1. Этапы проектирования по времени выполнения работы

- Научно-исследовательские работы(НИР)
  Результатом НИР является предложение новых методов, методологий решений
- Опытно-конструкторские работы(ОКР)
  Заключается в подготовке эскизного проекта(предложения по структуре схемы, её элементам). Результат – предварительная схема продукта
- Техническое проектирование
  Рабочий проект. Разрабатывают конкретную техническую документацию на изготовление проектируемого устройства.
- Испытания опытного образца
  Изготавливается опытный образец и проводятся его испытания

2. Этапы проектирования по характеру учитываемых свойств объекта

- функциональное проектирование
  Анализ ТЗ и корректировка, разработка структурных, функциональных, логических и принципиальных схем проектируемого объекта
- алгоритмическое проектирование
  Все что связано с ПО – от системы команд до ОС(разработка новых команд, алгоритмов, программной реализации алгоритмов)
- конструкторское проектирование
  Документация для изготовление изделия – все что связано с физической реализацией(реализация полученных решений в конкретной элементной базе, разработка монтажной схемы)
- технологическое проектирование

Включает разработку документации на технологию изготовления проектируемого объекта

3. Блочно-иерархический принцип проектирования

Пусть объект проектирования (ОП) характеризуется тройкой вида ОП={F,S,P}, где F, S и P - соответственно функциональное, структурное и параметрическое описания объекта.

Функциональное описание отражает траекторию ОП в пространстве времени-состояний как некоторую функцию, аргументами которой являются управляющие воздействия и пассивные воздействия внешней среды. Управляющие воздействия могут быть как внешними, так и внутренними, т.е. генерироваться в соответствии с некоторыми скрытыми внутри объекта правилами функционирования.

Задача проектирования в рамках формального определения связана с декомпозицией исходного F-описания на некоторые подфункции (компоненты): $F^0 = S(F_j^1)$, j=1,2, …,n , где S - оператор, определяющий такую композицию F_i^1, которая обеспечивает исходное функциональное описание (F^0). S-оператор в дальнейшем называется структурным описанием (S-описанием) и задает структуру ОП на рассматриваемом уровне детализации. Некоторой части из полученных в результате декомпозиции функциональных описаний компонент (F_i) могут соответствовать известные объекты, которые называются элементами. Элемент может быть достаточно сложной технической системой. Существенным в этом случае является то, что при проектировании F-описание *элемента* не требует дальнейшей декомпозиции и, *следовательно, он не имеет S-описания*.

Для оставшейся части Fi вновь необходима декомпозиция и т.д. до тех пор, пока все F-описания не будут соответствовать элементам.

Выбор варианта декомпозиции, как правило, определяется *качеством* полученного решения.

Пусть качество есть множество свойств ОП, называемых параметрами P={p_i}, i=1,2, …,k. При этом p_i есть *вычислимые* функции, значения аргументов которых определяются параметрическими описаниями компонент *следующего* уровня (P_j^{i+1}) поскольку другие уровни пока *не определены*. Выделение курсивом слова «вычислимые» означает, что значение параметров должно обязательно быть *скалярной величиной*. В противном случае значения параметров нельзя сравнивать, т.е. применять к ним операции отношения.

Каждый из этапов вертикального деления(по характеру учитываемых свойств объекта) делится, также по горизонтали:

ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

1. Системное проектирование;
2. Функционально-логическое проектирование;
3. Конструкторское(схемотехническое) проектирование;
4. Компонентное проектирование.

АЛГОРИТМИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

1. Программирование всей системы;
2. Программирование модулей;
3. Проектирование микропрограмм.

КОНСТРУКТОРСКОЕ ПРОЕКТИРОВАНИЕ

1. Конструирование шкафов(проектирование "шкаф-стойка");
2. Проектирование панелей;
3. Проектирование ТЭЗ (технический элемент замены);
4. Проектирование модуль, кристалл, ячейка.

ТЕХНОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ
1. Разработка принципиальных схем технологического процесса;
2. Маршрутная технология;
3.Разработка технологических операций.

4. Этапы проектирования по блочно-иерархическому принципу проектирования

Каждый из этапов вертикального деления(по характеру учитываемых свойств объекта) делится, также по горизонтали:

ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ
1. Системное проектирование;
2. Функционально-логическое проектирование;
3. Конструкторское(схемотехническое) проектирование;
4. Компонентное проектирование.

АЛГОРИТМИЧЕСКОЕ ПРОЕКТИРОВАНИЕ
1. Программирование всей системы;
2. Программирование модулей;
3. Проектирование микропрограмм.

КОНСТРУКТОРСКОЕ ПРОЕКТИРОВАНИЕ
1. Конструирование шкафов(проектирование "шкаф-стойка");
2. Проектирование панелей;
3. Проектирование ТЭЗ (технический элемент замены);
4. Проектирование модуль, кристалл, ячейка.

ТЕХНОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ
1. Разработка принципиальных схем технологического процесса;
2. Маршрутная технология;
3.Разработка технологических операций.

5. Задача синтеза. Общая храктеристика. Методы её решения на каждом этапе

Синтез – составление или создание описания системы по заданным законам функционирования.

Синтез – составление или создание описания объекта, выполняющего заданные функции и удовлетворяющего заданным ограничениям.

Описание – набор инструкций в каком–либо алфавите.

Задача синтеза выполняется в выбранном классе элементарных объектов, из которых составляется объект, реализующий заданный класс функций.

Исходные данные: описание функций, возлагаемых на пректируемый объект; перечень параметров, характеризующих качество и ограничения на их значения.

Результат – некоторая структура, реализующая заданный класс функций.

Под СТРУКТУРОЙ объекта понимается множество S = {C,H}, где C – множество элементов, входящих в структуру объекта, а H - множество связей между ними.

Две структуры называются равными, если они реализуют равные функции (F1=F2), состоит из одинаковых элементов ({C1}={C2}), которые связаны одинаковыми связями ({H1}={H2}).

Две структуры называются ЭКВИВАЛЕНТНЫМИ, если F1=F2, но C1≠C2 и(или) H1≠H2.

Задача синтеза может иметь формальные методы решения – такая задача алгоритмически разрешима, иначе алгоритмически неразрешима. Алгоритмически неразрешимые задачи решаются в ручную или с помощью эвристических методов (полный перебор).

Различают СИНТЕЗ СТРУКТУРНЫЙ и СИНТЕЗ ПАРАМЕТРИЧЕСКИЙ. Цель структурного синтеза - получение структурных схем объекта, содержащих сведения о составе элементов и способах соединения их между собой. Цель параметрического синтеза – определение числовых значений параметров элементов.

Синтез называется ОПТИМИЗАЦИЕЙ, если целью синтеза является улучшение, в заданном смысле, структуры и значений параметров. Задачу выбора оптимальной структуры называют СТРУКТУРНОЙ ОПТИМИЗАЦИЕЙ.

При расчете оптимальных значений параметров при заданной структуре говорят о ПАРАМЕТРИЧЕСКОЙ ОПТИМИЗАЦИИ.

6. Задача анализа. Общая храктеристика. Методы её решения на каждом этапе

АНАЛИЗ - это определение функционального и параметрического описания системы по заданному структурному описанию.

Предмет решения задачи анализа – исследование свойств F, S и P-описаний, полученных на некотором шаге при спуске по

дереву проектных решений. Целью такого исследования является оценка качества полученного варианта решения или верификация F-описания на соответствие заданному.
В отличие от задачи синтеза, задача анализа алгоритмически всегда разрешима. Утверждение справедливо, поскольку вариант решения задачи синтеза уже получен и известны, по крайней мере, соответствующие ему F и S –описания.
Задача анализа решается с помощью моделирования.
Наиболее общими методами анализа являются одновариантный (исследование объекта в заданной точке траектории поведения) и многовариантный (исследование свойств объекта в окресностях заданной точки траектории поведения).
Адекватность – показатель соответствия модели анализируемому объекту.

7. Общая храктеристика задачи оптимизации

Оптимальными считаются те значения, которые удовлетворяют ТЗ и являются лучшими из достижимых.
Задача ОПТИМИЗАЦИИ в САПР сводится к преобразованию физического представления об объекте, о его назначении и степени полезности(валентности) в математическую формулировку экстремальной задачи. Цель оптимизации выражается в критериях оптимизации.
Критерии – правила предпочтения сравниваемых вариантов. Основу критериев оптимизации составляет целевая функция F(x), где x – множество управляемых параметров. Векторы x с фиксированными значениями определяют один из вариантов объекта и его характеристики.
Целевая функция должна быть такой, чтобы по ее значениям можно было определить степень достижения цели, т.е. лучший вариант должен характеризоваться большим значением F(X), тогда оптимизация заключается в максимализации F(X) или наоборот при минимизации F(X) лучший вариант должен характеризоваться меньшими значениями параметров.
Кроме целевой функции F(X) и перечня управляемых параметров (X) в постановку задачи оптимизации могут входить ОГРАНИЧЕНИЯ ТИПА РАВЕНСТВ H(X)=0 и НЕРАВЕНСТВ H(X)≠0. Частным случаем ограничений типа неравенств являются прямые ограничения ai<=xi<=bi, где ai и bi - предельно допустимые значения параметра x_i.
Область пространства управляемых параметров, в которой выполняются заданные ограничения, называет ДОПУСТИМОЙ ОБЛАСТЬЮ XD.
Объект называется строго оптимальным, если значения всех параметров находятся в допустимой области значений параметров.
Объект называется квазиоптимальным, если некоторые параметры из вектора (X) выходят за границы ограничений, но при этом ограничения границ должны быть строго заданы.
При наличии ограничений задача оптимизации называется УСЛОВНОЙ ОПТИМИЗАЦИИ, в противном случае – БЕЗУСЛОВНОЙ.
Область, в которой выполняются как прямые ограничения, так и условия работоспособности, называется ОБЛАСТЬЮ РАБОТОСПОСОБНОСТИ.
Таким образом, итоговая формулировка задачи оптимизации при проектировании имеет вид: экстремизировать целевую функцию F(X) в области XD, заданной ограничениями H(X)=0 и ф(X)>0.
Задача оптимизации в такой постановке есть задача математического программирования. При линейности функций F(X), H(X), ф(X) - задача линейного программирования. Если хотя бы одна из них нелинейна - задача нелинейного программирования.
Если все (или часть) X - дискретны, то задача дискретного (или частично дискретного) программирования.
Дискретное программирование называется целочисленным, если X принадлежит множеству целых чисел. Если XD есть пространство булевых переменных, то – задача бивалентного программирования.

Задача структурной оптимизации сводится к построению оптимальной структуры S=(E,H). При этом под ОПТИМАЛЬНЫМ будем понимать такой вариант структуры, параметры которой удовлетворяют всем системным, конструктивным, технологическим, электрическим и экономическим требованиям ТЗ, а критерий оптимальности, описывающий качество проектируемой структуры, принимает экстремальное значение.

8. Общая храктеристика критериев оптимизации

Цель оптимизации выражается в критериях оптимизации.

Критерии – правила предпочтения сравниваемых вариантов. Основу критериев оптимизации составляет целевая функция F(x), где x – множество управляемых параметров. Векторы X с фиксированными значениями определяют один из вариантов объекта и его характеристики.

Целевая функция должна быть такой, чтобы по ее значениям можно было определить степень достижения цели, т.е. лучший вариант должен характеризоваться большим значением F(X), тогда оптимизация заключается в максимализации F(X) или наоборот при минимизации F(X) лучший вариант должен характеризоваться меньшими значениями параметров.

Кроме целевой функции F(X) и перечня управляемых параметров (X) в постановку задачи оптимизации могут входить ОГРАНИЧЕНИЯ ТИПА РАВЕНСТВ H(X)=0 и НЕРАВЕНСТВ H(X)≠0. Частным случаем ограничений типа неравенств являются прямые ограничения ai<=xi<=bi, где ai и bi - предельно допустимые значения параметра x_i.

При этом под ОПТИМАЛЬНЫМ будем понимать такой вариант структуры, параметры которой удовлетворяют всем системным, конструктивным, технологическим, электрическим и экономическим требованиям ТЗ, а критерий оптимальности, описывающий качество проектируемой структуры, принимает экстремальное значение.

Если частные критерии fi(x) следует минимизировать, то используется принцип минимакса

F(x0) = min max {fi(x)}.

Аддитивные критерии выбирают, когда существенное значение имеют абсолютные числовые значения критериев при выбранном векторе X.

Если существенную роль играют изменения абсолютных значений частных критериев при выборе вектора X, то целесообразно применять мультипликативный критерий.

Если стоит задача достижения равенства нормированных значений конфликтных частных критериев, то оптимизацию следует производить по максминному (минимаксному) критерию.

9. Частные критерии оптимизации

При проектировании по частным критериям в качестве целевой функции F(X) применяется наиболее важный выходной параметр проектируемого объекта, все остальные параметры в виде соответствующих условий работоспособности относятся к ограничениям. В этом случае задача оптимального проектирования является однокритериальной задачей математического программирования: экстремизировать значение целевой функции F(X) при наличии системы ограничений на параметры проектируемого объекта. Сложность такой задачи небольшая.

Частные критерии выбирают тогда, когда необходимо сравнить несколько эквивалентных решений, либо заранее задана необходимость оптимизации одного или нескольких частных критериев(без существенных ограничений на другие критерии).

10. Аддитивные критерии оптимизации

В аддитивных критериях целевая функция образуется путем сложения нормированных значений частных критериев. Нормированные критерии представляют собой отношение реального значения частного критерия к некоторой нормирующей величине, измеряемой в тех же единицах, что и сам критерий (приводит к безразмерной величине).

Возможны несколько подходов к выбору нормирующего делителя.

Первый подход предполагает принимать в качестве нормирующих делителей максимальные значения критериев, достигаемые в области существующих проектных решений(в мировой практике).

Второй подход предполагает принимать в качестве нормирующих делителей то оптимальное(наилучшее) значение, которое задано в ТЗ.

Третий подход предполагает в качестве нормирующих делителей использовать разность между максимальным и минимальным значением критерия в области компромисса.

Целевая ф-ция: $F(x) = \Sigma C_i * f_{i(x)} = \Sigma C_i * F_{i(x)}/F_{норм.i(x)}$, где C_i – коэффициент значимости i-го критерия(чаще всего находят методом экспертных оценок), $F_{норм.i(x)}$ – нормированная величина

Недостатки:

- формальный прием, не вытекает из объективной роли частного критерия;
- может происходить взаимная компенсация частных критериев.

11. Мультипликативные критерии оптимизации

Используется принцип справедливой компенсации абсолютных значений нормированных частных критериев, которые формируются так: справедливым следует считать такой компромисс, когда суммарный уровень относительногоснижения одного или нескольких критериев не превышает суммарного уровня относительного увеличения других критериев.

$\Sigma\ \Delta F_{i(x)}/F_{i(x)}$ в идеале равно 0(где $\Delta F_{i(x)}$ – приращение, $F_{i(x)}$ – начальные значения)

Если $\Delta F_{i(x)} << F_{i(x)}$, то $\Sigma\ \Delta F_{i(x)}/F_{i(x)} = \Sigma\ D(\ln F_{i(x)}) = \Pi\ Fi(x)$

Иногда важно учитывать не абсолютное значение критерия, а его изменение при решении некоторой задачи.

Целевая ф-ция: $F(x) = \Pi\ Fi(x)$

В случае неравноценности частных критериев необходимо ввести весовой коэффициент Ci и тогда мультипликативный критерий примет вид:

$$F(x) = \prod_{i=1}^{n} Fi^{Ci}(x) \text{ или } F(x) = \prod_{i=1}^{n} Ci\ Fi(x)$$

Достоинством мультипликативного критерия является то, что при его использовании не требуется нормирования частных критериев.

Недостаток: критерий может компенсировать чрезмерные изменения одних критериев за счет изменения других.

12. Минимаксные критерии оптимизации

Основан на принципе равномерного компромисса.

$F_{i(x)}/F_{норм.i(x)}=К$

С учетом важности критерия

$f_{i(x)} = F_{i(x)}/F_{норм.i(x)}$

$Ci\ f_{i(x)} = К$

Выполняются такие вариации значений параметров, при которых последовательно подтягиваются те нормированные значения, численные значения которых в оцениваемом варианте оказались наихудшими.

Максмин: Нужно выбрать такое значение целевой функции $F_{норм.(x)}$, которое обеспечивает максимальное значение $f_{(x)}$, при минимальном значении параметров

Если стоит задача достижения равенства нормированных значений конфликтных частных критериев, то оптимизацию следует производить по максминному (минимаксному) критерию.

Формально принцип максмина формулируется следующим образом:

Необходимо выбирать такое множество $X_0 \in X$, на котором реализуется максимум из минимальных значений частных критериев $F(x0) = \max \min \{ fi(x) \}$.

Если частные критерии fi(x) следует минимизировать, то используется принцип минимакса

$F(x0) = \min \max \{fi(x)\}$.

13. Способы определения коэффициентов важности(значимости) критериев

14. Краткая характеристика САПР

Проектирование:

- ручное (проектирование человека без использования ВТ для решения проектных задач – когда нет устоявшихся методик решения этой задачи),
- автоматизированное (решение проектных задач с помощью человека–проектировщика с использованием ВТ – существуют алгоритмы решения задачи с участием человека),
- автоматическое (задание – машине, от нее результат - существуют известные алгоритмы решения задачи).

Одной из основных целей деятельности человека в сфере создания материального продукта является автоматизация самого процесса его создания. Задача: облегчение и ускорение разработки объектов проектирования.

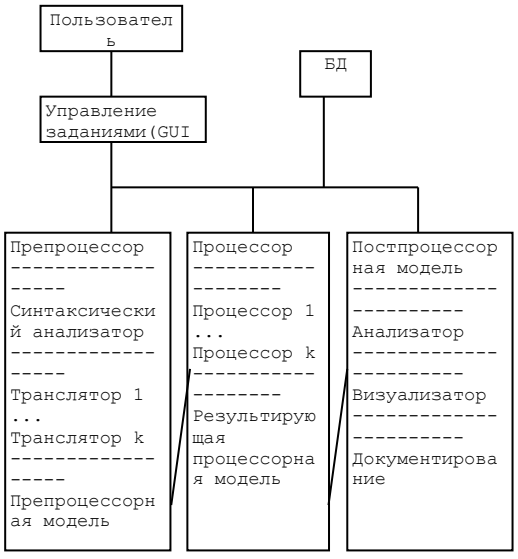
Система автоматизированного проектирования (САПР) - организационно-техническая система, состоящая из комплекса средств автоматизации проектирования (КСАП), взаимосвязанного с необходимыми подразделениями проектной организации или коллективом специалистов (пользователей системы) и выполняющая автоматизированное проектирование. Соответственно система автоматического проектирования выполняет автоматическое проектирование без участия человека.

САПР – совокупность средств и методов для осуществления автоматизированного проектирования, состоящее из ряда частей, называемых обеспечениями:

- техническое – это совокупность взаимосвязанных и взаимодействующих технических средств, предназначенных для выполнения проетирования. К ТО относятся устройства вычислительной и организационной техники, средства передачи данных, измерительные и другие устройства,

- математическое – математические методы, математические модели, алгоритмы проектирования, решения проектирования, необходимые для выполнения проетирования. Все математические средства делятся на общие и инвариантные
- лингвистическое - все языковые средства, используемые в САПР. К ним относятся языки программирования, специальные языки, языки описания входных данных, языки представления решений, языки внутреннего представления данных. Лингвистическое обеспечение - совокупность языков проектирования (ЯП), включая термины и определения, правила формализации естественного языка и методов сжатия и развертывания текстов, необходимых для выполнения АП, представленных в заданной форме.
- программное – все программы, используемые в САПР. Программное обеспечение(ПО) - это совокупность машинных программ, необходимых для выполнения проетирования, представленных в заданной форме. Все ПО делится на операционное и частное(для конкретных задач). Часть ПО, предназначенную для управления проектированием, называют операционной системой (ОС). Совокупность машинных программ (МП), необходимых для выполнения какой-либо проектной процедуры и представленных в заданной форме, называют пакетом прикладных программ (ППП),
- Информационное - это совокупность сведений, необходимых для выполнения АП, представленных в заданной форме. Основной частью ИО являются автоматизированные банки данных, которые состоят из баз данных (БД) САПР и систем управления базами данных (СУБД). В ИО входят нормативно-справочные документы, задания государственных планов, прогнозы технического развития, типовые проектные решения, системы классификации и кодирования технико-экономической информации, системы документации типа ЕСКД, ЕСТД, файлы и блоки данных на машинных носителях, фонды нормативные, плановые, прогнозные, типовых решений, алгоритмов и программ и т.д.,
- методическое – методики решения проектных задач, совокупность документов, устанавливающих состав и правила отбора и эксплуатации средств обеспечения АП, необходимых для выполнения АП. Отметим, что в некоторых работах и документах методическое обеспечение понимается более широко: в качестве компонентов включает МО и ЛО.
- Организационное - совокупность документов, устанавливающих состав проектной организации и ее подразделений, связей между ними, их функции, а также форму представления результата проектирования и порядок рассмотрения проектных документов, необходимых для выполнения АП. Компонентами ОО САПР являются методические и руководящие материалы, положения, инструкции, приказы и другие документы, обеспечивающие взаимодействие подразделений проектной организации при создании и эксплуатации САПР.

Типовая структура приложения САПР.



15. Техническое обеспечение САПР. Требования к техническому обеспечению

Техническое обеспечение – это совокупность взаимосвязанных и взаимодействующих технических средств, предназначенных для выполнения автоматизированного проетирования. К ТО относятся устройства вычислительной и организационной техники, средства передачи данных, измерительные и другие устройства.

Техническое обеспечение включает в себя технические средства (ЭВМ, периферийное оборудование), с помощью которых решаются задачи проектирования.

Состав технического обеспечения определяет техническое назначение разрабатываемых САПР.

Требования к нему:

- Удобство использования;
- Возможность оперативного вмешательства человека в выполнение проектной задачи;
- Достаточная производительность и объем ОП для решения задач всех задач проектирования за приемлемое время;
- Возможность одновременной работы с техническими средствами необходимого числа пользователей для эффективной деятельности всего коллектива разработчиков;
- Открытость комплекса технических средств для модернизации системы по мере прогресса техники;
- Высокая надежность, приемлемая стоимость.

Система, удовлетворяющая всем требованиям называется комплексом технических средств, который организовывается в виде специализированной вычислительной системы, допускающей функционирование в нескольких режимах.

16. Математическое обеспечение САПР. Требования к математическому обеспечению

МО САПР состоит из математических моделей объектов проектирования, методов и алгоритмов выполнения проектных операций и процедур(синтез структуры, получение математических моделей, одновариантный и множественный анализ, параметрическая оптимизация).

В МО САПР можно выделить *специальную часть* в значительной мере отражающую специфику объекта проектирования физические и информационные особенности его функционирования и тесно привязанную к конкретной иерархическим уровням и *инвариантную часть,* включающую в себя методы и алгоритмы, слабо связанные с особенностями математических моделей и используемые на многих иерархических уровнях (методы и алгоритмы многовариантного анализа и параметрической оптимизации).

Требования к МО САПР

Свойства МО Оказываются существенное а иногда и определяющее влияние на возможности и показатели САПР.

Универсальность. Под универсальностью МО понимается его применимость к широкому классу проектируемых объектов. Одно из отличий расчетных методов САПР от ручных – высокая степень универсальности.

Высокая степень универсальности нужна для того, чтобы САПР была применима к любым или к большинству объектов проектируемых на предприятии.

Степень универсальности не имеет количественной оценки. Реализуя ту или иную модель или метод, *разработчик МО должен оказать четкие границы их применимости.*

Алгоритмическая надежность. Свойство компонента МО давать при его применении правильные результаты называется алгоритмической надежностью.

К сожалению, не всегда условия применимости моделей и методов могут быть найдены, исследованы и сформулированы в виде конкретных инструкций пользователя. Часто применимость компонента МО зависит от конкретных условий, которые не всегда поддаются исчерпывающему учету и классификации. Методы и алгоритмы не имеющие строгого обоснования называются э*вристическими.* Отсутствие четко сформулированных условий применимости приводим к тому, что эвристические методы могут использоваться некорректно, что приведет к далекому от истинного решения, либо решение вообще не будет получено(вследствие отсутствия сходимости). При этом может не оказаться данных, позволяющих обнаружить некорректность решения. Следовательно, возможна ситуация, когда неверное решение будет использоваться дальше.

Степень универсальности характеризуется заранее оговоренными ограничениями, а алгоритмическая надежность – ограничениями, заранее не оговоренными.

Количественной оценкой алгоритмической надежности служит вероятность получения правильных результатов при соблюдении оговоренных ограничений неприменение метода. Если эта вероятность равна 1 или близка к ней – метод алгоритмически надежен.

Применение алгоритмически ненадежных методов нежелательно хотя и допустимо в случаях, когда неправильные результаты легко распознаются(конструкторское проектирование).

Обусловленность математических модулей и задач – свойство не увеличивать погрешности. О плохой обусловленности говорят, когда малые погрешности исходных данных приводят к большим погрешностям результатов.

На каждом этапе вычислений свои промежуточные исходные данные и результаты, свои источники погрешностей. При плохой обусловленности погрешности могут резко возрасти, что может привести как к *снижению точности*, так и к *росту затрат машинного времени.* Для анализа и оптимизации объектов с плохо обусловленными математическими моделями необходимо применять специальные методы с повышенной алгоритмической надежностью.

Точность – степень совпадения расчетных и истинных результатов. Алгоритмически надежные методы могут давать различную точность решения.

В большинстве случаев решение проектных задач характеризуется:

- совместным использованием многих компонентов МО, что затрудняет определение вклада в общую погрешность каждого из них;
- векторным характером результатов, т.е. результатом решения является значения не отдельного параметра, а многих параметров.

В связи с этим оценка точности производиться с помощью специальных вычислительных экспериментов, в которых создаются условия для раздельной оценки погрешностей, вносимых мат. Моделями элементов, алгоритмами анализа, и оптимизации. В этих экспериментах используются специальные задачи, называемые тестовыми. *Количественная оценка* решения тестовой задачи есть одна из норм вектора относительных погрешностей.

m – норма $\xi_{\max} = \max_{j\in[1..m]} |\xi_j|$ или 1 – норма $\xi_{îáù} = -\sqrt{\sum_{J=1}^{m} \xi_j^{\,2}}$

ξ_j - относительная погрешность j-того элемента вектора результатов, m – размерность вектора.

Относительная погрешность компонента ξ оценивается по совокупности учитываемых параметров одной из норм вектора $\xi = (\xi_1, \xi_2 ..., \xi_m)$, где $\xi_j = \frac{y_j^2 - y_j}{y_j}$.

Затраты машинного времени. Универсальные модели и методы характеризуются сравнительно большим объемом вычислений , растущим с увеличением размерности задач. Поэтому при решении большинства в САПР затраты машинного времени T_M значительны. Обычно именно T_M является главным ограничивающим фактором при попытке повысить сложность проектируемых на ЭВМ объектов. Потому минимизация T_M – одно из основных требований к МО САПР.

При использовании в САПР многопроцессорных ВС уменьшить время счета можно с помощью параллельных вычислений. В связи с этим одним из показателей экономичности МО является его *приспособленность к распараллеливания вычислительного процесса.*

Требования высокой степени универсальности, алгоритмической надежности, точности с одной стороны, и малых затрат машинного времени с ругой, противоречивы. Поэтому любой конкретный компонент МО, отражая определенный компромисс этих требований, может быть эффективным при решении одной задачи и малоэффективным при решении другой. Поэтому в САПР целесообразно иметь библиотеки с наборами моделей и методов. Каждый компонент МО определенного целевого назначения должен быть представлен несколькими разновидностями, обеспечивающими разную степень удовлетворения противоречат требований.

Затраты памяти являются вторым после T_M показателем экономичности МО. Они определятся длиной программы и объем используемых массивов данных. Несмотря на значительное увеличение емкости оперативной

памяти в современном ЭВМ, требование экономичности по ее затратам остается актуальным.

Для разрешения возникающих трудностей можно использовать внешнюю память, но частые обмены данными между внешней и оперативной памятью приводят к значительному увеличению Тм. Поэтому при больших объемах программ и массивов обрабатываемой информации целесообразно использовать МО, допускающее построение оверлейный программных структур и реализующие принципы диакоптической обработки информации.

Значительное влияние на развитие МО САПР оказало стремлении повысить экономичность используемых моделей и методов. Это достигается как разработкой экономных моделей и алгоритмов, имеющих частный характер, так и совершенствованием и использованием общих принципов создания МО, эффективного по затратам Тм и памяти. К таким принципам относятся учет разреженности матриц, исследовании сложных систем по частям (диакоптические методы исследования), макромоделирование, событийность анализа и рациональное использование способностей человека в интерактивных процедурах.

Учет разреженности матриц основан на хранении в ЭВМ только ненулевых элеметнов матриц и выполнении арифметических операций только над ними.

Диакоптика основана на использовании структурных особенностей схем, а не на принятии каких-либо упрощающих допущений. При этом производится расчленение математических моделей на части, исследуемые самостоятельно. Это позволяет упорядочить и минимизировать количество обменов информацией между оперативной и внешней памятью при анализе сложных систем, а также выбрать для исследования каждой части наиболее выгодные режимы.

Макромоделирование лежит в основе направления, связанного с рациональным выбором математических моделей элементов при построении математической модели системы. Оно реализует возможность использования при анализе одного и того же объекта нескольких моделей, различающихся сложностью точностью и полнотой отображения свойств объекта, трудоемкостью требующихся вычислений и т.д. Каждая из моделей на элементы и выбору определенных моделей полученных элементов. Наиболее детальное разбиение в рамках данного иерархического уровня приводит к получению полной математической модели, характеризующейся высокой точностью и большим объемом требуемых вычислений. Повысить экономичность можно делением системы на более крупные блоки с использованием для них упрощенных мат. моделей – макромоделей.

Событийность анализа заключается в том, что при имитации процессов, протекающих в исследуемом объекта во времени, вычисления проводятся только для тех элементов состояние которых на очередном временном шаге может изменится.

Рациональное использование эвристических способностей человека в интерактивных процедурах позволяет человеку вмешиваться в ход вычислений и выбирать наиболее перспективного продолжения на основе эвристических оценок.

17. Общая храктеристика инвариантного математического обеспечения

Инвариантное математическое обеспечение включает в себя методы и алгоритмы, слабо связаны с особенностями математических моделей и используемые на многих иерархических уровнях. Включает:

1. Методы и алгоритмы оптимизации;
2. Многовариантный анализ;
3. Логико-комбинаторные методы решения.

Основными методами многовариантного анализа являются анализ чувствительности и статический анализ.

Цель анализа чувствительности – определение коэффициентов чувствительности (коэффициентов влияния). Абсолютный коэффициент чувствительности определяет влияние i-го внутреннего параметра на j-й внешний параметр: aj,i = dyj / dxi;

Относительный коэффициент чувствительности выходного параметра yj к изменениям внутреннего параметра xi имеет вид: bj,i = aj,i*xiном/yjном.

Применяется при параметрической оптимизации. Основные методы определения:

- метод приращения;
- прямой и обратный;
- вариационный метод.

Цель статистического анализа – получение оценок рассеивания выходных параметров и вероятностей выполнения заданных условий работоспособности для проектируемого объекта.

Результатами статистического анализа являются: - гистограмма выходных параметров;
- оценки математических ожиданий и среднеквадратичные отклонения каждого из yj;
- максимально возможные отклонения yjном;
- оценки коэффициентов кореляции между yj и xj и выходных параметров СМО.
В качестве исходных данных используется сведения о рассеивании внутренних параметров и допустимых диапазонов изменения или законов распределения внешних параметров.
Наиболее часто для статистического анализа используются методы наихудшего случая и метод статистических испытаний (метод Монте-Карло).
Логико-комбинаторные методы используются в процессе синтеза и решения задач конструкторского проектирования. Относятся к области дискретного программирования. Для их постановки требуется найти перечень параметров, характеризующих свойства вариантов так, чтобы различить варианты и сформулировать целевую функцию F(x) и ограничения в пространстве параметров, входящих в вектор X.
Общая постановка задачи дискретного программирования – формулировка задачи оптимизации с дополнительными ограничениями: значения всех параметров должны принадлежать счетному множеству.
Комбинаторные методы делятся на: 1. Методы отсечения (наиболее часто используется метод Гомори); 2. Комбинаторные методы (метод ветвей и границ); 3. Приближенный метод (метод локальной оптимизации).

18. Анализ чувствительности

Цель анализа чувствительности – определение коэффициентов чувствительности, называемых также коэффициентами влияния:

$a_{ij} = \frac{\partial_{yj}}{\partial_{xj}}$; $b_{ij} = \frac{a_{ji} * x_{jíîì\ .}}{y_{jíîì\ .}}$, где a_{ji} и b_{ji} - абсолютный и относительной коэффициент чувствительности выходного параметра y_j к изменениями внутреннего параметра x_j.

Результатом анализа чувствительности m выходных параметров к изменениям n внутренних параметров представляют собой mn коэффициентов чувствительности, составляющих матрицу a_{ji} или b_{ji}. Если исследуется влияние внешних параметров, вместо x_i и $x_{iíîì\ .}$ фигурирует внешний параметр q_j и его номинальное значение $q_{jíîì\ .}$.

Анализ чувствительности применяется, если X и Q можно считать непрерывными величинами, а y_j являются дифференцируемыми функциями своих аргументов x_i и q_j.

Результаты анализа чувствительности используются при решении таких задач как параметрическая оптимизация, расчет допусков, оценка точности выходных параметров. Именно по значениям a_{ji} и b_{ji} определяются параметры, существенно влияющие от несущественных, определяют направление изменений внутренних параметров для улучшения выходных параметров, оценивают допустимые отклонения параметров X и Q для выполнения точностных требований к параметру У.

Основными методами анализа чувствительности являются методы приращений, прямой и вариационный методы.

Основным достоинством метода приращений является его универсальность, а также возможность распараллеливания вычислительного процесса, простоту программной реализации.

Недостатками являются невысокая точность, сравнительно большая трудоемкость вычисления.

Метод приращений – метод численного дифференцирования зависимости $Y = F(X,Q)$. Алгоритм метода приращений включает в себя (n+1) - кратны обращения к модели для вычисления У, где n – количество варьируемых параметров. В 10-м варианте задаются номинальные значения аргументов и , следовательно результатом обращения к модели будет $Y_{íîì\ .} = (Y_{1íîè}, ..., Y_{míîè})$, т. е. номинальное значение вектора У в очередном (i+1) варианте среди оставшихся n вариантов задается отклонение Δx_i от $\Delta x_{iíîì\ ,}$ только одному из варьируемых параметров. В результате выполнения (i+1) варианта получаются для вектора У значение $Y_{i.} = (Y_{1i}, ..., Y_{mi})$ по которому оценивается очередной столбец матрицы абсолютной чувствительности $A_i = \frac{Y_i - Y_{íîì\ .}}{\Delta x_i}$.
Любой из найденных коэффициентов A_i легко пересчитать в B_i.

19. Статистический анализ

Целью статистического анализа является получение оценок рассеяния выходных параметров У и вероятностей выполнения

заданных условий работоспособности для проектируемого объекта.

В случае объектов СМО сами выходные параметры имеют вероятностный смысл, тогда цель статического анализа – расчет таких параметров. Причинами рассеяния выходных параметров У являются нестабильность внешних параметров а и случайный характер внутренних параметров х. Результатами статистического анализа могут быть гистограммы выходных параметров, оценки мат. Ожиданий M_j и среднее квадратическое отклонений σ_j каждого из y_j , максимально возможные отклонения y_j от $y_{jном.}$ Оценки коэффициентов корреляции γ_{ij} между параметрами y_j и x_j, а также выходных параметров СМО.

В качестве исходных данных фигурируют статистические сведения о рассеянии внутренних параметров и данных ТЗ о допустимых диапазонах изменения или законах распределения Q.

Наибольшее распространение при статистическом анализе получили методы наихудшего случая и статистических испытаний (метод Монте-Карло).

Метод наихудшего случая. Служит для определения диапазонов возможного рассеяния выходных параметров без оценки плотности распределения этих параметров.

Пусть на некоторой выходной параметр задано условие работоспособности в виду y < ТТ. Тогда интерес представляет верхняя границы диапазона рассеяния, т.к. большие значения y наиболее опасны с точки зрения невыполнения условий работоспособности. Верхняя граница диапазона рассеяния достигается в наихудшем случае, когда все аргументы y = f(x) принимают самые неблагоприятные значения. Самым неблагоприятным значением x_i будет $x_{imax.}= x_{iном.}+ \Delta x_i$ при выполнении условия $(y < TT \cap \frac{dy}{dx_i} > 0) \cup (y < TT \cap \frac{dy}{dx_i} < 0)$ или $x_{imin.}=x_{iном.}-\Delta x_i$, если $(y < TT \cap \frac{dy}{dx_i} < 0) \cup (y > TT \cap \frac{dy}{dx_i} > 0)$, где Δx_i – допуск на внутренний параметр x_i.

Алгоритм метода наихудшего случая состоит из операторов:

1. Анализ чувствительности, в результате которого определяются коэффициенты чувствительности $\frac{dy}{dx_i}$
2. Задание параметрам x_i самых неблагоприятных значений
3. Выполнение анализа объекта в наихудшем случае.

Каждому выходному параметру y_i соответствует свой наихудший случай. Если объект характеризуется m выходными параметрами и n внутренними параметрами, то всего требуется n+m+1 вариантов обращения к модели объекта.

Преимущество данного метода в том, что для его применения не требуется знание законов распределения значений внутренний параметров. Достаточно знать лишь допуски Δx_i. Недостаток в том, что вероятность сочетания в объекте самых неблагоприятных или близких к ним значений параметров очень мала. Поэтому оценка рассеяния параметра по этому методу оказывается значительно больше чем по более точному методу статистический испытаний.

Метод Монте-Карло*(метод статистических испытаний)*. Этот метод позволяет получить более полные статистические сведения о выходных параметрах исследуемого объекта.

Алгоритм метода:

1. Задание значений внутренних параметров (аргументов зависимости y от x и Q) в очередном статистическом испытании.
2. Расчет y.
3. Накопление статистических сумм.
4. Обработка накопленных сумм для получения результатов статистического анализа.

Операторы 1-3 выполняются в каждом испытании и могут быть распараллелены.Оператор 4 завершает статистический анализ.

Задание случайных значений параметров выполняется в соответствии с их законами распределения.

Накопление статистических сумм нужно для последней оценки. Это суммы вида:

$$S_{j1} = \sum_{k=1}^{H} y_{jk};$$

$$S_{j2} = \sum_{k=1}^{M} y^2{}_{jk};$$

$$S_{ji3} = \sum_{k=1}^{N} y_{jk} * x_{ik};$$

$$S_y = \sum_{k=1}^{N} q_k \text{ , где}$$

y_{jk} и x_{jk} - значение y_i и x_i в к-том испытании;

q_k=1 если в к-том испытании некоторые условия выполнено, иначе q=0;

N = количество выполненных испытаний.

Отработка накопленных сумм производится после выполнения N испытаний.

При этом оценивается

Математическое ожидание M_j. и *дисперсия* $\sigma^2{}_j$ выходных параметров y_i:

$$M_j = \frac{S_{j1}}{N};\ \sigma_j^2 = \frac{S_{ji3} - N * M^2{}_j}{N-1}.$$

Коэффициенты корреляции γ_{ij} между параметрами y_i и x_i:

$$\gamma_{ij} = \frac{S_{ji3} - N * M_j * M_i}{(N-1) * \sigma_j * \sigma_i}$$

Вероятность выполнения условий работоспособности

$p = \frac{S_y}{N}$, где

M_j и σ_j - математические ожидание и среднеквадратичное отклонение параметра x_i.

Преимущество метода – в его универсальности

Недостаток – большой объем требующихся вычислений.

20. Логико-комбинаторные методы

Относятся к задачами дискретного программирования. Обычно используются в процессе решения задачи структурного синтеза. Для их постановки требуется:

1) найти перечень некоторых параметров, объединенных в вектор X, значения которых характеризуют свойства вариантов и позволяют различить варианты между собой.

2) сформулировать целевую функцию F(x) и ограничения в пространстве значений параметров, входящих в вектор X.

Общая формулировка задачи дискретного программирования:

extr F(x); $xD = \{x \mid \varphi(x > 0, \psi(x) = 0, x \in G)\}$ $x \in xD$, где

G – счетное множество (в частном случае конечное).

Виды методов:

1) описания (метод Гомори);
2) комбинаторные (ветвей и границ);
3) приближенные (локальной оптимизации).

Комбинаторные методы и методы отсечения являются итерационными Их называются точными, т.к. итерационный процесс сходится к экстремальной точке задачи.

В отличие от них применение приближенных методов от них не гарантирует получение экстремума, хотя в большинстве случаев найденное решение близко к оптимальному.

В *методах отсечения* каждая итерация представляет собой решение задачи непрерывного математического программирования. На десятой итерации решается исходная задача, но со снятым условием дискретности $x \in G$, т.е. допустимая область $xD = \{x \mid \varphi(x) \geq 0, \varphi(x) = 0\}$.

По результатам решения задачи вводится некоторое ограничение $U_1(x)$ сужающее xD, но таким образом, чтобы экстремальная точка исходной задачи оставалась в этой области. Далее процедура повторяется, вводятся новые ограничения $U_к(x)$ (производятся отсечения области xD) до тех пор, пока результат очередной итерации не будет удовлетворять условию $x \in G$.

Узловой момент для реализации идей отсечения наличие корректных правил для формулирование дополнительных ограничений $U_к(x)$. Такие правила удается найти в частных случаях.

Типичный пример методов отсечения – метод Гомори для решения задач целочисленного линейного программирования.

Комбинаторные методы предназначаются для решения комбинаторных задач, т.е. для поиска наилучшего варианта в конечном множестве.

Могут быть решены методом полного перебора всех вариантов. Однако полный перебор практически неосуществим в большинстве случаев из-за чрезмерно большого объема вычислений. Поэтому комбинаторные методы - это, как правило методы сокращенного перебора. Большие группы комбинаторных методов составляют методы сокращенного неявного перебора – методы ветвей и границ.

Так в задаче размещения p микросхем на плате количество вариантов может доходить до p.

В задаче синтеза жестов для проверки логических схем количество вариантов входных наборов при p входах – до 2^p.

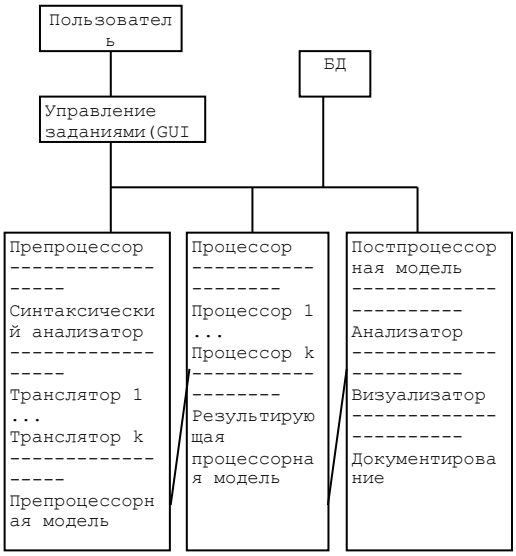
Приближенные методы отличаются большим разнообразием, многие из них жестко связанны со спецификой конкретных задач и относятся к специальному МО.

При описании и алгоритмической реализации большой группы приближенных методов используются представления и терминология теории графов.

К приближенным методам, имеющим более общий характер, относятся методы локальной оптимизации.

21. Типовая структура САПР

Типовая структура приложения САПР.



Функции управления заданиями сводятся к настройке компонентов приложения для решения конкретной задачи.

Функции интерактивного управления процессом решения.

Функции настройки решения делят на 3 части:

- Настройка препроцессора
- Настройка процессора
- Настройка постпроцессора
- Настройка БД для решения конкретной задачи

Функции препроцессора определяются различиями в описании проектных задач и исходных данных. Включает в себя синтаксический анализатор (чтобы удостовериться в правильности исходного описания) и трансляторов исходного описания в процессорную модель, с которой будет работать ПО.

Документирование – получение чертежей и описания полученного решения.

Настройка БД осуществляется отдельно для препроцессора, процессора, постпроцессора.

22. Общая характеристика информационного обеспечения САПР

Информационное обеспечение (ИО) АП - это совокупность сведений, необходимых для выполнения АП, представленных в заданной форме. Совокупность компонентов, входящих в ИО, образуют информационный фонд. Ядром ИО является база знаний. Основной частью ИО являются автоматизированные банки данных, которые состоят из баз данных (БД) САПР и систем управления базами данных (СУБД).

БД – структурированная совокупность связанных данных конкретной предметной области разнообразного назначения, в которой отражаются состояния объектов, их свойства и взаимоотношения. Объект – любой предмет, х-ющийся именем, свойствами, атрибутами. Имеет идентификатор(ы).

В ИО входят нормативно-справочные документы, задания государственных планов, прогнозы технического развития, типовые проектные решения, системы классификации и кодирования технико-экономической информации, системы документации типа ЕСКД, ЕСТД, файлы и блоки данных на машинных носителях, фонды нормативные, плановые, прогнозные, типовых решений, алгоритмов и программ и т.д.

23. Древовидные БД

Иерархическая БД состоит из упорядоченного набора деревьев; более точно, из упорядоченного набора нескольких экземпляров одного типа дерева.

Тип дерева состоит из одного "корневого" типа записи и упорядоченного набора из нуля или более типов поддеревьев (каждое из которых является некоторым типом дерева). Тип дерева в целом представляет собой иерархически организованный набор типов записи.

Все экземпляры данного типа потомка с общим экземпляром типа предка называются близнецами. Для БД определен полный порядок обхода - сверху-вниз, слева-направо.

Недостаток: информация может быть получена только через корневой вход и возможные типы связий заранее фиксированы, трудно вносить изменения.

Иерархические БД предназначены для облегчения хранения и поиска записей, которые можно организовать строго иерархическим образом. Иерархические БД очень просты и поэтому понять их не так уж сложно, особенно если вы используете их для упорядочения информации под соответствующими заголовками.

Сущности — это базовые типы информации, хранимой в БД (как правило, в отдельном файле или таблице). Они представляют собой фундаментальные блоки данных. Связи представляют собой соединения между частями (записями, таблицами, файлами)

БД. Сущности — это фактические данные, классифицируемые по типу, а связи показывают, как эти типы данных соотносятся один с другим. Когда вы таким образом описываете БД, вы говорите о *модели сущность-связь* для этой БД. Как выглядят эти модели, кратко называемые ERD (Entity Relationship Diagram — диаграммы сущность-связь)?

Линии со стрелками представляют связи между сущностями. Почему у каждой линии только одна стрелка (т.е. почему эти линии являются однонаправленными)? Стрелки — это специальные символы, отмечающие связь *один ко многим.*

Иерархическую БД можно представлять как "электронный шкаф с картотекой". Информация организуется таким образом, что первый уровень иерархии соответствует объединению папок, второй — отдельным папкам внутри секции, а третий — отдельным записям, документам или формам, находящимся внутри каждой папки. Нижний уровень в любой "бумажной" иерархической системе соответствует строкам на отдельной странице. Прелесть иерархической БД в компьютере заключается в том, что можно организовать сотни уровней, если в этом возникнет потребность.

24. Сетевые БД

Сетевая БД строится на связях «многие ко многим» и практически содержит несколько иерархий с повторяющимися данными и связями между иерархиями.

Этим базам не свойственно хранение данных в одном месте. При этом возникают сложности с репликацией (откат и сохранение данных во всех местах).

В иерархической БД каждая запись находится только в одной папке. Взаимосвязь между всеми записями в такой БД можно представить в виде единственной иерархической диаграммы.

Сетевая БД допускает одновременное наличие записей в нескольких папках. На диаграмме сетевой БД, показано, что отдельные записи соединены с другими записями несколькими линиями. В иерархической БД каждая запись связана с единственной записью-предком, расположенной над ней в данной иерархии. В сетевой БД каждая запись может быть связана с любой другой записью.

С концептуальной точки зрения сетевая БД позволяет вам *одновременно* организовать данные любым числом иерархических способов.

Почему эта БД называется сетевой? Вернемся к определению сети. Когда вы составляете схему записей и связей между ними, результат выглядит как сеть. Линии, соединяющие все записи, "пересекаются, ответвляются и соединяются", формируя "ажурную ткань или структуру".

Каждая стрелка говорит нам о том, что соответствующие записи в файле являются "потомками" записей в другом файле, из которого она исходит. Если две стрелки указывают на некоторый файл, то каждая из записей в этом файле может иметь двух "предков". Таким образом, одной иерархии в любом случае будет недостаточно для ответа на все вопросы по БД, в которой стрелки идут в нескольких направлениях, и для отражения нескольких иерархий требуется сеть.

Сетевые БД, безусловно, сложнее, чем иерархические. В иерархической структуре существуют только два направления, по которым можно двигаться из любой записи: вверх или вниз. В сетевой структуре количество вариантов намного больше

25. Реляционные БД

Реляционные БД - двухмерные таблицы, каждая строчка которой содержит данные одного элемента.

Любой столбик может быть ключом. Связь между таблицами осуществляется по любому элементу этих таблиц.

Последовательность атрибутов, относящихся к одному эл-у, наз. кортежем. Кажд. столбец наз. доменом.

Свойства таких таблиц:

- Каждый элемент таблицы представляет собой один элемент данных.
- Повторяющиеся записи отсутствуют. Все столбцы в таблице однородные, т.е. в каждом столбце записываются однотипные элементы.
- Столбцам присваиваются однозначные имена. В таблицах нет двух одинаковых строк.
- В операциях с таблицами строки и столбцы могут просматриваться в любом порядке безотносительно к их информации, содержанию и смыслу.

Понятие связи интерпритируется как запрос на языке SQL.

Для этих БД разработан строгий мат. аппарат – «реляционная алгебра».

В середине 70-х годов Эдгар Кодд (Edgar Codd), исследователь из IBM, начал работать над новой моделью БД, получившей название *реляционной* модели. Учитывая такое название, вы вправе предположить, что стержнем подобной модели являются связи между файлами. Это вовсе не так. И это лишь одно из ряда ошибочных представлений, которые сложились у многих людей о преимуществах и

недостатках реляционной технологии.

В математике отношением называется некая неупорядоченная совокупность n-кортежей. М-да... А *n-кортеж* — это всего лишь совокупность *n* различных значений — то, что можно было бы назвать записью. Отношение очень напоминает то, что обычно называется файлом.

В реляционном мире файлы являются *таблицами,* записи — *строками,* а поля — *столбцами.* Кодд справедливо подметил, что, хотя большие БД требуют сложной совокупности файлов с богатыми наборами постоянных связей, отражающих отношения, большинство пользователей, работающих с этими БД, живут в куда более простом мире. Действительно, большинство пользователей мыслят не категориями файлов, записей и полей, а категориями таблиц, строк и столбцов. Пользователи чувствуют себя намного увереннее, работая с простыми совокупностями таблиц, чем со сложными наборами файлов и отношений. Сделав такой вывод, Кодд фактически предвосхитил появление несколькими годами позже электронных таблиц.

Реляционные системы не работают непосредственно со связями.

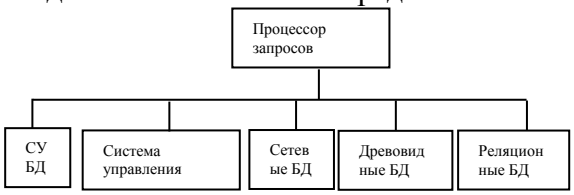
Реляционные БД, в отличие от сетевых и иерархических БД, существенно облегчают установление связей между не связанными до этого файлами. Работая с реляционной БД, программист или пользователь могут быстро и легко определить новую связь между любыми двумя таблицами. Связи можно определить между таблицами, у которых даже не было до этого никаких предварительно определенных связей. В случае сетевых БД связи необходимо сначала определить, и только после этого пользователь сможет получить к ним доступ. У реляционных БД нет такой проблемы — вот почему реляционные системы приобрели такую популярность. Для того чтобы все это стало возможным, в реляционных системах пришлось воспользоваться одной совершенно новой идеей.

С самого начала все реляционные БД комплектовались языком запросов. Этот язык — *SQL (Structured Query Language — язык структурированных запросов),* стал неотъемлемой частью БД. Сегодня нет ничего необычного в том, что SQL и БД неразделимы. Однако в середине 70-х годов между БД и языками запросов пролегала очень четкая граница. В узком смысле, БД — просто механизм для управления общими данными, а язык запросов — особый инструмент для указания совокупностей записей, которые надо извлечь из БД. В идеальном случае к каждой БД можно обращаться посредством многих языков запросов, а каждый язык запросов может работать со многими БД.

26. Модель БД OLE

Взаимодействуют 2 объекта, которые оперируют таблицами или наборами записей. Интерфейс имеет ключевую роль.

Модель OLE может быть представлена:



Процессор запросов выполняет следующие функции:

- Расшифровка запросов
- Планирование действий по его выполнению
- Определение интерфейса

Под интерфейсом понимают набор заранее определенных функций, вызовов процедур для того, чтобы выполнить:

- Запросить таблицу, в которой сведены характеристики доступных БД
- Для каждой БД запросить, какие существуют строки и столбцы(для реляционных БД), какова конфигурация записи, каковы типы значений записи
- Возможность запросить, какие изменения необходимо внести в записи(считать записи), сформировать новые записи

В качестве такого интерфейса используют SQL. С помощью SQL формируют множество предметно-ориентированных БД.

Каждая БД обладает интерфейсом. Каждая БД имеет какую-либо форму интерфейса, который позволяет ее процессору запросов обращаться к сокрытому в ней хранилищу. Затруднение состоит в том, что эти интерфейсы являются неопубликованными и нестандартными.

В начале 1995 г. Microsoft начала открывать разработчикам подробности набора интерфейсов, названного БД OLE. Саму технологию OLE (Object Linking and Embedding — связывание и внедрение объектов) мы рассмотрим подробнее несколько позже, а пока достаточно сказать, что и OLE, и "лежащая в ее основе COM (Component Object Model — модель компонентных объектов) предназначены для построения объектов. В контексте нашего обсуждения БД OLE — это предписание для перестройки самих БД во множество взаимодействующих компонентов.

Таким образом, БД OLE — это множество интерфейсов (определяемых в C и C++), которые описывают, как могут взаимодействовать любых два компонента, каждый из которых работает с таблицами и наборами записей. В БД OLE эти компоненты называются поставщиками табличных данных (Tabular Data Providers — TOP). После опубликования БД OLE в качестве потенциального стандарта интерфейса, разработанного специально для того, чтобы располагаться между предварительно определенными внутренними компонентами БД, стало ясно, как перестроить сами БД, чтобы они были множеством взаимодействующих компонентов. Теперь настало время исследовать некоторые интересные сценарии, которые порождаются этой новой картиной

• Пользователи могут хранить данные в различных источниках по своему выбору. Один пользователь может строить наброски бюджета в электронных таблицах, другой — в системе управления проектом, третий — в настоящей БД. Основанный на БД OLE процессор запросов — раньше неотделимая часть монолитной БД — может комбинировать данные из всех этих многочисленных источников данных, словно они являются одной большой БД. Пользователь процессора запросов, как только запрос был введен, уже не может сказать, откуда пришли данные.

• Многие из производимых в мире приложений запускаются на файловых системах ISAM и в иерархических БД, так как эти системы отличаются большим быстродействием. Тем не менее, внутри каждой релядионной БД имеется хранилище записей в стиле 15AM(стоит ли продолжать?).1 При разбиении БД на компоненты указанным образом, это скрытое хранилище записей становится непосредственно доступным разработчикам через интерфейс БД OLE. Многие разработчики предпочтут продолжать рассматривать всю систему как реляционную БД, получая доступ к данным только через процессор запросов и SQL. Другим разработчикам, однако, может понадобиться повышенная производительность, и они согласятся на дополнительную работу, развивая собственную стратегию выбора данных при использовании скрытого хранилища записей непосредственно. В обоих случаях данные все так же находятся в общей БД, доступной для всех и согласованной по своей природе.

• Что происходит, если вы заменяете процессор запросов? Мы рассмотрели идею о возможности хранения данных в широком множестве источников данных, которыми являются ваши любимые инструментальные средства для создания и редактирования данных, оставляя эти данные доступными для знакомого нам процессора запросов, основанного на SQL. По существу, вы имеете "замещаемый" механизм хранения. Таким же образом можно сделать и замещаемый процессор запросов.

27. Общая храктеристика программного обеспечения САПР

Программное обеспечение САПР включает в себя:

- Системное программное обеспечение, в качестве которого обычно используется ОС, инструментальные программы
- Предметно – ориентированное ПО, поддерживающее сам процесс проектирования.

Средства, поддерживающие процесс проектирования относятся к классу сложных программных систем, использующих базы данных.

Они «похожи» на ПО бизнес–приложений, однако бизнес–приложения являются быстро развивающейся областью, а ПО САПР значительно отстает.

Существующие и тиражируемые САПР как правило спроектированны не лучшим образом. Основные причины этого - ориентация на физическую архитектуру САПР в рамках заданной предметной области.

Исторически приложения САПР строятся как двухуровневые:

- слой предварительной обработки (АРМ с GUI в основе)
- слой окончательной обработки (некоторая БД, совместно с которой работают некоторые приложения, причем БД и приложения жестко взаимосвязаны).

Такая организация приложений является малогибкой, плохо сопровождаемой. Такие приложения крайне сложно погрузить в распределенную среду.

Чтобы облегчить трудности, возникающие при двухуровневой архитектуре, применяют трехуровневую логическую архитектуру приложений со следующими слоями:

- слой документов (приложения рабочего стола и GUI)

- правила проектирования, правила принятия проектных решений и управление проектом проетирования
- управление данными

При условии стандартизации межслойных интерфейсов логические слои можно делать независимо. Т.е. каждый из этих слоев может быть реализован как независимая компонента и распределен. В этом смысле говорят о сервере БД, сервере приложений и приложениях рабочего стола.
Физическая и логическая архитектуры мало связаны между собой, т.е. трехуровневое приложение можно физически реализовать на одной ЭВМ.

28. Двухуровневая-, трехуровневая система программного обеспечения САПР

Исторически приложения САПР строятся как двухуровневые:

- слой предварительной обработки (АРМ с GUI в основе)
- слой окончательной обработки (некоторая БД, совместно с которой работают некоторые приложения, причем БД и приложения жестко взаимосвязаны).

Такая организация приложений является малогибкой, плохо сопровождаемой. Такие приложения крайне сложно погрузить в распределенную среду.

Чтобы облегчить трудности, возникающие при двухуровневой архитектуре, применяют трехуровневую логическую архитектуру приложений со следующими слоями:

- слой документов (приложения рабочего стола и GUI)
- правила проектирования, правила принятия проектных решений и управление проектом проетирования
- управление данными

При условии стандартизации межслойных интерфейсов логические слои можно делать независимо. Т.е. каждый из этих слоев может быть реализован как независимая компонента и распределен. В этом смысле говорят о сервере БД, сервере приложений и приложениях рабочего стола.
Физическая и логическая архитектуры мало связаны между собой, т.е. трехуровневое приложение можно физически реализовать на одной ЭВМ.

29. Лингвистическое обеспечение САПР

Включает в себя все языковые средства, которые используются в САПР. Делятся на две части:

- языки программирования – инструментальные языки (универсальные, машинноориентированые);
- языки проектирования (входные языки);
- языки управления(языки управления заданиями, языки структурного и функционального описания, языки документирования).

Языки управления заданиями – диалоговые языки, состоят из определенного набора команд, которые задаются текстовой строкой и могут настраивать приложения на выполнение соответствующих функций.
Языки GUI – накладывают определенные стандарты, разработаны для каждой отдельной системы.

Входные языки делятся на:
- языки описания задания;
- языки описания объекта.

Языки описания задания служат для описания маршрута проектирования. Похожи на операционные системы, но ориентированы на САПР.
Языки описания объекта служат для описания функций параметров и структурного описания объекта, входных, промежуточных и результирующих данных.
Требования к входным языкам:
- универсальность;
- удобство восприятия алфавита и синтаксиса языка;
- максимальная лаконичность описания;
- однозначность толкования элементов и конструкций языка;
- возможность развития и расширения.

Языки описания объекта делятся на:
- процедурные языки;
- автоматные языки.

Процедурные языки:
- языки функционального описания;
- языки структурного описания.

Языки функционального описания предназначены для задания закона функционирования объекта без привязки к конкретным схемам. Примеры языков: DDL, HSL-FX.
Языки структурного описания предназначены для описания структуры. Примеры: SDL, HDL.

Автоматные языки служат непосредственно для описания данных, вводимых в задачи проектирования. Делятся на:
- списковые языки;
- табличные языки.

30. Требования к языкам

Требования к входным языкам:

- универсальность;
- удобство восприятия алфавита и синтаксиса языка;
- максимальная лаконичность описания;
- однозначность толкования элементов и конструкций языка;
- возможность развития и расширения.

Можно сформулировать следующие требования к языку:

- Все конструкции языка должны естественно и элегантно определяться в нем.
- Для решения определенной задачи должна быть возможность использовать сочетания конструкций, чтобы избежать необходимости вводить для этой цели новую конструкцию.
- Должно быть минимальное число неочевидных конструкций специального назначения.
- Конструкция должна допускать такую реализацию, чтобы в неиспользующей ее программе не возникло дополнительных расходов.
- Пользователю достаточно знать только то множество конструкций, которое непосредственно используется в его программе.

31. Общая характеристика инструментальных языков

Инструментальные языки - это особая категория программных средств. С их помощью создаются все другие программы. Существует широкая номенклатура языков программирования, каждый из которых характеризуется определенными свойствами.

К категории инструментальных средств относятся не только трансляторы с языков высокого уровня, но и ассемблеры, загрузчики, отладчики и другие системные программы. С помощью инструментальных средств создается и прикладное программное обеспечение, и новые средства системного программирования, включая трансляторы с языков высокого уровня.

Парадигма в программировании – способ концептуализации, который определяет, как следует проводить вычисления, и как работа, выполняемая компьютером, должна быть структурирована и организована.

Можно сформулировать следующие требования к языку:

- Все конструкции языка должны естественно и элегантно определяться в нем.
- Для решения определенной задачи должна быть возможность использовать сочетания конструкций, чтобы избежать необходимости вводить для этой цели новую конструкцию.
- Должно быть минимальное число неочевидных конструкций специального назначения.
- Конструкция должна допускать такую реализацию, чтобы в неиспользующей ее программе не возникло дополнительных расходов.
- Пользователю достаточно знать только то множество конструкций, которое непосредственно используется в его программе.

Важнейшие парадигмы программирования на данный момент времени:

- *директивного* программирования
- *объектно-ориентированного* программирования
- *функционально-логического* программирования

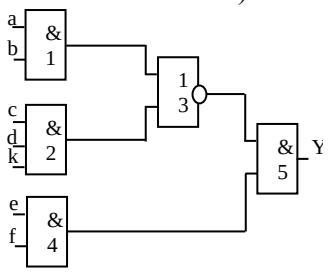
Директивное программирование - когда разработчик программы пытается создать код, должным образом воздействующий на данные. Активным началом считается программа (код), пассивным - данные.

Объектно-ориентированное программирование – первичными считаются объекты (данные), которые могут активно взаимодействовать друг с другом с помощью механизма передачи сообщений (называемого также и механизмом вызова методов). Программист должен придумать и реализовать такие объекты, взаимодействие которых после старта программы приведет к достижению необходимого конечного результата.

Функциональное и логическое программирование - главным является *точная формулировка задачи*, а выбор и применение необходимого для ее алгоритма решения – проблема исполняющей системы, но не программиста.

32. Языки структурного описания объекта

Языки структурного описания предназначены для описания структуры объекта проектирования.
Чаще всего это графовые модели. Могут быть процедурными(похожи на языки программирования) и автоматными(матрицы связности и инцидентности, табличные языки, списковые языки).



Табличное описание:
TF

&	&	1	&	&

TBx

a	b	c	d	k	e	f

T

a	b	c	d	k	1	2	e	f	3	4

KTB

1	3	6	8	10

TBых

3	3	5	5	Y

Списковое описание:
Q,7,a,b,c,d,k,e,f,1,Y,5,...

1,&,2,1,a,1,b

2,&,3,1,d,1,k,1,1,3

Примеры: SDL, HDL.

33. Процедурные языки

34. Этапы системного проектирования

Исходные данные: ТЗ
Результат: структурная схема всей системы, элементами являются устройства, блоки.
Математический аппарат: Теория вычислительных систем
Задача анализа выполняется вероятностными методами, в основе которых лежат теория СМО и теория сетей Петри.
Задача САПР состоит в проверке правильности составления структурной схемы. Проверка производится методом моделирования:

- Модель
- Выбор методов моделирования

Основным назначением моделирования является оценка эксплуатационных качеств объекта, проверка логической правильности структурной организации будущей системы.
Так как нет точных оценок параметров, то система является вероятностной. Поэтому чаще всего в качестве модели используют сети Петри, системы массового обслуживания.
СМО – совокупность приборов для обслуживания заявок и класс задач, решаемых в системе.
Модель системы СМО – это как проходят заявки через систему.
Характеристики входного потока:

- Интенсивность (количество заявок в единицу времени)
- Приоритеты заявок
- Состояние заявки в данный момент времени в системе

Устройства обслуживания:

- Аппараты обслуживания (непосредственно обслуживают заявки; х-ки:занят/не занят, производительность)
- Аппараты хранения заявок(х-ка: объем, занят/не занят)

Очереди к устройствам не относятся к анализируемой системе.
Каждое устройство х-ется (зан/не зан) и очередью (дисциплина обслуживания).
Для анализа СМО необходимы:

- Данные о структуре системы
- Дисциплины обслуживания очередей каждого устройства
- Внутренние параметры устройства
- Значения параметров входного потока заявок

Y=F(X,Q)
Y - система
X – внутренние параметры
Q – выходные параметры

Выходные параметры:

- Производительность системы
- Среднее количество обслуженных заявок
- Среднее количество вышедших необслуженных заявок

- Вероятность обслуживания заявки
- Коэффициент загрузки оборудования
- Величины очередей

Аналитические модели могут быть использованы для простейших СМО.
Простые СМО должны обладать следующими свойствами:

- Ординарность (в единицу времени одна заявка)
- Стационарность (в одинаковые промежутки времени одно и то же количество заявок)
- Отсутствие последействия (вероятность различных непересекающихся интервалов не зависит друг от друга, все заявки независимы)

Заявки разделяются на 2 группы:

- Независимые
- Зависимые

Основу моделирования независимых заявок составляют алгоритмы вычисления моментов появления заявок, которые реализуют выборку значения случайной величины, распределенной по заданному закону (интервал времени между заявками).
Зависимые – момент появления такой заявки зависит от появления другой заявки (синхронизирующей). Любая зависимая заявка появляется внутри системы.

Модель обслуживающего аппарата – алгоритм выработки значений интервала обслуживания. Для каждого типа заявок может быть установлен свой закон распределения.

Алгоритм определения объема памяти, требуемого для обслуживания заявок, зависит от типа заявки.
Заявка, поступившая в память, занимает вычисленный объем вплоть до времени освобождения.

Узлы могут быть следующих типов:

- Для направления заявок по тому или иному маршруту в зависимости от типа заявки или выполнения некоторых условий
- Для разделения заявки на части или объединения таких частей
- Для изменения параметров заявок

В целом имитационная модель – это алгоритмы, состоящие из упорядоченных обращений к моделям элементов (источникам, устройствам, памяти).
Последовательность обращений определяется свойствами моделируемой системы и режимом ее работы.
Описание модели:статическое и динамическое.
Реальное время, модельное, машинное – разница в приращении.

35. Задача синтеза на этапе системного проектирования

36. Задача анализа на этапе системного проектирования

Задача анализа выполняется вероятностными методами, в основе которых лежат теория СМО и теория сетей Петри.
Задача САПР состоит в проверке правильности составления структурной схемы. Проверка производится методом моделирования:

- Модель
- Выбор методов моделирования

Основным назначением моделирования является оценка эксплуатационных качеств объекта, проверка логической правильности структурной организации будущей системы.
Так как нет точных оценок параметров, то система является вероятностной. Поэтому чаще всего в качестве модели используют сети Петри, системы массового обслуживания.
СМО – совокупность приборов для обслуживания заявок и класс задач, решаемых в системе.
Модель системы СМО – это как проходят заявки через систему.
Характеристики входного потока:

- Интенсивность (количество заявок в единицу времени)
- Приоритеты заявок
- Состояние заявки в данный момент времени в системе

Устройства обслуживания:

- Аппараты обслуживания (непосредственно обслуживают заявки; х-ки:занят/не занят, производительность)
- Аппараты хранения заявок(х-ка: объем, занят/не занят)

Очереди к устройствам не относятся к анализируемой системе.
Каждое устройство х-ется (зан/не зан) и очередью (дисциплина обслуживания).
Для анализа СМО необходимы:

- Данные о структуре системы
- Дисциплины обслуживания очередей каждого устройства

- Внутренние параметры устройства
- Значения параметров входного потока заявок

Y=F(X,Q)
Y - система
X – внутренние параметры
Q – выходные параметры

Выходные параметры:
- Производительность системы
- Среднее количество обслуженных заявок
- Среднее количество вышедших необслуженных заявок
- Вероятность обслуживания заявки
- Коэффициент загрузки оборудования
- Величины очередей

Аналитические модели могут быть использованы для простейших СМО.
Простые СМО должны обладать следующими свойствами:
- Ординарность (в единицу времени одна заявка)
- Стационарность (в одинаковые промежутки времени одно и то же количество заявок)
- Отсутствие последействия (вероятность различных непересекающихся интервалов не зависит друг от друга, все заявки независимы)

37. Структура СМО

СМО – совокупность приборов для обслуживания заявок и класс задач, решаемых в системе.
Модель системы СМО – это как проходят заявки через систему.
Характеристики входного потока:
- Интенсивность (количество заявок в единицу времени)
- Приоритеты заявок
- Состояние заявки в данный момент времени в системе

Устройства обслуживания:
- Аппараты обслуживания (непосредственно обслуживают заявки; х-ки:занят/не занят, производительность)
- Аппараты хранения заявок(х-ка: объем, занят/не занят)

Очереди к устройствам не относятся к анализируемой системе.
Каждое устройство х-ется (зан/не зан) и очередью (дисциплина обслуживания).

Для анализа СМО необходимы:
- Данные о структуре системы
- Дисциплины обслуживания очередей каждого устройства
- Внутренние параметры устройства
- Значения параметров входного потока заявок

Y=F(X,Q)
Y - система
X – внутренние параметры
Q – выходные параметры

Выходные параметры:
- Производительность системы
- Среднее количество обслуженных заявок
- Среднее количество вышедших необслуженных заявок
- Вероятность обслуживания заявки
- Коэффициент загрузки оборудования
- Величины очередей

Аналитические модели могут быть использованы для простейших СМО.
Простые СМО должны обладать следующими свойствами:
- Ординарность (в единицу времени одна заявка)
- Стационарность (в одинаковые промежутки времени одно и то же количество заявок)
- Отсутствие последействия (вероятность различных непересекающихся интервалов не зависит друг от друга, все заявки независимы)

Заявки разделяются на 2 группы:
- Независимые
- Зависимые

Основу моделирования независимых заявок составляют алгоритмы вычисления моментов появления заявок, которые реализуют выборку значения случайной величины, распределенной по заданному закону (интервал времени между заявками).
Зависимые – момент появления такой заявки зависит от появления другой заявки (синхронизирующей). Любая зависимая заявка появляется внутри системы.

Модель обслуживающего аппарата – алгоритм выработки значений интервала обслуживания. Для каждого типа заявок может быть установлен свой закон распределения.

Алгоритм определения объема памяти, требуемого для обслуживания заявок, зависит от типа заявки.
Заявка, поступившая в память, занимает вычисленный объем вплоть до времени освобождения.

Узлы могут быть следующих типов:

- Для направления заявок по тому или иному маршруту в зависимости от типа заявки или выполнения некоторых условий
- Для разделения заявки на части или объединения таких частей
- Для изменения параметров заявок

В целом имитационная модель – это алгоритмы, состоящие из упорядоченных обращений к моделям элементов (источникам, устройствам, памяти).
Последовательность обращений определяется свойствами моделируемой системы и режимом ее работы.

38. Аналитическая модель СМО

Основным назначением моделирования является оценка эксплуатационных качеств объекта, проверка логической правильности структурной организации будущей системы.
Так как нет точных оценок параметров, то система является вероятностной. Поэтому чаще всего в качестве модели используют сети Петри, системы массового обслуживания.
СМО – совокупность приборов для обслуживания заявок и класс задач, решаемых в системе.
Модель системы СМО – это как проходят заявки через систему.
Характеристики входного потока:

- Интенсивность (количество заявок в единицу времени)
- Приоритеты заявок
- Состояние заявки в данный момент времени в системе

Устройства обслуживания:

- Аппараты обслуживания (непосредственно обслуживают заявки; х-ки:занят/не занят, производительность)
- Аппараты хранения заявок(х-ка: объем, занят/не занят)

Очереди к устройствам не относятся к анализируемой системе.
Каждое устройство х-ется (зан/не зан) и очередью (дисциплина обслуживания).
Для анализа СМО необходимы:

- Данные о структуре системы
- Дисциплины обслуживания очередей каждого устройства
- Внутренние параметры устройства
- Значения параметров входного потока заявок

Y=F(X,Q)
Y - система
X – внутренние параметры
Q – выходные параметры

Выходные параметры:

- Производительность системы
- Среднее количество обслуженных заявок
- Среднее количество вышедших необслуженных заявок
- Вероятность обслуживания заявки
- Коэффициент загрузки оборудования
- Величины очередей

Аналитические модели могут быть использованы для простейших СМО.
Простые СМО должны обладать следующими свойствами:

- Ординарность (в единицу времени одна заявка)
- Стационарность (в одинаковые промежутки времени одно и то же количество заявок)
- Отсутствие последействия (вероятность различных непересекающихся интервалов не зависит друг от друга, все заявки независимы)

39. Имитационная модель СМО

Простые СМО должны обладать следующими свойствами:

- Ординарность (в единицу времени одна заявка)
- Стационарность (в одинаковые промежутки времени одно и то же количество заявок)
- Отсутствие последействия (вероятность различных непересекающихся интервалов не зависит друг от друга, все заявки независимы)

Заявки разделяются на 2 группы:

- Независимые
- Зависимые

Основу моделирования независимых заявок составляют алгоритмы вычисления моментов появления заявок, которые реализуют выборку значения случайной величины, распределенной

по заданному закону (интервал времени между заявками).
Зависимые – момент появления такой заявки зависит от появления другой заявки (синхронизирующей). Любая зависимая заявка появляется внутри системы.

Модель обслуживающего аппарата – алгоритм выработки значений интервала обслуживания. Для каждого типа заявок может быть установлен свой закон распределения.

Алгоритм определения объема памяти, требуемого для обслуживания заявок, зависит от типа заявки.
Заявка, поступившая в память, занимает вычисленный объем вплоть до времени освобождения.

Узлы могут быть следующих типов:

- Для направления заявок по тому или иному маршруту в зависимости от типа заявки или выполнения некоторых условий
- Для разделения заявки на части или объединения таких частей
- Для изменения параметров заявок

В целом имитационная модель – это алгоритмы, состоящие из упорядоченных обращений к моделям элементов (источникам, устройствам, памяти).
Последовательность обращений определяется свойствами моделируемой системы и режимом ее работы.

40. Общая храктеристика активностей, процессов, событий

Динамическая структура может быть представлена в виде:

- активность
- процесс
- событие

Функционирование системы – это последовательность активных и пассивных состояний.

Любая модель ориентирована на представление активности. Активность является единым динамическим объектом, указывающим на совершение какого-либо действия. Задается интервал времени, необходимый для его выполнения. Надо задать условие выполнения и длительность выполнения, действия по завершению выполнения.

Любая активность может синициировать другую активность, тогда такая активность – инициатор. Любая активность может быть представлена мелкими активностями. Совокупность активностей представляет длительность выполнения одного объекта одной активности.

Активность – любое действие, которое требует время для своего выполнения. Активность – самое подробное описание модели.

Процесс – логически связанный набор активностей.

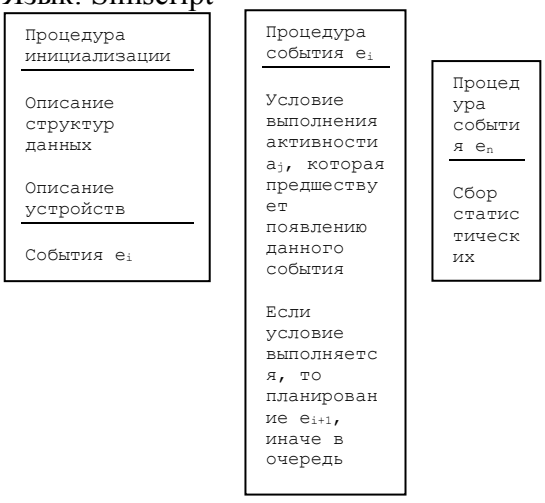
Разница с точки зрения модели:

Процесс может синициировать другой процесс. Процессы, состоящие из одинаковых действий, могут быть представлены одним классом процессов, такой класс – одним программным объектом. Каждый экземпляр такого класса может быть представлен своими значениями атрибутов. Одновременно в системе может выполняться несколько процессов. Один процесс может выполняться много раз в одно и то же время, находясь на разных стадиях выполнения.

События не требуют времени, это мгновенное изменение состояния. Это начало и окончание активности, процесса. Есть событие следования (изменения состояния объекта внутри процесса) и изменение состояния (к разным процессам и представляют собой начало выполнения другого процесса).

41. Системы языков, ориентированные на описание событий

Язык: Simscript



T – время
Устройства: занято/свободно
Очереди к устройствам
Состояние транзакта(таблица транзактов)

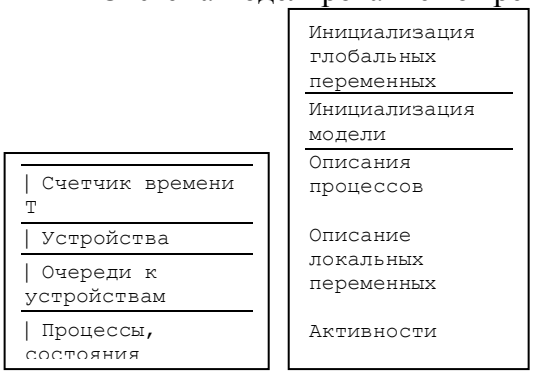
42. Структура системы языков, ориентированной на описание процессов

Языки:

- SOL
- ASPOL
- GASP
- GPSS

Любая система моделирования состоит из 2-х систем:

- Сама модель
- Система моделирование во времени



Состояния процессов:

- готовность
- активность(выполнение)
- задержки
- ожидание

43. Общая характеристика ASPOL

Sim <имя>; основная программа – описание программы, переменных, памяти, инициализация. Главный процесс в системе.

End.

Process <имя>(p1,p2,p3...pn);
формальные параметры
End process <имя>;

Инициализация экземпляра процесса выполняется:
Initiate <имя процесса>(a1,a2,a3,...)
фактические параметры
Может вызываться из другого процесса

С каждым процессом связан приоритет, при инициализации он получает приоритет процесса-инициатора. Приоритет может быть изменен: priority <выражение>;

Транзакт – к какой заявке относится.

Готовые процессы

Имя процесса	Транзакт	T начала выполнения

Процесс может задержать свое выполнение, если: hold(t);

Задержанные процессы

Имя процесса	Транзакт	Время реактивации	Точка реактивации

Ожидающие процессы

Имя процесса	Транзакт	T постановки в очередь	Точка реактивации

Выполняемые процессы

Имя процесса	Транзакт	Атрибуты

События:
С выполнением процесса связаны события, в ASPOL события могут быть заданы явно, неявно. События следования, как правило, задаются неявно. События изменения состояния могут быть заданы явно.
Event e1,e2,e3...en;
Это объявление этих событий, они еще не произошли.
Set(e1); событие сработало
Queue(e); если есть очередь событий

Устройства:
Facility <имя устройства>
В момент создания устройство не занято
Reserve <имя> занять устройство
Release <имя> освободить устройство

Reserve(f,i) Если множество одинаковых устройств и надо занять/освободить i-ое
Release(f,i)

Память:
Storage(<имя памяти>) указывает, какие памяти есть
Size(s) <выражение>указывает весь объем памяти s

Allocate(s) <выражение> сколько памяти s надо занять
Deallocate(s) <выражение> сколько памяти s надо освободить

Распределения:
expute(U) экспоненциальный закон с мат. ожиданием U
erlang(U,V) U – мат. Ожидание, V - дисперсия
random(a,b)
pr = random(0,1)
irandom(a,b)

44. Пример входной программы на ASPOL

n процессоров
m дисководов
k каналов передачи данных

Задания появляются на входе системы со средним интенсивностью 5 зад/сек, интервалы между заданиями имеют отрицательное показательное распределение. Время решения процессорами – Эрланг U=10 мсек, V=8 мсек^2

Средняя скорость ввода/вывода равномерно распределена в диапазоне 2-10 записей в мсек. Время обработки одной записи в процессоре распределено по отрицательному показательному закону с мат. ожиданием = время решения задания в процессоре/общее число записей, обрабатываемых этим заданием.

Операции ввода/вывода равномерно распределены между дисководами, запросы ресурсов общей памяти требуют объема памяти – равномерное распределение 20-60 страниц памяти.

При поступлении в систему задания получают приоритеты, обратно пропорциональные используемой ими памяти.

Получив требуемый объем памяти, задание занимает любой свободный процессор. После выдачи запроса на операцию вв/выв освобождает процессор. После завершения вв/выв задание занимает любой свободный процессор.

```
sim mpos;
  def (n=2),(m=4),(k=2);
  facility cpu(n),disk(m),channel(k);
  storage cm; size(cm) 128;
        while (time<1200) do
        begin
              initiate job; hold(expute 12);
        end
end.
```

```
process job;
  integer space,unit;real tcp,tio,record,dt;
  event ioend;

  tcp = erlang(10,8); record = tcp*random(2,10);
  tio = tcp/record;
  space = irandom(20,60);
  priority = 60-space;
  allocate(cm) space;

  while (tcp>0) do
  begin
        reserve (cpu);
        dt = expute(tio);
        tcp = tcp-dt;
        hold(dt);
        unit = irandom(1,m);
        initiate io (unit,ioend);
        release(cpu);
        wait(ioend);
  end;
  deallocate(cm) space;
end process job;
```

```
process io;
  integer unit; real ts,tc; event flag;

  reserve(disk(unit));
  ts = random(0,75);hold(ts);
  reserve(channel);
  tc = 0,01*(2,5+random(0,25));
  hold(tc);
  release(channel);
  release(disk(unit));
  set(flag);
end process io;
```

45. Внутренняя структура ASPOL

Sim <имя>; основная программа – описание программы, переменных, памяти, инициализация. Главный процесс в системе.
End.

Process <имя>(p1,p2,p3...pn);
формальные параметры
End process <имя>;

Инициализация экземпляра процесса выполняется:

Initiate <имя процесса>(a1,a2,a3,...) фактические параметры
Может вызываться из другого процесса

С каждым процессом связан приоритет, при инициализации он получает приоритет процесса-инициатора. Приоритет может быть изменен: priority <выражение>;

Транзакт – к какой заявке относится.

Готовые процессы

Имя процесса	Транзакт	Т начала выполнения

Процесс может задержать свое выполнение, если: hold(t);

Задержанные процессы

Имя процесса	Транзакт	Время реактивации	Точка реактивации

Ожидающие процессы

Имя процесса	Транзакт	Т постановки в очередь	Точка реактивации

Выполняемые процессы

Имя процесса	Транзакт	Атрибуты

События:
С выполнением процесса связаны события, в ASPOL события могут быть заданы явно, неявно. События следования, как правило, задаются неявно. События изменения состояния могут быть заданы явно.
Event e1,e2,e3...en;
Это объявление этих событий, они еще не произошли.
Set(e1); событие сработало
Queue(e); если есть очередь событий

Устройства:
Facility <имя устройства>
В момент создания устройство не занято
Reserve <имя> занять устройство
Release <имя> освободить устройство
Reserve(f,i) Если множество одинаковых устройств и надо занять/освободить i-ое
Release(f,i)

Память:
Storage(<имя памяти>) указывает, какие памяти есть
Size(s) <выражение>указывает весь объем памяти s
Allocate(s) <выражение> сколько памяти s надо занять
Deallocate(s) <выражение> сколько памяти s надо освободить

Распределения:
expute(U) экспоненциальный закон с мат. ожиданием U
erlang(U,V) U – мат. Ожидание, V - дисперсия
random(a,b)
pr = random(0,1)
irandom(a,b)