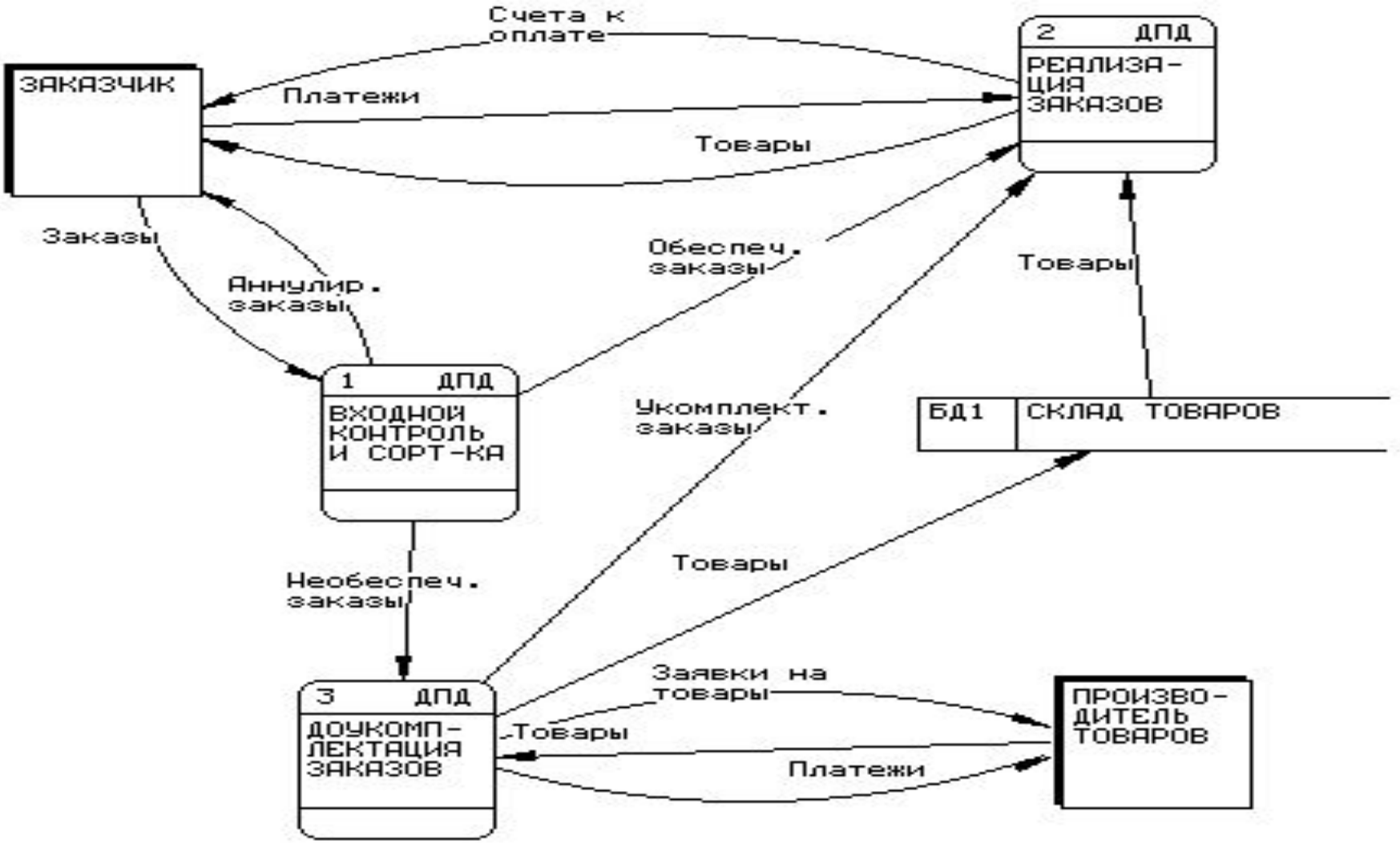
Таким образом, методы в рассматриваемых подходах представляют собой "кулинарную книгу" с рецептами, помогающими от чистого листа бумаги или экрана перейти к хорошо организованной модели системы. Эти рецепты основаны на простой концепции нисходящего поэтапного разбиения функций системы на подфункции.

На первом этапе формируется контекстная диаграмма верхнего уровня, идентифицирующая границы системы и определяющая интерфейсы между системой и окружением. Затем, после интервьюирования эксперта предметной области, формируется список внешних событий, на которые система должна реагировать. Для каждого из таких событий строится пустой процесс ("bubble") в предположении, что его функция обеспечивает требуемую реакцию на это событие, которая в большинстве случаев включает генерацию выходных потоков и событий (но может также включать и занесение информации в хранилище данных для ее использования другими событиями и процессами). На следующем уровне детализации аналогичная деятельность осуществляется для каждого из пустых процессов.

DFD

В качестве примера рассмотрим верхний уровень функциональной модели компании, занимающейся распределением товаров по заказам. Заказы подвергаются входному контролю и сортировке. Если заказ не отвечает номенклатуре товаров или оформлен неправильно, то он аннулируется с соответствующем уведомлением заказчика. Если заказ не аннулирован, то определяется, имеется ли на складе соответствующий товар. В случае положительного ответа выписывается счет к оплате и предъявляется заказчику, при поступлении платежа товар отправляется заказчику. Если заказ не обеспечен складскими запасами, то отправляется заявка на товар производителю. После поступления требуемого товара на склад компании заказ становится обеспеченным и повторяет вышеописанный маршрут. При построении данной модели использована нотация Гейна-Сарсона.

DFD



Пример диаграммы Гейна-Сарсона

SADT

SADT (Structured Analysis and Design Technique) - одна из самых известных методологий анализа и проектирования систем, введенная в 1973 г. Россом (Ross). SADT успешно использовалась в военных, промышленных и коммерческих организациях для решения широкого спектра задач, таких как программное обеспечение телефонных сетей, системная поддержка и диагностика, долгосрочное и стратегическое планирование, автоматизированное производство и проектирование, конфигурация компьютерных систем, обучение персонала, встроенное ПО для оборонных систем, управление финансами и материально-техническим снабжением и др. Данная методология широко поддерживается Министерством обороны США, которое было инициатором разработки стандарта IDEF0 как подмножества SADT.

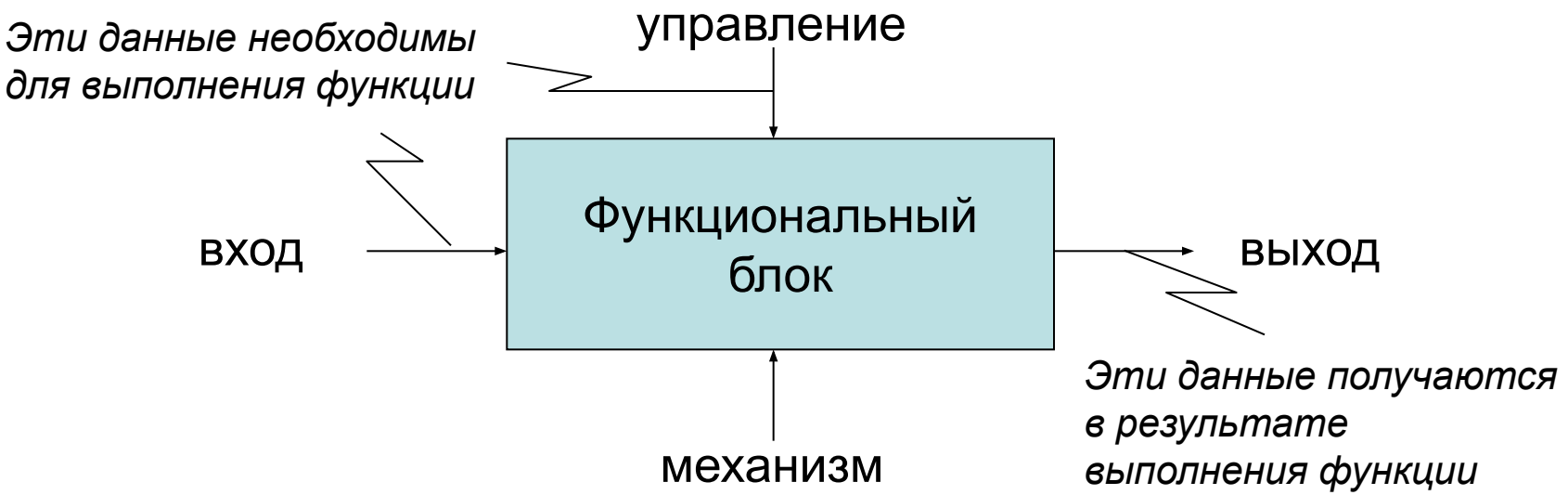
С точки зрения SADT модель может основываться либо на функциях системы, либо на ее предметах (планах, данных, оборудовании, информации и т.д.). Соответствующие модели принято называть активностными моделями и моделями данных. Активностная модель представляет с нужной степенью подробности систему активностей, которые в свою очередь отражают свои взаимоотношения через предметы системы.

SADT

Основным рабочим элементом при моделировании является диаграмма. Модель SADT объединяет и организует диаграммы в иерархические древовидные структуры, при этом этом чем выше уровень диаграммы, тем она менее детализирована. В состав диаграммы входят блоки, изображающие активности моделируемой системы, и дуги, связывающие блоки вместе и изображающие взаимодействия и взаимосвязи между блоками. SADT требует, чтобы в диаграмме было 3-6 блоков: в этих пределах диаграммы и модели удобны для чтения, понимания и использования. Вместо одной громоздкой модели используются несколько небольших взаимосвязанных моделей, значения которых взаимодополняют друг друга, делая понятной структуризацию сложного объекта. Однако такое жесткое требование на число блоков на диаграмме ограничивает применение SADT для ряда предметных областей. Например, в банковских структурах имеется 15-20 равноправных деятельностей, которые целесообразно отразить на одной диаграмме. Искусственное их растаскивание по разным уровням SADT-модели явно не улучшает ее понимаемость.

SADT

Блоки на диаграммах изображаются прямоугольниками и сопровождаются текстами на естественном языке, описывающими активности. В отличие от других методов структурного анализа в SADT каждая сторона блока имеет вполне определенное особое назначение: левая сторона блока предназначена для ***Входов***, верхняя - для ***Управления***, правая - для ***Выходов***, нижняя - для ***Исполнителей***. Такое обозначение отражает определенные принципы активности: *Входы* преобразуются в *Выходы*, *Управления* ограничивают или предписывают условия выполнения, *Исполнители* описывают, за счет чего выполняются преобразования



SADT

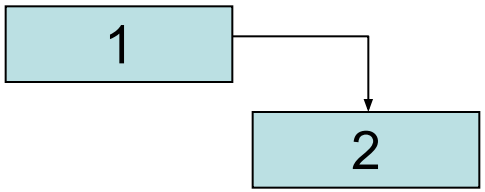
Дуги в SADT представляют наборы предметов и маркируются текстами на естественном языке. Предметы могут состоять с активностями в четырех возможных отношениях: *Вход, Выход, Управление, Исполнитель*. Каждое из этих отношений изображается дугой, связанной с определенной стороной блока - таким образом, стороны блока чисто графически сортируют предметы, изображаемые дугами. Входные дуги изображают предметы, используемые и преобразуемые активностями. Управляющие дуги обычно изображают информацию, управляющую действиями активностей. Выходные дуги изображают предметы, в которые преобразуются входы. Исполнительские дуги отражают (по крайней мере частично) реализацию активностей.

Блоки на диаграмме размещаются по "ступенчатой" схеме в соответствии с их доминированием, которое понимается как влияние, оказываемое одним блоком на другие. Кроме того, блоки должны быть пронумерованы, например, в соответствии с их доминированием. Номера блоков служат однозначными идентификаторами для активностей и автоматически организуют эти активности в иерархию модели.

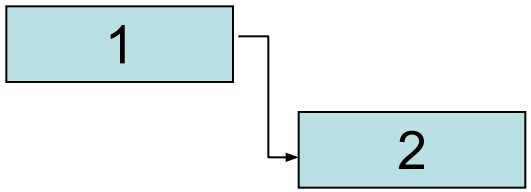
SADT

Взаимовлияние блоков может выражаться либо в пересылке *Выхода* к другой активности для дальнейшего преобразования, либо в выработке управляющей информации, предписывающей, что именно должна делать другая активность. Таким образом, диаграммы SADT являются предписывающими диаграммами, описывающими как преобразования между *Входом* и *Выходом*, так и предписывающие правила этих преобразований.

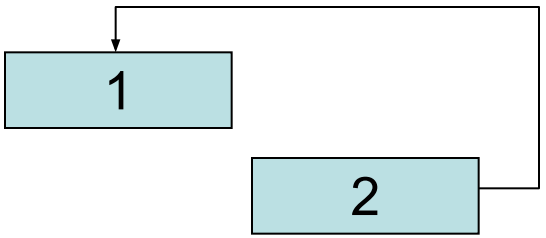
В SADT требуются только пять типов взаимосвязей между блоками для описания их отношений:



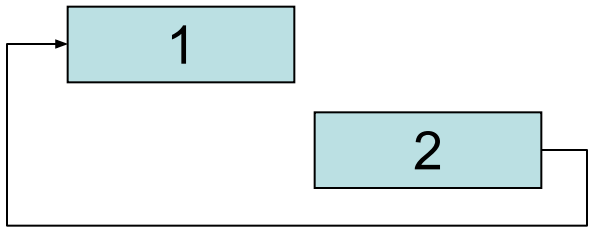
*а) отношение **управление***



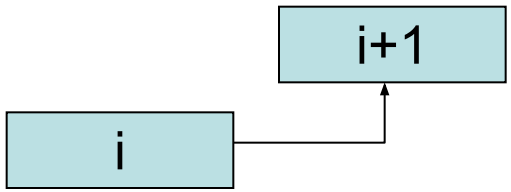
*б) отношение **вход***



*в) отношение **обратная связь по управлению***



*г) отношение **обратная связь по входу***



*д) отношение **выход-механизм***

SADT

Отношения *Управления* и *Входа* являются простейшими, поскольку они отражают интуитивно очевидные прямые воздействия. Отношение *Управления* возникает тогда, когда *Выход* одного блока непосредственно влияет на блок с меньшим доминированием. Отношение *Входа* возникает, когда *Выход* одного блока становится *Входом* для блока с меньшим доминированием. Обратные связи более сложны, поскольку они отражают итерацию или рекурсию - *Выходы* из одной активности влияют на будущее выполнение других функций, что впоследствии влияет на исходную активность. *Управленческая Обратная Связь* возникает, когда *Выход* некоторого блока влияет на блок с большим доминированием, а отношение *Входной Обратной Связи* имеет место, когда *Выход* одного блока становится *Входом* другого блока с большим доминированием. Отношения *Выход - Исполнитель* встречаются нечасто и представляют особый интерес. Они отражают ситуацию, при которой *Выход* одной активности становится средством достижения цели другой активностью

SADT

Пример SADT-диаграммы

SADT

Дуги SADT, как правило, изображают наборы предметов, поэтому они могут разветвляться и соединяться вместе различным образом. Разветвления дуги означают, что часть ее содержимого (или весь набор предметов) может появиться в каждом ответвлении дуги. Дуга всегда помечается до разветвления, чтобы дать название всему набору. Кроме того, каждая ветвь дуги может быть помечена в соответствии со следующими правилами: считается, что непомеченная ветвь содержит все предметы, указанные в метке перед разветвлением; каждая метка ветви уточняет, что именно содержит эта ветвь. Слияние дуг указывает, что содержимое каждой ветви участвует в формировании после слияния объединенной дуги. После слияния дуга всегда помечается для указания нового набора, кроме того, каждая ветвь перед слиянием может помечаться в соответствии со следующими правилами: считается, что непомеченные ветви содержат все предметы, указанные в общей метке после слияния; каждая метка ветви уточняет, что именно содержит эта ветвь.

SADT

При создании модели одна и та же диаграмма чертится несколько раз, что создает различные ее варианты. Чтобы различать различные версии одной и той же диаграммы, в SADT используется схема контроля конфигурации диаграмм, основанная на их номерах. Если диаграмма замещает более старый вариант, то предыдущий номер помещается в скобках для указания связи с предыдущей работой.

SADT, как и другие методологии проектирования, целесообразно использовать на ранних этапах ЖЦ: для понимания системы до ее воплощения. SADT позволяет сократить дорогостоящие ошибки на ранних этапах создания системы, улучшить контакт между пользователями и разработчиками, сгладить переход от анализа к проектированию. Несомненное достоинство SADT заключается в том, что она легко отражает такие характеристики как управление, обратная связь и исполнители.