ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1.В.ДВ.09.02 Основы компьютерного проектирования и моделирования АСОИ

Направление подготовки *09.03.01 Информатика и вычислительная техника*

Профиль образовательной программы *Автоматизированные системы обработки информации и управления*

Форма обучения *заочная*

1. КОНСПЕКТ ЛЕКЦИЙ

1.1 Лекция № 1 (2 часа)

Тема: «Общие принципы проектирования информационных систем**»**

1.1.1 Вопросы лекции:

1. Объект проектирования.
2. Процесс проектирования.
3. Формальная логика.
4. Процесс образования понятия.
5. Диалектическая логика.
6. Содержательно-генетическая логика.
7. Логическая схема проектирования.
8. Логические отношения.
9. Определение задач проектирования.
10. Определение логической схемы проектирования (методологии проектирования).
11. Решение задач проектирования.
12. Понятие системного подхода и системного анализа.
13. Документ. Электронный документ. Информационная система.
14. Информационная технология.

1.1.2 Краткое содержание вопросов:

1. Объект проектирования.

Различные виды проектирования ориентированы на создание и преобразование разных объектов и предметов. *Объект проектирования* — это среда или процесс, в контексте которых находится предмет. *Предмет проектирования* — это предполагаемый продукт, образ которого первоначально представлен в проекте. Это то, созданию чего посвящена проектная деятельность. Объект и предмет проектирования соотносятся между собой как общее и частное.

2. Процесс проектирования.

Проектирование—это комплекс работ с целью получения описаний нового или модернизируемого технического объекта, достаточных для реализации или изготовления объекта в заданных условиях. Объектами проектирования могут быть изделия (например, обрабатывающий центр, двигатель внутреннего сгорания, ЭВМ) или процессы (например, технологические, вычислительные). Комплекс проектных работ включает в себя теоретические и экспериментальные исследования, расчеты, конструирование.

Проектирование, осуществляемое человеком при взаимодействии с ЭВМ, называют *автоматизированным.* Степень автоматизации может быть различной и оценивается долей s проектных работ, выполняемых на ЭВМ без участия человека. При s=0 проектирование называют *неавтоматизированным,* при s=1—*автоматическим.*

Автоматизированное проектирование осуществляется в рамках САПР. В соответствии с ГОСТом *система автоматизированного проектирования*—это организационно-техническая система, состоящая из комплекса средств автоматизации проектирования (АП), взаимодействующего с подразделениями проектной организации, и выполняющая автоматизированное проектирование.

Проектирование делится на стадии, этапы и процедуры. При проектировании сложных объектов выделяют стадии

- научно-исследовательских работ (НИР)
- опытно-конструкторских работ (ОКР)
- технического проекта
- рабочего проекта
- испытаний опытного образца.

Стадию *НИР* во многих случаях можно разделить на стадии

- предпроектных исследований
- технического задания
- технического предложения.

На этих стадиях последовательно изучаются потребности в получении новых изделий с заданным целевым назначением, исследуются физические, информационные, конструктивные и технологические принципы построения изделий. Далее исследуются возможности реализации этих принципов, прогнозируются возможные значения характеристик и параметров объектов. *Результатом НИР является формулировка технического задания (ТЗ) на разработку нового объекта.*

На стадии *ОКР* разрабатывается эскизный проект изделия, проверяются, конкретизируются и корректируются принципы и положения, установленные на стадии НИР.

На стадии *технического проекта* принимаются подробные технические решения и прорабатываются все части проекта.

На стадии *рабочего проекта* создается полный комплект конструкторско-технологической документации, достаточный для изготовления объекта.

На стадии *испытаний опытного образца* (или пробной партии при крупносерийном производстве) получают результаты, позволяющие выявить возможные ошибки и недоработки проекта, принимаются меры к их устранению, после чего документация передается на предприятия, выделенные для серийного производства изделий.

Проектирование разделяется также на этапы. Используются при этом следующие понятия. *Проектное решение*—описание объекта или его составной части, достаточное для рассмотрения и принятия заключения об окончании проектирования или путях его продолжения. *Проектная процедура*—часть проектирования, заканчивающаяся получением проектного решения. Примерами проектных процедур служат синтез функциональной схемы устройства, оптимизация параметров функционального узла, трассировка межсоединений на печатной плате и т. п. *Этап проектирования* —это условно выделенная часть проектирования, сводящаяся к выполнению одной или нескольких проектных процедур, объединенных по признаку принадлежности получаемых проектных решений к одному иерархическому уровню и (или) аспекту описаний.

На любой стадии или этапе проектирования можно выявить ошибочность или неоптимальность ранее принятых решений и, следовательно, необходимость или целесообразность их пересмотра. Подобные возвраты характерны для проектирования и обусловливают его *итерационный характер*. Может быть также выявлена необходимость корректировки ТЗ. Вводят понятия *процедур внешнего и внутреннего проектирования.* К внешнему проектированию относят процедуры формирования или корректировки технического задания, а к внутреннему—процедуры реализации сформированного ТЗ. Тогда можно сказать, что происходит чередование процедур внешнего и внутреннего проектирования, что особенно характерно для ранних стадий (НИР, ОКР). При этом различают нисходящее (сверху вниз) и восходящее (снизу вверх) проектирование. При *нисходящем проектировании* задачи высоких иерархических уровней решаются прежде, чем задачи более низких иерархических уровней. При *восходящем проектировании* последовательность противоположная. Функциональное проектирование сложных систем чаще всего является нисходящим, конструкторское проектирование—восходящим.

3. Формальная логика.

Формальная логика — конструирование и исследование правил преобразования высказываний, сохраняющих их истинностное значение безотносительно к содержанию входящих в эти высказывания понятий. Формальная логика, в отличие

от неформальной, организована как формальная система, обладающая высоким уровнем абстракции и чётко определёнными методами, правилами и законами[1]. Формальная логика как наука занимается выводом нового знания на основе ранее известного без обращения в каждом конкретном случае к опыту, а применением законов и правил мышления. Начальной ступенью формальной логики можно считать традиционную логику, а математическую логику — её следующей ступенью, использующей математические методы, символический аппарат и логические исчисления

4. Процесс образования понятия.

Простейшей категорией формальной логики является **понятие - оно *фиксирует мысль о предмете.*** Обычно понятие определяется через более широкое понятие путем добавления к родовому признаку видового различия.

Для человека, занимающегося научными изысканиями, постоянно необходимо получать новую информацию. Для этого ученый читает множество литературы по избранному предмету, ведет наблюдение, делает опыты. Однако вся эта деятельность была бы бесполезной, если бы не приводила к образованию новых понятий. Иными словами, полученная информация в таком случае так и осталась бы лишь информацией, не облеченной в форму, пригодную для закрепления и передачи.

Именно поэтому необходимо знать о приемах образования понятий. Такими приемами являются: абстрагирование, анализ, синтез, сравнение и обобщение.

Абстрагирование — это прием образования понятий, при котором необходимо отвлечься от ряда несущественных признаков предмета, отринуть их и оставить лишь существенные.

В процессе абстрагирования значительную роль играет сравнение.

Анализ — это мысленное дробление предмета, процесса или явления на составные части с целью установления взаимодействия этих частей и взаимосвязей между ними, а также выявления происходящих внутри исследуемого объекта процессов.

Анализ необходим для получения отражения уже существующего понятия.

Синтез — это мысленная сборка составных частей предмета, явления или процесса воедино.

Синтез — это процесс, обратный анализу, и обычно используется, когда последний уже проведен. Зачастую мысленному синтезу предшествует, если речь идет о предмете, практическая сборка данного предмета со строгим соблюдением последовательности постановки составных частей.

Синтез применяется для создания новых понятий на основе уже существующих, подвергнутых синтезу, или выявления неточностей в понятии, а также внесения в эти понятия изменений.

Сравнение — это мысленное установление сходства или различия предметов по существенным или несущественным признакам.

Обобщение — мысленное объединение группы предметов в новый ряд или добавление одного предмета в уже существующий на основе присущих этим предметам признаков.

Сравнение и обобщение позволяют достичь большей точности в суждениях, отделить одно от другого или, наоборот, объединить несколько предметов в одну группу (класс). Как факультативный признак, способствуют лучшему усвоению информации человеческим мозгом.

Все логические приемы образования понятий имеют важнейшее значение. Они связаны между собой, их невозможно представить один без другого. Часто применяются вместе или предшествуют один другому.

5. Диалектическая логика.

Диалектическая логика — философский раздел марксизма. В широком смысле понималась как систематически развёрнутое изложение диалектики мышления:диалектическое изложение науки о научно-теоретическом

мышлении («диалектики как логики»), которая тем самым является и научной теорией познания объективного мира. В узком смысле понималась как логическая дисциплина о формах правильных рассуждений.

Предмет диалектической логики — мышление. Диалектическая логика имела своей целью развернуть его изображение в необходимых его моментах и притом в независящей ни от воли, ни от сознания последовательности, а также утвердить свой статус как логической дисциплины.

6. Содержательно-генетическая логика.

СОДЕРЖАТЕЛЬНО-ГЕНЕТИЧЕСКАЯ ЛОГИКА - логико-эпистемологическая концепция, которая разрабатывалась в 1950-1960 в Московском логическом (в дальнейшем - методологическом) кружке.

Такая логика полагалась содержательной в противовес формальной. С другой стороны, логическая разработка диалектики в советской философии оценивалась еще более негативно и критически. Вопреки всей европейской традиции, логика рассматривалась как эмпирическая наука. Предметом эмпирических исследований объявлялись способы мышления, зафиксированные в научных текстах. Таким образом, основной эмпирией исследования становился текст. Тезис об эмпиричности был вместе с тем способом преодоления схоластичности существовавшей диалектики и нормативной бессодержательности формальной логики.

В целом концепция С.-Г.Л., выдвинутая Щедровицким, представляла собой разработку трех основных положений:

1) В основе концепции лежала гипотеза о "двух-плоскостной" структуре мышления, включающей плоскость "объективного содержания" и плоскость "знаковой формы". Формальная логика, начиная с Аристотеля, строилась на основе принципа параллелизма формы и содержания мышления, т.е. на основе предположения, что а) каждому элементу знаковой формы языковых выражений соответствует определенный субстанциальный элемент содержания и б) способ связи элементов содержания в точности соответствует способу связи элементов знаковой формы. На основе этого принципа в формальной логике производилась редукция содержания к плоскости знаковой формы. Такой прием ограничивал реальное содержание аристотелевской логики атрибутивными свойствами объектов. Другие типы содержания, не сводящиеся к атрибутивным свойствам, не могли быть в ней адекватно представлены. В противовес этому, С.-Г.Л. выдвигала принцип непараллелизма формы и содержания. Реализация этого принципа требовала описывать и изображать плоскость содержания (оперирование с объектами) отдельно от плоскости знаковой формы, а также описывать механизм связи между содержанием и формой. Таким образом, предметом критики выступала не "формальность" классической логики сама по себе (т.е. ее структурность и нормативность), а редукция содержания мышления к якобы "всеобщим" и "универсальным" логическим формам.

2) В качестве основного объяснительного принципа по отношению к мышлению выдвигалась категория "деятельности". Само мышление рассматривалось двояко: как фиксированное знание и как деятельность по его получению. Принцип деятельности требовал рассматривать мышление не по содержанию предметного сознания, а структурно-процессуально, т.е. исследовать и описывать, с одной стороны, процедуры и операции мышления, а с другой - типологически обобщенную структуру знаний. Таким образом, концепция С.-Г.Л. смыкалась с программой исследования мышления как деятельности. Деятельностный подход позволял сохранить установку на содержательность, избегая психологизма: содержание сознания выносилось "за скобки", а деятельностное содержание трактовалось операционально. "Воспроизвести содержание мышления" означало воспроизвести схему оперирования с объектами, схему их сопоставлений и т.п. Соответственно этому отвергалась и трактовка мышления как "отражения". Основной мыслительной связью объявлялось не отражение, а "замещение" исходного практического оперирования с объектами знаками и оперированием с ними.

Мышление представлялось как деятельность в иерархической структуре последовательно надстраивающихся друг над другом плоскостей знакового замещения, каждая из которых задавала определенные системы оперирования.

3) Общая теоретическая задача С.-Г.Л. состояла в том, чтобы на основе анализа единичных эмпирически заданных текстов проанализировать и воспроизвести в форме "исторической теории" мышление вообще, мышление как таковое, как один органический предмет. При этом мышление изначально рассматривалось в синтетическом единстве с языком, как языковое мышление. Раздельный анализ мышления и языка признавался неэффективным. Генетичность новой логики определялась, с одной стороны, ориентацией на исследование процессов развития мышления, а с другой стороны, использованием метода восхождения от абстрактного к конкретному.

7. Логическая схема проектирования.

Проектирование – процесс, позволяющий провести некую техническую идею до её инженерной модели. Результатом этого процесса является проект, который представляет из себя, как правило, графическую часть (чертежи, схемы) и пояснительную записку (описание назначения изделия, функции, технические характеристики и т.д.).

Алгоритм проектирования – совокупность предписаний, необходимых для выполнения проектирования. Алгоритм проектирования может быть общим (для определенного класса объектов) и специальным (для одного объекта). Под выполнением проектирования понимается нахождение результата проектирования.

Результат проектирования – проектное решение (совокупность проектных решений), удовлетворяющее заданным требованиям, необходимое для создания объекта проектирования. В заданные требования должны быть включены требования к форме представления проектного решения.

Проектное решение – промежуточное или конечное описание объекта проектирования, необходимое и достаточное для рассмотрения и определения дальнейшего направления или окончания проектирования.

Типовое проектное решение – уже существующее проектное решение, используемое при проектировании.

Цель процесса проектирования состоит, прежде всего, в том, чтобы на основе исходной информации, получаемой в процессе проектирования, разработать техническую документацию для изготовления объекта проектирования. Проектирование включает в себя разработку технического задания (ТЗ), отражающего потребности, и реализацию ТЗ в виде проектной документации.

Проектирование, по существу, представляет собой процесс управления с обратной связью. Техническое задание формирует входы, которые сравниваются с результатами проектирования, и если они не совпадают, цикл проектирования повторяется вновь до тех пор, пока отклонение от заданных технических требований не окажется в допустимых пределах.

Проектная процедура соответствует части проектной подсистемы, в результате выполнения которой принимается некоторое проектное решение. Она состоит из элементарных проектных операции, имеет твердо установленный порядок их выполнения и направлена на достижение локальной цели в процессе проектирования.

Под проектной операцией понимают условно выделенную часть проектной процедуры или элементарное действие, совершаемое конструктором в процессе проектирования. Примерами проектных процедур могут служить процедуры разработки кинематической или компоновочной схемы станка, технологии обработки изделий и т.п., а примерами проектных операций – расчет припусков, решение какого-либо уравнения и т.п.

8. Логические отношения.

Так как все предметы находятся во взаимодействии и взаимообусловленности, то и понятия, отражающие данные предметы, также находятся в определенных отношениях.

Конкретные виды отношений устанавливаются в зависимости от содержания и объема понятий, которые сравниваются.

Если понятия не имеют общих признаков, далеки друг от друга по свое-му содержанию, то они называются несравнимыми. Например, «симфониче-ская музыка» и «кассационная жалоба», «процессуальные акты предварительного расследования» и «общая тетрадь».

Сравнимыми называются понятия, отражающие некоторые общие суще-ственные признаки предмета или класса однородных предметов. Например, «юрист» и «адвокат», «взятка» и «кража».

В логических отношениях могут находиться только сравнимые понятия. В зависимости от того, как соотносятся их объемы, понятия делятся на две группы: совместимые и несовместимые.

Совместимые - это такие понятия, объемы которых совпадают полно-стью или частично. Несовместимые - это понятия, объемы которых не сов-падают ни в одном элементе, но которые могут быть включены частично или полностью в объем общего для них понятия. На представленной схеме по-казаны виды совместимых и несовместимых понятий.

Отношения между понятиями принято иллюстрировать при помощи кругов Эйлера (круговых схем).

В отношениях равнозначности находятся совместимые понятия, объемы которых полностью совпадают. В таких понятиях мыслится один и тот же предмет или класс однородных предметов. Однако содержание этих понятий различно, так как каждое из них отражает только определенную сторону (существенный признак) данного предмета или класса однородных предметов.

В отношении пересечения находятся совместимые понятия, у которых объемы частично совпадают. Частично совпадает и содержание данных понятий.

В отношении подчинения находятся совместимые понятия, объем одно-го из которых полностью входит в объем другого, составляя его часть.

Объем первого понятия шире объема второго понятия: кроме кражи личного имущества граждан в него входит также кража государственного, кооперативного имущества.

Из двух понятий, находящихся в отношении подчинения, понятие с большим объемом (подчиняющее) является родовым, или родом по отношению к понятию с меньшим объемом (подчиненному), а последнее по отношению к первому называется видовым, или видом. Родовидовые отноше-ния лежат в основе логических операций ограничения и обобщения понятий, деления объема понятий и некоторых видов определения.

Перейдем к рассмотрению несовместимых понятий.

При иллюстрации отношений между несовместимыми понятиями возни-кает потребность во введении более широкого по объему понятия, которое включало бы объемы несовместимых понятий.

В отношении соподчинения находятся два или более непересекающихся понятий, принадлежащих общему родо-вому понятию.

Соподчиненные понятия В и С - это виды одного рода А, у них общий родовой признак, но видовые признаки различны.

В отношении противоположности находятся понятия, которые являются видами одного и того же рода, и при этом одно из них содержит какие-то признаки, а другое эти признаки отрицает и заменяет противоположными при-знаками.

В отношении противоречия находятся такие два поня-тия, которые являются видами одного и того же рода, и при этом одно понятие указывает на некоторые признаки, а другое эти признаки отрицает, исключает, не заменяя их никакими другими признаками.

9. Определение задач проектирования.

На стадии проектирования проектировщик должен проделать следующую работу:

1. Обследовать предметную область автоматизации.

2. Определить объекты и перечень их атрибутов, для каждого объекта выделить первичные ключи и провести нормализацию.

3. Установить все связи между объектами. Начертить схему проекта со всеми объектами и связями.

4. Выработать технологию обслуживания, т.е. определить порядок сбора, хранения данных в БД частоту и форматы ввода-вывода данных, правила работы всех групп пользователей.

5. Выбрать компьютер и инструментальные средства (конкретную СУБД) для реализации.

6. Проверить корректность проекта. Проект должен адекватно, на требуемом уровне детальности, отображать предметную область.

10. Определение логической схемы проектирования (методологии проектирования).

Методология логического проектирования

Подразделяется на следующие этапы:

1. Построение и проверка локальной логической модели для отдельных представлений каждого из пользователей.

2. Создание и проверка глобальной логической модели данных.

На первом этапе, на основе концептуальной модели данных строится локальная логическая модель. Данный этап включает следующие моменты:

1. Локальная концептуальная модель преобразуется в локальную логическую.

2. Исходя из структуры локальной логической модели определяются наборы отношений.

3. Проверка модели с помощью правил нормализации.

4. Проверка модели в отношении транзакции пользователя.

5. Создание уточненной диаграммы “сущность – связь”.

6. Определение требований поддержки целостности данных.

7. Обсуждение разработанной локальной логической модели с конечным пользователем.

11. Решение задач проектирования.

Одним из путей предсказания поведения проектируемых систем является путь создания математических моделей и последующего проведения исследования систем на этих моделях. Построение или проектирование систем, удовлетворяющих заранее заданным свойствам, можно осуществить, когда имеются управляющие переменные, при помощи которых можно влиять на поведение проектируемой системы.

1. Понятие системного подхода и системного анализа.

Основной общий принцип системного подхода заключается в рассмотрении частей явления или сложной системы с учетом их взаимодействия. *Системный подход* включает в себя выявление структуры системы, типизацию связей, определение атрибутов, анализ влияния внешней среды. Системный подход рассматривают как направление научного познания и социальной политики. Он является базой для обобщающей дисциплины «Теория систем» (другое используемое название - «Системный анализ»). *Теория систем* - дисциплина, в которой конкретизируются положения системного подхода; она посвящена исследованию и проектированию сложных экономических, социальных, технических систем, чаще всего слабоструктурированных. Характерными примерами таких систем являются производственные системы. При проектировании систем цели достигаются в

многошаговых процессах принятия решений. Методы принятия решений часто выделяют в самостоятельную дисциплину, называемую «Теория принятия решений».

В технике дисциплину, в которой исследуются сложные технические системы, их проектирование и которая аналогична теории систем, чаще называют *системотехникой.* Предметом системотехники являются, во-первых, организация процесса создания, использования и развития технических систем, во-вторых, методы и принципы их проектирования и исследования. В системотехнике важно уметь сформулировать цели системы и организовать ее рассмотрение с позиций поставленных целей. Тогда можно отбросить малозначимые части при проектировании и моделировании, перейти к постановке оптимизационных задач.

Системы автоматизированного проектирования и управления относятся к числу наиболее сложных современных искусственных систем. Их проектирование и сопровождение невозможны без системного подхода. Поэтому идеи и положения системотехники входят составной частью в дисциплины, посвященные изучению современных автоматизированных систем и технологий их применения.

Интерпретация и конкретизация системного подхода имеют место в ряде известных подходов с другими названиями, которые также можно рассматривать как компоненты системотехники. Таковы структурный, блочно-иерархический, объектно-ориентированный подходы.

При *структурном подходе,* как разновидности системного, требуется синтезировать варианты системы из компонентов (блоков) и оценивать варианты при их частичном переборе с предварительным прогнозированием характеристик компонентов.

Блочно-иерархический подход к проектированию использует идеи декомпозиции сложных описаний объектов и соответственно средств их создания на иерархические уровни и аспекты, вводит понятие стиля проектирования (восходящее и нисходящее), устанавливает связь между параметрами соседних иерархических уровней.

Ряд важных структурных принципов, используемых при разработке информационных систем и прежде всего их программного обеспечения (ПО), выражен в *объектно-ориентированном подходе* к проектированию. Такой подход имеет следующие преимущества в решении проблем управления сложностью и интеграции ПО: 1) вносит в модели приложений большую структурную определенность, распределяя представленные в приложении данные и процедуры между классами объектов; 2) сокращает объем спецификаций благодаря внедрению в описания иерархии объектов и отношений наследования между свойствами объектов разных уровней иерархии; 3) уменьшает вероятность искажения данных вследствие ошибочных действий за счет ограничения доступа к определенным категориям данных в объектах. Описание в каждом классе объектов допустимых обращений к ним и принятый форматов сообщений облегчает согласование и интеграцию ПО.

Для всех подходов к проектированию сложных систем характерны также следующие особенности.

- Структуризация процесса проектирования, выражаемая декомпозицией проектных задач и документации, выделением стадий, этапов, проектных процедур. Эта структуризация является сущностью блочно-иерархического подхода к проектированию.
- Итерационный характер проектирования.
- Типизация и унификация проектных решений и средств проектирования.

Существуют различные точки зрения на содержание понятия «системный анализ» и область его применения. Изучение различных определений системного анализа позволяет выделить четыре его трактовки.

Первая трактовка рассматривает системный анализ как один из конкретных методов выбора лучшего решения возникшей проблемы, отождествляя его, например, с анализом по критерию стоимость — эффективность.

Такая трактовка системного анализа характеризует попытки обобщить наиболее разумные приемы любого анализа (например, военного или экономического), определить общие закономерности его проведения.

В первой трактовке системный анализ — это, скорее, «анализ систем», так как акцент делается на объекте изучения (системе), а не на системности рассмотрения (учете всех важнейших факторов и взаимосвязей, влияющих на решение проблемы, использование определенной логики поиска лучшего решения и т.д.)

В ряде работ, освещающих те или иные проблемы системного анализа, слово «анализ» употребляется с такими прилагательными, как количественный, экономический, ресурсный, а термин «системный анализ» применяется значительно реже.

Согласно второй трактовке системный анализ — это конкретный метод познания (противоположность синтезу).

Третья трактовка рассматривает системный анализ как любой анализ любых систем (иногда добавляется, что анализ на основе системной методологии) без каких-либо дополнительных ограничений на область его применения и используемые методы.

Согласно четвертой трактовке системный анализ — это вполне конкретное теоретико-прикладное направление исследований, основанное на системной методологии и характеризующееся определенными принципами, методами и областью применения. Он включает в свой состав как методы анализа, так и методы синтеза, кратко охарактеризованные нами ранее.

Нам представляется правильной четвертая трактовка8, наиболее адекватно отражающая направленность системного анализа и совокупность используемых им методов.

Итак, системный анализ — это совокупность определенных научных методов и практических приемов решения разнообразных проблем, возникающих во всех сферах целенаправленной деятельности общества, на основе системного подхода и представления объекта исследования в виде системы. Характерным для системного анализа является то, что поиск лучшего решения проблемы начинается с определения и упорядочения целей деятельности системы, при функционировании которой возникла данная проблема. При этом устанавливается соответствие между этими целями, возможными путями решения возникшей проблемы и потребными для этого ресурсами.

Системный анализ характеризуется главным образом упорядоченным, логически обоснованным подходом к исследованию проблем и использованию существующих методов их решения, которые могут быть разработаны в рамках других наук.

Целью системного анализа является полная и всесторонняя проверка различных вариантов действий с точки зрения количественного и качественного сопоставления затраченных ресурсов с получаемым эффектом.

Системный анализ, по существу, является средством установления рамок для систематизированного и более эффективного использования знаний, суждений и интуиции специалистов; он обязывает к определенной дисциплине мышления.

2. Документ. Электронный документ. Информационная система.

Документ является основным способом представления информации. ***Электронный документ*** — это бумажный документ, введённый в*компьютер* для обработки. Финансовые электронные документы могут снабжаться *электронной подписью*. Электронные документы бывают структурированными, и тогда они находятся в базах данных, и неструктурированными, содержащими тексты на естественном языке.

В общем же принято считать под *электронным документом* (ЭД) структурированный информационный *объект*, в соответствие которому может быть поставлена совокупность файлов, хранящихся на накопителе компьютера. Необходимым признаком ЭД является "регистрационная карточка", состоящая из реквизитов документа, содержащих перечень необходимых данных о нём. Согласно законодательству,

электронным является документ, в котором *информация* представлена в электронно-цифровой форме.

Электронные документы можно разделить на два основных типа: неформализованные (произвольные) и служебные (официальные). Неформализованный *электронный документ* — это любое сообщение, записка, текст, записанный на машинном носителе. Под служебным *электронным документом* понимается записанное на машинном носителе электронное сообщение, реквизиты которого оформлены в соответствии с нормативными требованиями.

Электронные документы *по* сравнению с бумажными обладают следующими преимуществами:

- более низкая стоимость и время передачи электронного документа из одного места в другое;
- более низкая стоимость и время тиражирования ЭД;
- более низкая стоимость архивного хранения ЭД;
- возможность контекстного поиска;
- новые возможности защиты документов;
- упрощение подготовки ЭД в сочетании с широкими возможностями;
- принципиально новые возможности представления ЭД. Документ может иметь динамическое содержание (например, аудио-, видеовставки).

Информационная система (сокр. ИС) — система, предназначенная для хранения, поиска и обработки информации и соответствующие организационные ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию.

Информационная система предназначена для своевременного обеспечения надлежащих людей надлежащей информацией, то есть для удовлетворения конкретных информационных потребностей в рамках определенной предметной области, при этом результатом функционирования информационных систем являетсяинформационная продукция — документы, информационные массивы, базы данных и информационные услуги.

Понятие информационной системы интерпретируют по-разному, в зависимости от контекста.

Достаточно широкое понимание информационной системы подразумевает, что её неотъемлемыми компонентами являются данные, техническое и программное обеспечение, а также персонал и организационные мероприятия. Широкотрактует понятие «информационной системы» федеральный закон Российской Федерации«Об информации, информационных технологиях и о защите информации», подразумевая под информационной системой совокупность содержащейся в базах данныхинформации и обеспечивающих её обработку информационных технологий и технических средств[7].

Среди российских ученых в области информатики, наиболее широкое определение ИС дает М. Р. Когаловский, по мнению которого в понятие информационной системы помимо данных, программ, аппаратного обеспечения и людских ресурсов следует также включать коммуникационное оборудование, лингвистические средства и информационные ресурсы, которые в совокупности образуют систему, обеспечивающую «поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей».

Более узкое понимание информационной системы ограничивает её состав данными, программами и аппаратным обеспечением. Интеграция этих компонентов позволяет автоматизировать процессы управления информацией и целенаправленной деятельности конечных пользователей, направленной на получение, модификацию и хранение информации. Так, российский стандарт ГОСТ РВ 51987 подразумевает под ИС «автоматизированную систему, результатом функционирования которой является

представление выходной информации для последующего использования». ГОСТ Р 53622-2009 использует термин*информационно-вычислительная система* для обозначения совокупности данных (или баз данных), систем управления базами данных и прикладных программ, функционирующих на вычислительных средствах как единое целое для решения определенных задач.

В деятельности организации информационная система рассматривается как программное обеспечение, реализующее деловую стратегию организации. При этом хорошей практикой является создание и развертывание единой корпоративной информационной системы, удовлетворяющей информационные потребности всех сотрудников, служб и подразделений организации. Однако на практике создание такой всеобъемлющей информационной системы слишком затруднено или даже невозможно, вследствие чего на предприятии обычно функционируют несколько различных систем, решающих отдельные группы задач: управление производством,финансово-хозяйственная деятельность, электронный документооборот и т. д. Часть задач бывает «покрыта» одновременно несколькими информационными системами, часть задач — вовсе не автоматизирована. Такая ситуация получила название «лоскутной автоматизации» и является довольно типичной для многих предприятий.

3. Информационная технология.

Основной составляющей частью автоматизированной информационной системы является информационная технология (ИТ), развитие которой тесно связано с развитием и функционированием ИС.

Понятие "***технология***"в переводе с греческого означает искусство, мастерство, умение. Технология, как процесс, означает последовательность ряда действий с целью переработки чего-либо. Технологический процесс реализуется различными средствами и методами.

Процесс материального производства предполагает обработку ресурсов с целью получения материальных продуктов (товаров). Если речь идет об информационных технологиях, то роль ресурсов играют данные.

Информационная технология – процесс, использующий совокупность средств методов сбора, обработки и передачи первичной информации для получения информации нового качества о состоянии объекта, т.е. информационного продукта.

Информационный продукт используется, в частности, для принятия решений. Существует *разница между понятиями "информационная система" и "информационная технология"*.

Информационная технология является процессом, состоящим из четко регламентированных операций по преобразованию информации (сбор данных, их регистрация, передача, хранение, обработка, использование).

Компьютерная информационная система является человеко-машинной системой обработки информации с целью организации, хранения и передачи информации. Например, технология, работающая с текстовым редактором, не является информационной системой.

Информационные технологии состоят из этапов, каждый из них включает операции, а последние состоят из элементарных действий, таких как нажатие какой-нибудь клавиши, выбор позиции в меню и т.д.

В информационных технологиях экономических систем широкое распространение получили офисные программы, включающие: табличные процессоры; текстовые процессоры; СУБД; интегрированные пакеты и пр.

Информационные технологии прошли несколько этапов. Каждый этап определяется техникой, программными продуктами, которые используются, т.е. уровнем научно-технического прогресса в этой области.

В настоящее время используется понятие "***новая информационная технология***". Это понятие предполагает:

1. Использование персональных компьютеров и сетей ПК.
2. Наличие коммуникационных средств.
3. Наличие диалоговой (интерактивной) работы с компьютером.
4. Наличие интеграционного подхода.
5. Гибкость процессов изменения данных и постановок задач.
6. Органическое "встраивание" компьютеров в существующую на предприятиях технологию управления.

Основная цель автоматизированной информационной технологии – получать посредством переработки первичных данных информацию нового качества, на основе которой вырабатываются оптимальные управленческие решения. Это достигается за счет интеграции информации, обеспечения ее актуальности и непротиворечивости, использования современных технических средств для внедрения и функционирования качественно новых форм информационной поддержки деятельности аппарата управления.

Информационная технология справляется с существенным увеличением объемов перерабатываемой информации и ведет к сокращению сроков ее обработки. ИТ является наиболее важной составляющей процесса использования информационных ресурсов в управлении. Автоматизированные информационные системы для информационной технологии – это основная среда, составляющими элементами которой являются средства и способы для преобразования данных. ***Информационная технология*** представляет собой процесс, состоящий из четко регламентированных правил выполнения операций над информацией, циркулирующей в ИС.

1.2 Лекция №2 (2 часа)

Тема: «Виды систем проектирования АСОИ**»**

1.2.1 Вопросы лекции:

1. Унифицированный процесс – управляемый вариантами использования.
2. Унифицированный процесс - ориентирован на архитектуру.
3. Унифицированный процесс - итеративный и инкрементный.
4. Жизненный цикл в унифицированном процессе.
5. Продукт унифицированного процесса.
6. Унифицированный процесс – методология разработки.

1.2.2 Краткое содержание вопросов:

1. Унифицированный процесс – управляемый вариантами использования.

Сегодня развитие программного обеспечения происходит в сторону увеличения и усложнения систем. Это связано отчасти с тем, что компьютеры с каждым годом становятся все мощнее, что побуждает пользователей ожидать от них все большего. Эта тенденция также связана с возрастающим использованием Интернета для обмена всеми видами информации. Потребности в отношении более продвинутого программного обеспечения растут по мере того, как мы начинаем понимать с выходом каждого следующего выпуска, как еще можно улучшить этот продукт. Мы желаем иметь программное обеспечение, еще лучше приспособленное для наших нужд, а это, в свою очередь, приводит к усложнению программ.

Мы также желаем получить это программное обеспечение побыстрее. Время выхода на рынок - это другой важный стимул.

Сделать это, однако, нелегко. Наше желание получить мощные и сложные программы не сочетается с тем, как эти программы разрабатываются. Сегодня

большинство людей разрабатывает программы, используя те же методы, что и 25 лет назад, что является серьезной проблемой. Если мы не улучшим наши методы, мы не сможем выполнить свою задачу по разработке так необходимого сегодня сложного программного обеспечения.

Проблема программного обеспечения сводится к затруднениям разработчиков, вынужденных преодолевать в ходе разработки больших программ множество преград. Общество разработчиков программного обеспечения нуждается в управляемом методе работы. Ему нужен процесс, который объединил бы множество аспектов разработки программ. Ему нужен общий подход, который:

- обеспечивал бы руководство деятельностью команды;
- управлял бы задачами отдельного разработчика и команды в целом;
- указывал бы, какие артефакты следует разработать;
- предоставлял бы критерии для отслеживания и измерения продуктов и функционирования проекта.

Унифицированный процесс разработки программного обеспечения - это решение проблемы программного обеспечения.

Прежде всего, Унифицированный процесс есть процесс разработки программного обеспечения. *Процесс разработки программного обеспечения - это сумма различных видов деятельности, необходимых для преобразования требований пользователей в программную систему*. Однако Унифицированный процесс - это больше, чем единичный процесс, это обобщенный каркас процесса, который может быть специализирован для широкого круга программных систем, различных областей применения, уровней компетенции и размеров проекта.

Унифицированный процесс компонентно-ориентирован. Это означает, что создаваемая программная система строится на основе программных компонентов, связанных хорошо определенными интерфейсами.

Для разработки чертежей программной системы Унифицированный процесс использует *Унифицированный язык моделирования*.

Однако действительно специфичные аспекты Унифицированного процесса заключаются в трех словосочетаниях - управляемый вариантами использования, архитектурно-ориентированный, итеративный и инкрементный. Это то, что делает Унифицированный процесс уникальным.

Программная система создается для обслуживания ее пользователей. Следовательно, для построения успешной системы мы должны знать, в чем нуждаются и чего хотят ее будущие пользователи.

Понятие *пользователь* относится не только к людям, работающим с системой, но и к другим системам. Таким образом, понятие *пользователь* относится к кому-либо или чему-либо, что взаимодействует с системой, которую мы разрабатываем. Пример взаимодействия - человек, использующий банкомат. Он вставляет в прорезь пластиковую карту, отвечает на вопрос, высвечиваемый машиной на экране, и получает деньги. В ответ на вставленную карту и ответы пользователя система осуществляет последовательность действий, которые обеспечивают пользователю ощутимый и значимый для него результат, а именно получение наличных.

Взаимодействие такого рода называется вариантом использования. *Вариант использования - это часть функциональности системы, необходимая для получения пользователем значимого для него, ощутимого и измеримого результата.* Варианты использования обеспечивают функциональные требования. Сумма всех вариантов использования составляет модель вариантов использования, которая описывает полную функциональность системы. Однако варианты использования - это не только средство описания требований к системе. Они также направляют ее проектирование, реализацию и тестирование, то есть они направляют процесс разработки. Основываясь на модели вариантов использования, разработчики создают серию моделей проектирования и

реализации, которые осуществляют варианты использования. Тестеры тестируют реализацию для того, чтобы гарантировать, что компоненты модели реализации правильно выполняют варианты использования. Таким образом, варианты использования не только инициируют процесс разработки, но и служат для связи отдельных его частей. *Управляемый вариантами использования* означает, что процесс разработки проходит серии рабочих процессов, порожденных вариантами использования.

2. Унифицированный процесс - ориентирован на архитектуру.

Понятие архитектуры программы включает в себя наиболее важные статические и динамические аспекты системы. Архитектура вырастает из требований к результату в том виде, как их понимают пользователи и другие заинтересованные лица. Эти требования отражаются в вариантах использования. Однако они также зависят от множества других факторов, таких как выбор платформы для работы программы (то есть компьютерной архитектуры, операционной системы, СУБД, сетевых протоколов), доступность готовых блоков многократного использования, соображения развертывания, унаследованные системы и нефункциональные требования. Архитектура - это представление всего проекта с выделением важных характеристик и затушевыванием деталей. Процесс помогает архитектору сконцентрироваться на правильных целях, таких, как понятность, легкость внесения изменений и возможность повторного использования.

Каждый продукт имеет функции и форму. Одно без другого не существует. В удачном продукте эти две стороны должны быть уравновешены. В этом примере функции соответствуют вариантам использования, а форма - архитектуре. Мы нуждаемся во взаимодействии между вариантами использования и архитектурой. Это вариант традиционной проблемы "курицы и яйца". Реально архитектура и варианты использования разрабатываются параллельно.

Таким образом, архитектор придает системе форму. Это означает, что форма, архитектура, должна быть спроектирована так, чтобы позволить системе развиваться не только в момент начальной разработки, но и в будущих поколениях. Чтобы найти такую форму, архитектор должен работать, полностью понимая ключевые функции, то есть ключевые варианты использования системы. Эти ключевые варианты использования составляют от 5 до 10 % всех вариантов использования и крайне важны, поскольку содержат функции ядра системы. Проще говоря, архитектор совершает следующие шаги:

- Создает грубый набросок архитектуры, начиная с той части архитектуры, которая не связана с вариантами использования.
- Далее архитектор работает с подмножеством выделенных вариантов использования, каждый из которых соответствует одной из ключевых функций разрабатываемой системы. Каждый из выбранных вариантов использования детально описывается и реа-лизуется в понятиях подсистем, классов и компонентов.
- После того как варианты использования описаны и полностью разработаны, большая часть архитектуры исследована, созданная архитектура, в свою очередь, будет базой для полной разработки других вариантов использования.

Этот процесс продолжается до тех пор, пока архитектура не будет признана стабильной.

3. Унифицированный процесс - итеративный и инкрементный.

Разработка коммерческих программных продуктов - это серьезное предприятие, которое может продолжаться от нескольких месяцев до года и более. Практично было бы разделить работу на небольшие куски или мини-проекты. Каждый мини-проект является итерацией, результатом которой будет приращение. Итерации относятся к шагам рабочих процессов, а приращение - к выполнению проекта. Для максимальной эффективности

итерации должны быть управляемыми, то есть они должны выбираться и выполняться по плану. Поэтому их можно считать минипроектами.

Разработчики выбирают задачи, которые должны быть решены в ходе итерации, под воздействием двух факторов. Во-первых, в ходе итерации следует работать с группой вариантов использования, которая повышает применимость продукта в ходе дальнейшей разработки. Во-вторых, в ходе итерации следует заниматься наиболее серьезными рисками. Последовательные итерации используют артефакты разработки в том виде, в котором они остались при окончании предыдущей итерации. Это минипроекты, поскольку для выбранных вариантов использования проводится последовательная разработка - анализ, проектирование, реализация и тестирование, и в результате варианты использования, которые разрабатывались в ходе итерации, представляются в виде исполняемого кода.

На каждой итерации разработчики определяют и описывают уместные варианты использования, создают проект, использующий выбранную архитектуру в качестве направляющей, реализуют проект в компоненты и проверяют соответствие компонентов вариантам использования. Если итерация достигла своей цели, процесс разработки переходит на следующую итерацию. Если итерация не выполнила своей задачи, разработчики должны пересмотреть свои решения и попробовать другой подход.

Для получения максимальной экономии команда, работающая над проектом, должна выбирать только те итерации, которые требуются для выполнения цели проекта. Для этого следует расположить итерации в логической последовательности. Разумеется, до определенного предела; так, незамеченные ранее проблемы приведут к добавлению итераций или изменению их последовательности, и процесс разработки потребует больших усилий и времени. Минимизация незамеченных проблем является одной из целей снижения рисков.

Управляемый итеративный процесс имеет множество преимуществ.

- Управляемая итерация ограничивает финансовые риски затратами на одно приращение. Если разработчикам нужно повторить итерацию, организация теряет только усилия, затраченные на одну итерацию, а не стоимость всего продукта.
- Управляемая итерация снижает риск непоставки продукта на рынок в запланированные сроки. При раннем обнаружении соответствующего риска время, которое тратится на его нейтрализацию, вносится в план на ранних стадиях, когда сотрудники менее загружены, чем в конце планового периода.
- Управляемая итерация ускоряет темпы процесса разработки в целом, поскольку для более эффективной работы разработчиков и быстрого получения ими хороших результатов короткий и четкий план предпочтительнее длинного и вечно сдвигающегося.
- Управляемая итерация признает тот факт, что желания пользователей и связанные с ними требования не могут быть определены в начале разработки. Они обычно уточняются в последовательных итерациях.

Эти концепции - управляемая вариантами использования, архитектурно-ориентированная и итеративная и инкрементная разработка - одинаково важны. Архитектура предоставляет нам структуру, направляющую нашу работу в итерациях, в каждой из которых варианты использования определяют цели и направляют нашу работу.

1. Жизненный цикл в унифицированном процессе.

Унифицированный процесс циклически повторяется. Эта последовательность повторений Унифицированного процесса представляет собой жизненный цикл системы. Каждый цикл завершается поставкой выпуска продукта заказчикам.

Каждый цикл состоит из четырех фаз - анализа и планирования требований, проектирования, построения и внедрения. Каждая фаза, как будет рассмотрено ниже, далее подразделяется на итерации.

Каждый цикл осуществляется в течение некоторого времени. Это время, в свою очередь, делится на четыре фазы: фазу анализа и планирования требований, фазу проектирования, фазу построения и фазу внедрения. Внутри каждой фазы руководители или разработчики могут потерпеть неудачу в работе - но только на данной итерации и в связанном с ней приращении. Каждая фаза заканчивается вехой. Мы определяем каждую веху по наличию определенного набора артефактов, например, некая модель документа должна быть приведена в предписанное состояние.

Вехи служат многим целям. Наиболее важная из них - дать руководителям возможность принять некоторые критические решения перед тем, как работа перейдет на следующую фазу. Вехи также дают руководству и самим разработчикам возможность отслеживать прогресс в работе при проходах этих четырех ключевых точек.

В ходе фазы анализа и планирования требований хорошая идея превращается в концепцию готового продукта и создается бизнесплан разработки этого продукта. В частности, на этой фазе должны быть получены ответы на вопросы:

- Что система должна делать в первую очередь для ее основных пользователей?
- Как должна выглядеть архитектура системы?
- Каков план и во что обойдется разработка продукта?

Упрощенная модель вариантов использования, содержащая наиболее критичные варианты использования, дает ответ на первый вопрос. На этом этапе создается пробный вариант архитектуры. Обычно он представляет собой набросок, содержащий наиболее важные подсистемы. На этой фазе выявляются и расставляются по приоритетности наиболее важные риски, детально планируется фаза проектирования и грубо оценивается весь проект.

В ходе фазы проектирования детально описываются большинство вариантов использования и разрабатывается архитектура системы.

Архитектура определяется в виде представлений всех моделей системы, которые в сумме представляют систему целиком. Это означает, что существуют архитектурные представления модели вариантов использования, модели анализа, модели проектирования, модели реализации и модели развертывания. Представление модели реализации включает в себя компоненты для доказательства того, что архитектура выполнима. В ходе этой фазы определяются наиболее критичные варианты использования. Результатом выполнения этой фазы является базовый уровень архитектуры.

В конце фазы проектирования менеджер проекта занимается планированием действий и подсчетом ресурсов, необходимых для завершения проекта. Ключевым вопросом в этот момент будет следующий: достаточно ли проработаны варианты использования, архитектура и план и взяты ли риски под контроль настолько, чтобы можно было давать контрактные обязательства выполнить всю работу по разработке?

В ходе фазы построения происходит создание продукта - к скелету (архитектуре) добавляются мышцы (законченные программы). На этой фазе базовый уровень архитектуры разрастается до полной развитой системы. Концепции развиваются до продукта, готового к передаче пользователям. В ходе фазы объем требуемых ресурсов вырастает. Архитектура системы стабильна, однако, поскольку разработчики могут найти лучшие способы структурирования системы, от них могут исходить предложения о внесении в архитектуру системы небольших изменений. В конце этой фазы продукт включает в себя все варианты использования, которые руководство и заказчик договорились включить в текущий выпуск. Правда, они могут содержать ошибки. Большинство дефектов будут обнаружены и исправлены в ходе фазы внедрения. Ключевой вопрос окончания фазы: удовлетворяет ли продукт требованиям пользователей настолько, что некоторым заказчикам можно делать предварительную поставку?

Фаза внедрения охватывает период, в ходе которого продукт существует в виде бета-выпуска или бета-версии. Небольшое число квалифицированных пользователей,

работая с бета-выпуском продукта, сообщает об обнаруженных дефектах и недостатках. После этого разработчики исправляют обнаруженные ошибки и вносят некоторые из предложенных улучшений в главный выпуск, подготавливаемый для широкого распространения. Фаза внедрения включает в себя такие действия, как производство тиража, тренинг сотрудников заказчика, организацию поддержки по горячей линии и исправление дефектов, обнаруженных после поставки.

Итогом проекта по разработке программного обеспечения является продукт, в создании которого принимает участие множество различных людей.

2. Продукт унифицированного процесса.

Результатом каждого цикла является новый выпуск системы, а каждый выпуск - это продукт, готовый к поставке. Он включает в себя тело - исходный код, воплощенный в компоненты, которые могут быть откомпилированы и выполнены, плюс руководство и дополнительные компоненты поставки. Однако готовый продукт должен также быть приспособлен для нужд не только пользователей, а всех заинтересованных лиц. Программному продукту следовало бы представлять собой нечто большее, чем исполняемый машинный код.

Окончательный продукт включает в себя требования, варианты использования, нефункциональные требования и варианты тестирования. Он включает архитектуру и визуальные модели - артефакты, смоделированные на Унифицированном языке моделирования. Эти средства позволяют заинтересованным лицам использовать систему и модифицировать ее от поколения к поколению.

Даже если исполняемые компоненты с точки зрения пользователей являются важнейшим артефактом, их одних недостаточно. Системное окружение меняется. Операционные системы, СУБД и сами компьютеры прогрессируют. По мере того как цель понимается все лучше, изменяются и требования. В конце концов, разработчики вынуждены начинать новый цикл, а руководство вынуждено их финансировать. Для того чтобы эффективно выполнять новый цикл, разработчикам необходимо иметь полное представление о программном продукте:

- модель вариантов использования со всеми вариантами использования и их связями с пользователями;
- модель анализа, которая имеет две цели - уточнить детали вариантов использования и создать первичное распределение поведения системы по набору объектов, предоставляющих различные варианты поведения;
- модель проектирования, которая определяет статическую структуру системы, такую как подсистемы, классы и интерфейсы, и варианты использования, реализованные в виде кооперации между подсистемами, классами и интерфейсами;
- модель реализации, которая включает в себя компоненты (представленные исходным кодом) и раскладку классов по компонентам;
- модель развертывания, которая определяет физические компьютеры - узлы сети и раскладку компонентов по этим узлам;
- модель тестирования, которая описывает варианты тестов для проверки вариантов использования;
- и, разумеется, представление архитектуры.

Все эти модели связаны. Вместе они полностью описывают систему. Элементы одной модели имеют трассировочные зависимости вперед и назад, организуемые с помощью связей с другими моделями. Например, вариант использования может быть оттрассирован на соответствующую реализацию варианта использования в модели проектирования и вариант тестирования в модели тестирования.

3. Унифицированный процесс – методология разработки.

Унифицированный процесс как процесс разработки программного обеспечения представляет собой методологию, содержащую детальное описание работ по созданию и внедрению ПО (авторы Айвар Якобсон, Грэди Буч и Джеймс Рамбо). Она отвечает "на вопросы *когда*, *как, кто, что* и *с помощью чего* реализуется проект", а именно содержит описание:

- технологических процессов (*когда*) – последовательности видов деятельности (работ), дающих ощутимый результат. Технологический процесс, как правило, представляется в виде диаграммы, отображающей состав работ и их последовательность на той или иной стадии разработки ПО;
- видов деятельности (*как*) – работ, осуществляемых исполнителями;
- исполнителей (*кто*) – заинтересованных в реализации проекта отдельных лиц или групп. Исполнитель характеризуется строго определенным поведением и обязанностями (ролью). Поведение выражается через виды деятельности, осуществляемые исполнителем, а обязанности – через результаты, получаемые в процессе выполнения работ. В процессе реализации проекта один и тот же человек может выступать в разных ролях;
- артефактов (*что*) – информации, создаваемой, изменяемой или используемой исполнителями в проекте. Другими словами, артефакт – это не только то, что создается в результате деятельности (***технические артефакты*** – модели системы, исходные коды программ, готовый программный продукт, документация к нему и т. д.), но и то, что направляет эту деятельность (***артефакты управления*** –календарный план, техническое задание, инструкции и т. д.);
- используемых утилит (*с помощью чего*) – программных продуктов, рекомендуемых при выполнении работ.

Унифицированный процесс придерживается спиральной модели (стратегии) жизненного цикла ПО, предложенной Барри Боэмом.

Каждый виток характеризуется приращением (инкрементом) функциональности системы и одинаковым набором технологических процессов и стадий. В рамках одной стадии также используется идея спиральной разработки. Перед началом выполнения каждой стадии планируется количество итераций, каждая из которых характеризуется некоторым приращением результатов. В рамках одной итерации выполняются основные процессы, начиная от формирования требований и заканчивая внедрением.

В Унифицированном процессе принято временное разбиение жизненного цикла на четыре стадии: начало, уточнение, конструирование и переход. Каждая стадия должна завершаться достижением конкретного результата (созданием артефактов), используемого далее в качестве управления последующими работами или завершающего реализацию проекта.

Все виды деятельности направлены на создание артефактов, самым главным и ценным из которых является разработанная информационная система. С точки зрения разработчиков не менее ценными артефактами являются разработанные модели системы, так как они, с одной стороны, фиксируют результаты одних работ, а с другой – выступают в качестве управляющей и направляющей информации для других. В Унифицированном процессе модели, как правило, соответствуют основным технологическим процессам. Каждая модель представляет собой набор взаимосвязанных диаграмм UML и документов. Краткая характеристика моделей дана в следующей таблице.

Таблица 1. Краткая характеристика моделей

Процесс	Модель	Назначение
Формирование требований	Модель вариантов	Отображает существенные функциональные требования к системе в форме, удобной для всех заинтересованных лиц.

	использования	Под существенными требованиями понимаются требования, реализация которых принесет пользователям ощутимый и значимый результат. Наименее формальная, но управляет всем процессом разработки
Анализ требований	Модель анализа	Детализирует варианты использования с точки зрения организации внутренней архитектуры системы, а именно: состава основных сущностей (классов анализа) и взаимодействия между ними. Класс анализа – укрупненный класс (сущность), который в дальнейшем будет разбит на составляющие
Проектирование	Модель проектирования	Содержит полное детализированное описание внутренней архитектуры и алгоритмов работы системы. Применяется внутри организации, разрабатывающей систему
Реализация	Модель реализации	Содержит описание исполняемой системы: компонентов (исходных текстов программ, исполняемых модулей, таблиц БД и т. д.) и схемы развертывания системы
Тестирование	Модель тестирования	Предназначена для проверки соответствия полученного ПО требованиям

Модели описывают проектируемую систему с различных точек зрения и на разном уровне абстракции. При этом некоторые элементы (например, диаграммы или классы) могут одновременно входить в разные модели. Более того, один и тот же элемент может входить в две и более моделей с разной степенью детализации.

Модели могут быть вложены друг в друга. Вложенность моделей изображается двумя способами.

В Унифицированном процессе набор получаемых артефактов, как и технологический процесс, также может отображаться в виде диаграммы. На следующем рисунке показан пример диаграммы артефактов для процесса "Управление проектом".

Качественное и своевременное выполнение проекта невозможно без применения средств автоматизации деятельности – утилит. К утилитам относятся различные инструментальные средства, поддерживающие жизненный цикл ПО. В качестве примера системного подхода к разработке информационных систем можно привести продукты компании IBM Rational, лидера в разработке и сопровождении средств, поддерживающих создание объектно-ориентированных систем:

- управление требованиями – IBM RationalRequisitePro;
- визуальное моделирование и генерация объектного кода – IBM RationalRose, IBM Rational XDE;
- разработку – IBM Rational RapidDeveloper;
- конфигурационное управление – IBM Rational ClearCase;
- управление изменениями – IBM RationalClearQuest;
- автоматизированное документирование – IBM RationalSoDA;
- автоматизированное тестирование – IBM Rational TeamTest, IBM Rational TestFactory, IBM Rational Robot, IBM Rational PurifyPlus, IBM Rational SiteCheckи IBM Rational SiteLoad.

Для эффективного применения этих идей необходимо, чтобы они образовывали единый многоплановый процесс, поддерживающий циклы, фазы, рабочие процессы, снижение рисков, контроль качества, управление проектом и конфигурацией. Унифицированный процесс создает каркас, объединяющий все аспекты. Такой комплекс знаний, называют методологией разработки.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРОВЕДЕНИЮ ПРАКТИЧЕСКИХЗАНЯТИЙ

2.1 Практическое занятие №1 (2 часа).

Тема: «Виды моделей компонентов информационных систем**»**

2.1.1 Задание для работы:

1. Изучить понятие системного подхода и системного анализа.
2. Изучить документ. Электронный документ. Информационная система.
3. Изучить информационную технологию.
4. Изучить модели жизненного цикла информационных систем.
5. Изучить каскадную модель.

2.1.2 Краткое описание проводимого занятия:

Основной общий принцип системного подхода. Трактовки понятия системного анализа. Определение документа, электронного документа, понятие и назначение информационной системы. Понятие информационной технологии. Понятие жизненного цикла информационных систем. Виды моделей жизненного цикла. Понятие каскадной модели. Преимущества и недостатки каскадной модели.

2.1.3 Результаты и выводы:

Системный подход включает в себя выявление структуры системы, типизацию связей, определение атрибутов, анализ влияния внешней среды.

Системный анализ характеризуется главным образом упорядоченным, логически обоснованным подходом к исследованию проблем и использованию существующих методов их решения, которые могут быть разработаны в рамках других наук.

Целью системного анализа является полная и всесторонняя проверка различных вариантов действий с точки зрения количественного и качественного сопоставления затраченных ресурсов с получаемым эффектом.

Документ является основным способом представления информации. ***Электронный документ*** — это бумажный документ, введённый в *компьютер* для обработки.

Информационная система предназначена для своевременного обеспечения надлежащих людей надлежащей информацией, то есть для удовлетворения конкретных информационных потребностей в рамках определенной предметной области, при этом результатом функционирования информационных систем является информационная продукция — документы, информационные массивы, базы данных и информационные услуги.

Информационная технология – процесс, использующий совокупность средств методов сбора, обработки и передачи первичной информации для получения информации нового качества о состоянии объекта, т.е. информационного продукта.

Информационные технологии состоят из этапов, каждый из них включает операции, а последние состоят из элементарных действий, таких как нажатие какой-нибудь клавиши, выбор позиции в меню и т.д.

Жизненный цикл ИС можно представить как ряд событий, происходящих с системой в процессе ее создания и использования.

Модель жизненного цикла отражает различные состояния системы, начиная с момента возникновения необходимости в данной ИС и заканчивая моментом ее полного выхода из употребления. Модель жизненного цикла - структура, содержащая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения программного продукта в течение всей жизни системы, от определения требований до завершения ее использования.

Каскадная модель жизненного цикла предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на

следующий этап означает полное завершение работ на предыдущем этапе. Требования, определенные на стадии формирования требований, строго документируются в виде технического задания и фиксируются на все время разработки проекта. Каждая стадия завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

2.2 Практическое занятие №2 (2 часа).

Тема: «Виды систем проектирования АСОИ**»**

2.2.1 Задание для работы:

1. Изучить унифицированный процесс – управляемый вариантами использования.
2. Изучить унифицированный процесс - ориентирован на архитектуру.
3. Изучить унифицированный процесс - итеративный и инкрементный.
4. Изучить жизненный цикл в унифицированном процессе.
5. Изучить продукт унифицированного процесса.
6. Изучить унифицированный процесс – методология разработки.

2.2.2 Краткое описание проводимого занятия:

Унифицированный процесс разработки программного обеспечения с точки зрения управляемости вариантами использования. Унифицированный процесс с точки зрения ориентации на архитектуру. Итеративный и инкрементный унифицированный процесс. Фазы жизненного цикла в унифицированном процессе. Что включает в себя продукт унифицированного процесса. Модели унифицированного процесса. Методология разработки унифицированного процесса.

2.2.3 Результаты и выводы:

Унифицированный процесс есть процесс разработки программного обеспечения. Процесс разработки программного обеспечения - это сумма различных видов деятельности, необходимых для преобразования требований пользователей в программную систему. Однако Унифицированный процесс - это больше, чем единичный процесс, это обобщенный каркас процесса, который может быть специализирован для широкого круга программных систем, различных областей применения, уровней компетенции и размеров проекта.

Унифицированный процесс компонентно-ориентирован. Это означает, что создаваемая программная система строится на основе программных компонентов, связанных хорошо определенными интерфейсами.

Для разработки чертежей программной системы Унифицированный процесс использует Унифицированный язык моделирования.

Архитектура - это представление всего проекта с выделением ключевых составляющих и затушевывание деталей. Архитектура вырастает из требований к результату, в том виде, как их понимает пользователь и другие заинтересованные лица.

Каждый продукт имеет функции и форму, причем одно без другого не существует. В нашем случае функции, как мы ранее отмечали, соответствуют вариантам использования, а форма – архитектуре. Согласно знаниям, заложенным в методологии (унифицированного процесса),сначала должны быть разработаны варианты использования, то есть функции, а потом для того чтобы обеспечить выполнение этих функций разрабатывается архитектура системы. С другой стороны архитектура должна обеспечить реализацию необходимых сейчас и в будущем функций, то есть вариантов использования. Реально архитектура и варианты использования разрабатываются параллельно.

Таким образом, архитектор придает системе форму и архитектор, проектируя форму, должен заложить такие решения, которые бы позволили системе развиваться не только в момент начальной разработки, но и в будущих поколениях системы.

На каждой итерации разработчики определяют и описывают уместные варианты использования, создают проект, использующий выбранную архитектуру в качестве направляющей, реализуют проект в компоненты и проверяют соответствие компонентов вариантам использования. Если итерация достигла своей цели, процесс разработки переходит на следующую итерацию. Если итерация не выполнила своей задачи, разработчики должны пересмотреть свои решения и попробовать другой подход.

Унифицированный процесс циклически повторяется. Эта последовательность повторений Унифицированного процесса представляет собой жизненный цикл системы. Каждый цикл завершается поставкой выпуска продукта заказчикам.

Каждый цикл состоит из четырех фаз - анализа и планирования требований, проектирования, построения и внедрения. Каждая фаза, как будет рассмотрено ниже, далее подразделяется на итерации.

Каждый цикл осуществляется в течение некоторого времени. Это время, в свою очередь, делится на четыре фазы: фазу анализа и планирования требований, фазу проектирования, фазу построения и фазу внедрения. Внутри каждой фазы руководители или разработчики могут потерпеть неудачу в работе - но только на данной итерации и в связанном с ней приращении. Каждая фаза заканчивается вехой. Мы определяем каждую веху по наличию определенного набора артефактов, например, некая модель документа должна быть приведена в предписанное состояние.

Результатом каждого цикла является новый выпуск системы, а каждый выпуск - это продукт, готовый к поставке. Он включает в себя тело - исходный код, воплощенный в компоненты, которые могут быть откомпилированы и выполнены, плюс руководство и дополнительные компоненты поставки. Однако готовый продукт должен также быть приспособлен для нужд не только пользователей, а всех заинтересованных лиц. Программному продукту следовало бы представлять собой нечто большее, чем исполняемый машинный код.

Окончательный продукт включает в себя требования, варианты использования, нефункциональные требования и варианты тестирования. Он включает архитектуру и визуальные модели - артефакты, смоделированные на Унифицированном языке моделирования. Эти средства позволяют заинтересованным лицам использовать систему и модифицировать ее от поколения к поколению.

Для эффективного применения этих идей необходимо, чтобы они образовывали единый многоплановый процесс, поддерживающий циклы, фазы, рабочие процессы, снижение рисков, контроль качества, управление проектом и конфигурацией. Унифицированный процесс создает каркас, объединяющий все аспекты. Такой комплекс знаний, называют методологией разработки.

2.3 Практическое занятие №3 (2 часа).

Тема: «Типы диаграмм в языке UML**»**

2.3.1 Задание для работы:

1. Изучить классификацию диаграмм, принятые обозначения.
2. Изучить изображение ассоциаций на диаграммах классов.
3. Иерархии классов.
4. CRC-карточки.
5. Диаграмма классов.
6. Статические (static) и динамические классы.
7. Диаграммы объектов.

8. Диаграммы прецедентов.
9. Диаграмма состояний (конечных автоматов).
10. Диаграммы последовательностей.
11. Диаграммы коммуникации (взаимодействия).
12. Зачем так много различных диаграмм?
13. Диаграммы видов деятельности.
14. Диаграммы компонентов.
15. Диаграммы развертывания.
16. Диаграммы пакетов.
17. Временная диаграмма.

2.3.2 Краткое описание проводимого занятия:

Диаграмма UML. Структурные и поведенческие диаграммы. Ассоциации на диаграмме классов. Порядок наследования отношений в иерархии классов. Назначение CRC-карточек. Цель построения диаграммы классов. Описание статических и динамических классов. Определение диаграммы объектов, назначение и представление диаграммы. Понятие и цели создания диаграммы прецедентов. Цель создания диаграммы состояний. Понятие диаграммы последовательности. Назначение диаграммы коммуникации. Отличительные особенности. Использование диаграммы видов деятельности. Назначение диаграммы компонентов. Назначение диаграммы развертывания, пакетов и временной диаграммы. Узлы диаграммы развертывания. Расширенная временная диаграмма.

2.3.3 Результаты и выводы:

Рассматривая диаграмму UML, необходимо помнить, что основной принцип UML заключается в том, что любая информация на конкретной диаграмме может быть подавлена. Это подавление может носить глобальный характер – скрыть все атрибуты – или локальный – не показывать какие-нибудь конкретные классы. Поэтому по диаграмме вы никогда не можете судить о чем-нибудь по его отсутствию. Даже если метамодель UML имеет поведение по умолчанию, например для атрибутов, когда вы не видите эту информацию на диаграмме, это может быть обусловлено либо поведением по умолчанию, либо тем, что она просто подавлена.

Диаграммы классов используются при моделировании ПС наиболее часто. Они являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. Диаграмма классов не отображает динамическое поведение объектов изображенных на ней классов. На диаграммах классов показываются классы, интерфейсы и отношения между ними.

Диаграммы классов используются при моделировании ПС наиболее часто. Они являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. Диаграмма классов не отображает динамическое поведение объектов изображенных на ней классов. На диаграммах классов показываются классы, интерфейсы и отношения между ними.

Самым распространенным видом отношения зависимости является соединение между классами, когда один класс использует другой в качестве параметра операции.

Моделирование отношений наследования осуществляется в таком порядке:

1. Найти атрибуты, операции и обязанности, общие для двух или более классов из данной совокупности
2. Вынести эти элементы в некоторый общий класс (если надо, создайте новый, но следите, чтобы уровней не оказалось слишком много).
3. Отметить в модели, что более специализированные классы наследуют более общим, включив отношение обобщения, направленное от каждого потомка к его родителю.

CRC-карточки - эффективный способ анализа сценариев. Карточки можно раскладывать так, чтобы представить формы сотрудничества объектов. С точки зрения

динамики сценария, их расположение может показать поток сообщений между объектами, с точки зрения статики они представляют иерархии классов.

Диаграмма классов описывает типы объектов системы и различного рода статические отношения, которые существуют между ними. На диаграммах классов отображаются также свойства классов, операции классов и ограничения, которые накладываются на связи между объектами.

Структурная классификация описывает системные сущности и их отношения между собой. В число классификаторов, имеющихся в моделях UML, входят классы, варианты использования, компоненты и узлы. Классификаторы являются базой, на которой строится динамическое поведение системы.

Динамическое поведение описывает поведение системы во времени. Поведение можно определить как ряд изменений в мгновенных снимках системы, полученных со статической точки зрения.

Диаграмма объектов показывает взаимосвязи экземпляров некоторых классов. Она используется для пояснения некоторых частей системы со сложными отношениями между объектами, особенно в случае использования рекурсивных отношений.

Диаграммы прецедентов относятся к той группе диаграмм, которые представляют динамические или поведенческие аспекты системы. Это отличное средство для достижения взаимопонимания между разработчиками, экспертами и конечными пользователями продукта. Такие диаграммы очень просты для понимания и могут восприниматься и, что немаловажно, обсуждаться людьми, не являющимися специалистами в области разработки ПО.

Главное назначение диаграммы состояний - описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение моделируемой системы в течение всего ее жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий. Системы, которые реагируют на внешние действия от других систем или от пользователей, иногда называют реактивными. Если такие действия инициируются в произвольные случайные моменты времени, то говорят об асинхронном поведении модели.

Диаграммы последовательностей - это отличное средство документирования поведения системы, детализации логики сценариев использования; но есть еще один способ - использовать диаграммы взаимодействия.

Диаграмма коммуникации показывает во многом ту же информацию, что и диаграмма последовательности, но из-за другого способа представления информации какие-то вещи на одной диаграмме видеть проще, чем на другой. Диаграмма коммуникаций нагляднее показывает, с какими элементами взаимодействует каждый элемент, а диаграмма последовательности яснее показывает в каком порядке происходят взаимодействия.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Диаграммы деятельности состоят из ограниченного количества фигур, соединённых стрелками.

Диаграмма компонентов обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода. Одни компоненты могут существовать только на этапе компиляции программного кода, другие на этапе его исполнения. Диаграмма компонентов отражает общие зависимости между компонентами, рассматривая последние в качестве классификаторов.

Диаграмма развёртывания моделирует физическое развертывание артефактов на узлах

Узлы устройств — это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК,

мобильные телефоны. Узел среды выполнения — это программный вычислительный ресурс, который работает внутри внешнего узла и который предоставляет собой сервис, выполняющий другие исполняемые программные элементы.

Диаграммы пакетов могут использовать пакеты, содержащие прецеденты для иллюстрации функциональности программного обеспечения системы. Диаграммы могут использовать пакеты, которые представляют различные слои программного комплекса для иллюстрации его слоистой архитектуры. Зависимости между этими пакетами могут быть снабжены метками / стереотипами, чтобы указать механизм связи между слоями.

Временные диаграммы особенно удобны при отображении общего хода проекта - его состояния, истории события и того, что должно быть сделано. Обычно временные диаграммы включают важные события и маркеры интервалов.