Технология программирования

Курс лекций для гр. 3057, 4057

Лекция №5

Содержание

Формы документации ПП

- Проектная - адресованная разработчикам и заказчику:
  - *отчуждаемые* артефакты – результаты этапов разработки (нужны при разработке и сопровождении)
- Эксплуатационная - адресованная пользователям:
  - конечным пользователям – потребителям ПП
  - системным администраторам
- Рабочая – оперативные документы в ходе проекта
  - переписка
  - календарные планы
  - сообщения об ошибках
  - отчеты, служебные записки, предложения и т.п.
    - напр., Postmortem – извлечение уроков из завершенного проекта
- Нормативная – описывающая технологический процесс
  - стандарт кодирования
  - описания процессов (этапов) ЖЦ
  - шаблоны проектных и эксплуатационных документов
  - должностные инструкции
  - и т.п.

Состав проектной документации

Перечень проектных документов, рекомендуемых в стандарте по качеству ПП ISO 9000-3:

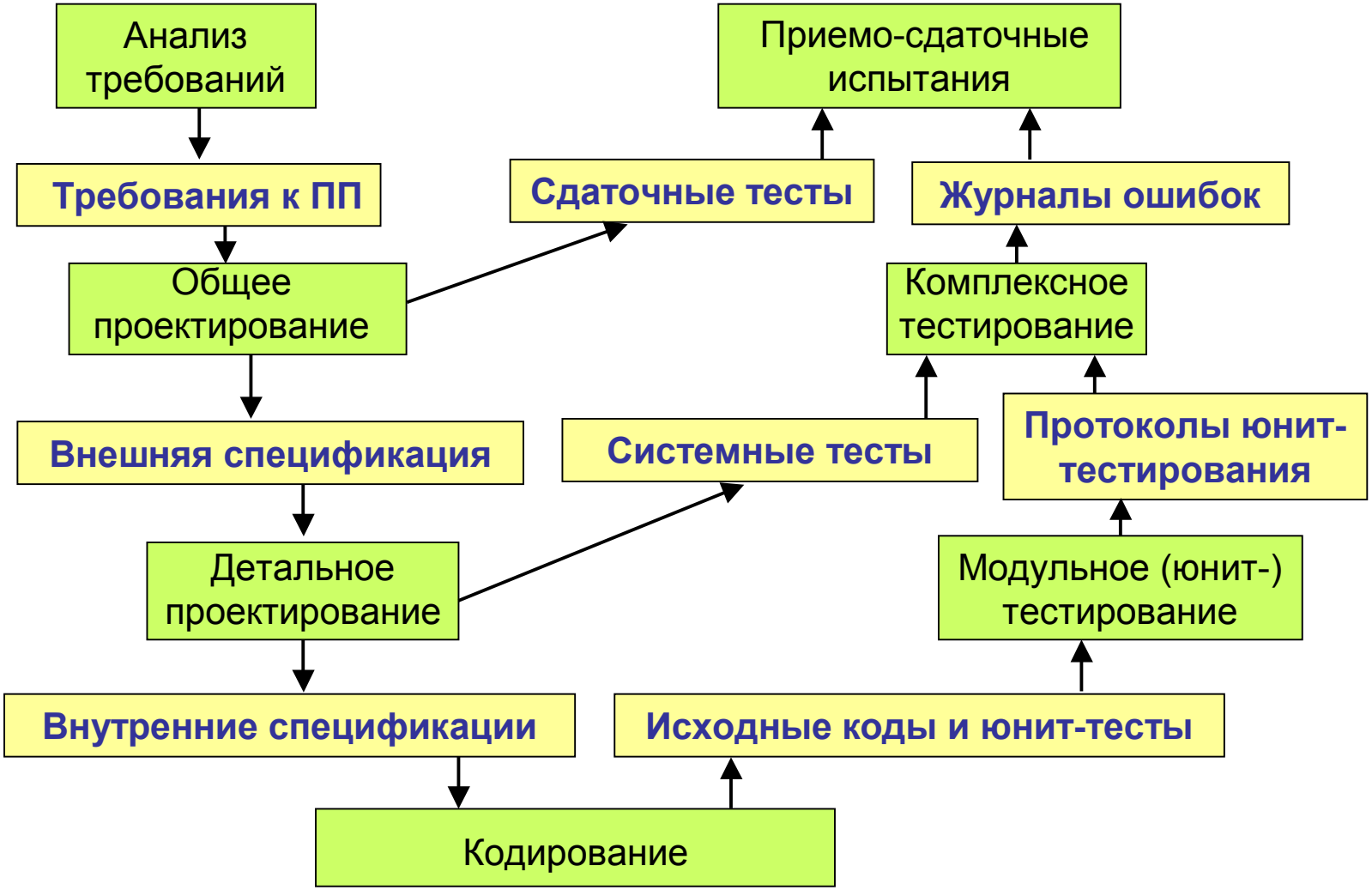
1. Marketing (or Customer) Requirements – Требования
2. Software Requirements Specification – Внешняя спецификация
3. Project Development Plan – Календарный план
4. Software Design Document(s) – внутренние спецификации (результаты общего и детального проектирования):
   UML-диаграммы, описания интерфейсов, алгоритмы
5. System Test Specification – Описание системных тестов
6. Alpha Evaluation Plan – План альфа-тестирования
7. Beta Evaluation Plan – План бета-тестирования

В каждой корпоративной технологии имеется свой перечень ПД и их шаблоны, напр. в MSF:

- **Концепция** (Vision Document) = док. №1 + черновик №3
- **Project Plan** = №2 + №3

Исходные коды ПП тоже относятся к проектной документации

Место проектных документов в ЖЦ



Внешняя спецификация

Синонимы: функциональная спецификация, техническое задание

Это – основной *отчуждаемый* проектный документ, пишется системными аналитиками:

- *Контракт* между разработчиком и заказчиком
- Отправная точка для других проектных и эксплуатационных документов
- *Технический паспорт* ПП на этапе его сопровождения

Его общая структура:

1. Общая характеристика

   Цели, требования, ограничения

2. Информационная и функциональная структура
   - ТЗ на дальнейшее проектирование и реализацию - для программистов
   - Описание внешнего интерфейса - для заказчиков
   - Обоснование достижимости (feasibility) поставленных целей

3. План тестирования

   План приемо-сдаточных испытаний (acceptance test) и спецификация комплексного тестирования (system test) т.е. ТЗ на разработку сдаточных и системных тестов

4. Календарный план разработки проекта (опционально)

Внешняя спецификация: детальная структура

1. Общая характеристика
   - 1.1 Цели и назначение
     Рыночная ниша, потенциальные пользователи, основная функциональность
   - 1.2 Технико-экономическое обоснование
     Экономический эффект для потребителей и для производителя, сравнение с аналогами
   - 1.3 Требования и ограничения
     - Функциональные требования (что должен делать ПП)
     - Нефункциональные требования (уровень показателей качества (см. 1 лекцию: производительность, надежность, …), соответствие стандартам и т. д.)
     - Системные требования (конфигурация целевой платформы и пр.)
     - Чего не будет, хотя может тщетно ожидаться пользователем
2. Описание информации
   - 2.1 Потоки данных
     Пути преобразования данных от входных через промежуточные к выходным
   - 2.2 Содержание и структура данных
     Форматы входных / выходных / промежуточных данных

Внеш. спф: детальная структура (2)

3. Описание функций

3.1 Функциональная структура

- Описание функций, ключевых алгоритмов и математических моделей
- Разделение на части (подсистемы, модули; реже - процедуры, объекты) и взаимосвязи между частями
- Межмодульные интерфейсы

3.2 Внешний интерфейс

- Для продукта-приложения – интерфейс пользователя
- Для продукта-библиотеки – API (Application Programming Interface)

3.3 Системный интерфейс

- Сопряжение с аппаратно-программным окружением

4. План тестирования

4.1 Измеряемые показатели функционирования

- Методика испытаний на соответствие требованиям к показателям в п. 1.3.

4.2 Классы тестов

- Перечень тестовых испытаний на нормальное функционирование, предельные и аварийные ситуации и пр.

Ссылочные документы

- Перечень стандартов, статей, монографий и т.д., на которые есть ссылки в тексте

Эксплуатационная документация

Руководство пользователя (User manual, UM)

- Назначение, принцип работы
- Интерфейс пользователя
- Учебник для начинающих ("Getting started")
- Как преодолевать возможные трудности ("Troubleshooting")
- Глоссарий (толковый словарь терминов); индекс терминов, имен и сокращений

Справочное руководство (Reference manual)

- Для приложения - перечень команд интерфейса
- Для библиотеки – описание API (перечень форматов вызова модулей-функций в алфавитном порядке их имен)

Руководство программиста (Programmer's manual)

- Инструкция по установке, настройке, эксплуатации
- Как преодолевать возможные трудности ("Troubleshooting")

Современная тенденция: переход от печатных документов к цифровым:

- гипертекстовые системы: помощь (Help), описание API (с помощью Doxygen и др.)
- обучающие программы
- FAQ-форумы как форма интерактивной консультации и архива вопросов/ответов

Нормативная документация

Стандарт кодирования, который определяет:

- структуру программного кода (с целью хорошей модульности)
- шаблоны заголовков модулей и функций
- стиль кодирования (с целью ясности, читабельности), в том числе форматирование, требования к комментариям
  - Рекомендуется, чтобы комментарии занимали в среднем не менее 20% строк исходного кода
  - Хорошие комментарии содержат информацию о структуре и функциях программы в целом
    - с этой целью многие shareware и freeware исходные коды содержат более 50% строк комментариев
- соглашения об именовании объектов кода
- ограничения на использование конструкций языка

  (с целью переносимости или согласования с требованиями инструментальных средств)
- приемы предупреждения ошибок программирования

Описания процессов ЖЦ

Шаблоны проектных и эксплуатационных документов } Есть в MSF и RUP

Каждая проектная команда должна иметь свой стандарт кодирования

Фирмы, формальные и неформальные объединения программистов и т.д. публикуют свои СК, напр., GNU coding standards:
http://www.gnu.org/prep/standards_toc.html (May 2002)

Преимущества и недостатки цифровых документов

- Простота создания и редактирования
  - \+ Смерть профессии машинистки
  - \+ Множественные черновики = последовательные версии документа
- Быстрота пересылки и публикации
  - \+ Глобальная коммуникационная среда - Интернет
  - \+ Коллективное и распределенное редактирование
  - \+ Оперативная публикация изменений
  - – Сетевой спам
- Простота хранения и быстрота поиска
- Дешевизна копирования
  - \+ Доступность документа широким массам
  - \+ – Смерть копирайта – прав на интеллектуальную собственность
  - – Легкость плагиата
- Возможность создания гипертекстовой структуры
  - \+ Структура графа произвольного вида → гибкая навигация по документу и по совокупности взаимосвязанных документов
  - – Опасность запутанной структуры
- Возможность автоматической проверки орфографии и подсказок
  - – Грамотность должна быть не в машине, а в голове

Средства коллективного редактирования документов

Wiki-сайт проекта: его структуру и содержимое разработчики могут изменять сообща

- Коллективное создание общих, концептуальных документов
- Обсуждение идей и проблем
- Рабочие материалы (повестки дня, презентации, планы, …)
- Гипертекстовые ссылки между документами различного типа

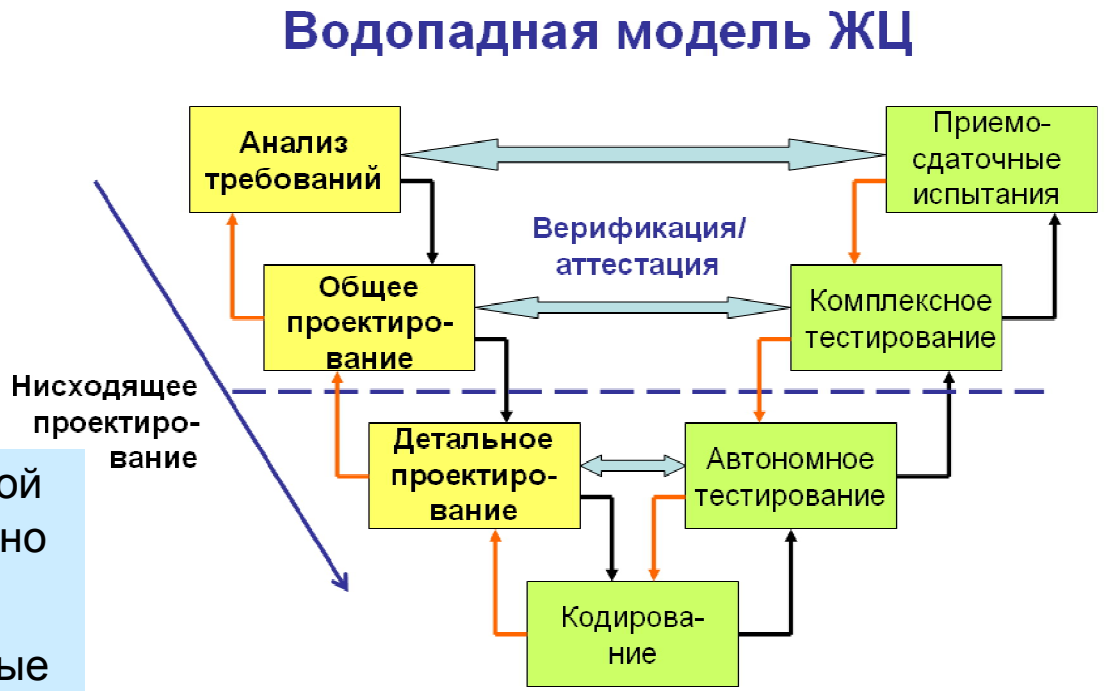
Есть десятки вики-движков: FlexWiki, Twiki, Clearspace, Confluence, …

Пример: Confluence – движок для корпоративных вики

- Отдельные сайты команд и проектов
- Импорт документов MS Office и PDF
- Параллельное редактирование
- Оповещения через эл. почту или блоги
- «Приборный пульт» (как в Jira)
- Используется в паре с Jira

Обзор процесса проектирования

Последовательность этапов проектирования - это серия «переводов» с одного языка на другой



- В итерационной и спиральной модели ЖЦ этапы неоднократно повторяются
- При первом проходе – базовые требования и базовый проект, дальше – уточнения и изменения

- Общее проектирование называют разработкой *архитектуры* ПП
  - ✓ результат – внешняя спецификация
- Детальное проектирование иногда называют *конструированием*

Работа с требованиями

Разработка требований

- ***Выявление*** **-** определение всех возможных источников и сбор требований из них
- ***Анализ*** с целью устранения противоречий и неоднозначностей в требованиях, их уточнения, классификации и ранжирования
- ***Описание*** в самостоятельном документе или в составе внешней спецификации

Управление требованиями

- ***Управление изменениями*****:** предложение изменений, анализ их влияния, утверждение, обновление планов
- ***Контроль версий*** спецификации требований – поддержание соответствия между версиями требований и другими артефактами ПП (документами, тестами и кодом)
- ***Контроль состояния*** требования: *предложено, одобрено, встроено, проверено*
- ***Трассировка -*** определение связей требования с другими требованиями и другими артефактами ПП
  - основа для анализа, какие части ПП затронуты изменением в требованиях
  - в идеале желательно прослеживать связь требования с конкретным фрагментом кода
  - связи представляются в виде графов зависимости или *матриц трассировки*

К.Вигерс. Разработка требований к программному обеспечению. М., 2004. – 576 с.

Характеристики требований

Требования к требованиям :)

- Ясность, недвусмысленность — однозначность понимания требований заказчиком и и разработчиками
- Полнота и непротиворечивость
- Прослеживаемость (traceability)
  - достигается путем дробления требований на отдельные, элементарные требования, присвоение им идентификаторов и создание трассировочной модели
- Тестируемость и проверяемость — наличие способов оттестировать и проверить данное требование.
- Модифицируемость - определяет процедуры внесения изменений в требования

Уровни формализации требований

- Неформальная постановка требований в переписке по электронной почте
  - ✓ хорошо работает в небольших проектах, при вовлеченности заказчика в разработку
- Требования в виде документа: в техническом задании (приложении к контракту), во внешней спецификации для разработчиков и т.д.
  - ✓ наиболее типичная форма
- Требования в виде графа зависимостей или матрицы трассировки
- Формальная модель требований для формальной верификации программы

Примеры требований

Функциональные требования

№ 47. Вывод экранной формы «Состояние счета клиента»

47.1 Параметры вызова формы:

47.1.1 Номер счета или фамилия клиента

47.1.2 Дата и время состояния; по умолчанию – текущий момент

47.2 Вид экранной формы: требование № 142.5

47.3 Реакция на ошибку ввода

47.3.1 Вид экранной формы сообщения об ошибке ввода: требование № 215.1

47.3.2 Тексты сообщений об ошибке ввода:

47.3.2.1 «Отсутствует счет с введенным №»

47.3.2.2 «Клиент с такой фамилией не существует»

<…>

Взаимо-связь

№ 142. Общий вид экранных форм ответов на запросы

142.5 Форма с тремя полями текстового ввода *<перечень элементов или эскиз>*

Нефункциональное требование

№ 229. Задержка ответа на запрос вывода экранных форм должна быть не более 5 с

229.1 Способ измерения задержки – утилитой Abc при одновременных запросах с 20 клиентских машин

Сложность работы с требованиями

Проблемы

- Большое число потенциальных заинтересованных лиц, требования которых нужно выявить и зафиксировать
- Разнообразие типов требований, каждый из которых требует специфического описания, своих атрибутов и степени детализации
- В сложных проектах - необходимость создания и поддержания сложной иерархической структуры взаимосвязанных требований
- Необходимость трассировать требования
- Требования неоднократно меняются в ходе выполнения проекта

Типичные причины срыва сроков и бюджетов проектов:

- не удалось полностью выявить требования заказчиков
- требования не были четко сформулированы
- не удалось отследить многочисленные изменения требований

Статистика работы с требованиями

Объемы

- Спецификация требований к ПП из 1 млн строк обычно занимает 4000-5000 стр.
- В НАСА на работу с требованиями тратится 30% времени и 15% бюджета проекта

Изменчивость

- Во время разработки требования изменяются в среднем на 25%, на что приходится 70-85% объема повторной работы над проектом

Последствия ошибок

- Устранение ошибки в требованиях на стадии сопровождения готового ПП обходится в 20 - 100 раз дороже, чем на стадии спецификации требований
- Исправления ошибок в требованиях, выявляемых на поздних фазах проекта, съедают 30 — 40% общей стоимости проекта ПП

Содержание процесса проектирования

Проектирование – это процесс преобразования задания на разработку (внешней спф) в задание на кодирование (внутренние спф)

Содержание процесса - построение и анализ *моделей* создаваемого ПП совместно с моделями управляемых объектов

Модель (в инженерном и научном смысле) - это упрощенное описание объекта или явления, в котором отражены те его свойства, которые существенны для целей его изучения или построения

От несущественных свойств абстрагируемся, т.е., ***модель есть абстракция***, и уместно говорить об уровнях иерархии абстракций

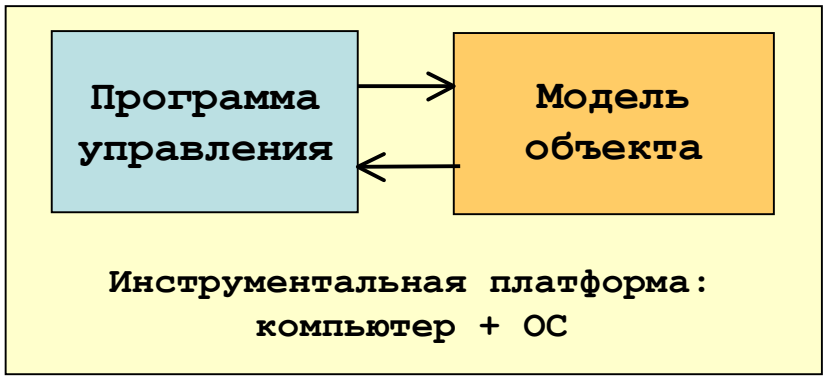
Чем выше уровень абстракции, тем более общая и краткая, обозримая, но зато и менее точная (менее детальная) модель

Виды моделей (вообще, не только ПП)

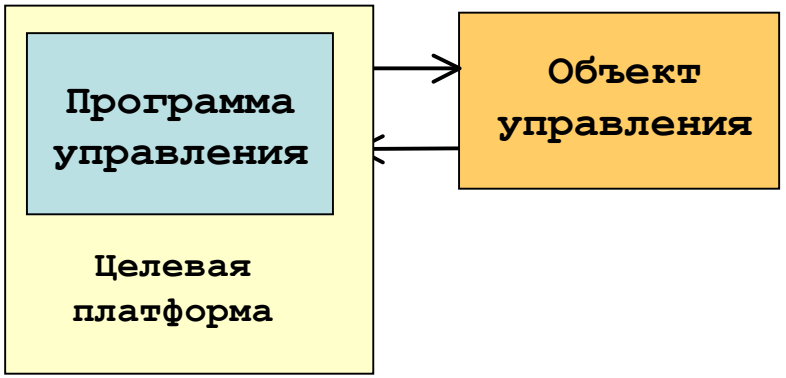
- Символические: описания, структурные схемы, математические (т.е., ***аналитические***, напр., системы уравнений и их символьные решения)
- Машинные аналитические модели: программы *символьного* (вычислительная алгебра) или *численного* решения математических моделей
- Машинные ***имитационные*** модели: программы имитации процессов реального мира

ПП в составе управляющей системы

- разрабатывается с использованием модели объекта управления



Во время разработки ПП



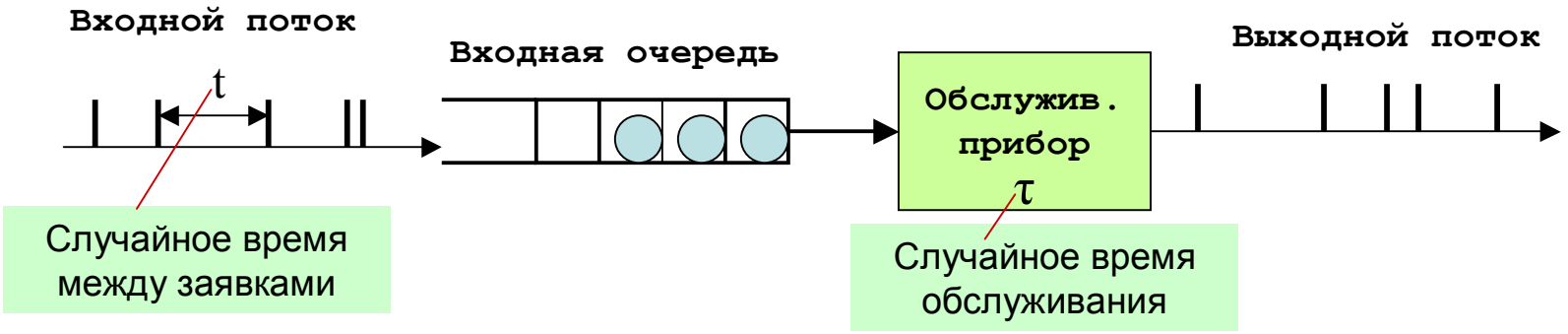
Во время эксплуатации системы

Тип объекта управления		Тип модели
➢ Транспортное средство	--	Механика (кинематика и динамика) движущихся объектов
▪ Ракета, робот		(системы дифференциальных уравнений)
➢ Технологический процесс	--	Физические процессы в непрерывных средах
▪ Нефтеперегонная установка		(системы дифференциальных уравнений в частных производных)
➢ Система связи и обработки инф.	--	1. Системы массового обслуживания (производительность)
▪ Цифровая АТС, сервер		2. Событийные модели (алгоритмы)
➢ Информационная система ERP	--	Модели бизнес-процессов:
▪ Банк, предприятие, …		- документооборот
➢ и пр.		- управление ресурсами (материальными, денежными, …)

Аналитическое моделирование

Обычно стараются строить модели и проводить эксперименты с ними в аналитической форме, решая уравнения моделей либо в символьном виде (MatCAD, Mathematica, ...), либо с помощью численных методов

Пример: простейшая система массового обслуживания (СМО)



Частный случай: система M/M/1:

Пуассоновское распределение времен t и τ: $F(t) = e^{-\lambda t}$, $F(\tau) = e^{-\lambda \tau}$

Среднее время между заявками = $1/\lambda$, среднее время обслуживания = $1/\mu$

Тогда средняя длина очереди $L = \lambda/\mu\ (1 - \lambda/\mu)$,

среднее время нахождения заявки в системе $T = L / \lambda$

(эти формулы – не для запоминания к экзамену)

Имитационное моделирование

Для сложных систем трудно или невозможно строить адекватные аналитические модели из-за большой размерности, стохастичности, нелинейностей, логико-семантических операций и т.д.

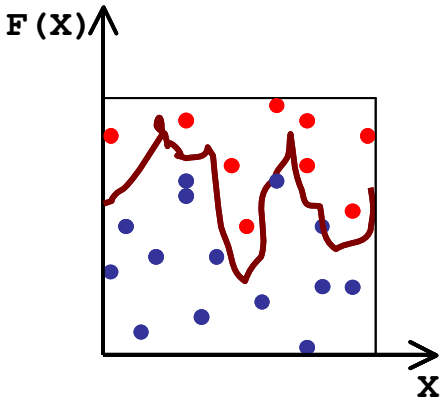
Даже если модель построена, может быть трудно получить решение в символьной или численной форме

В таких случаях применяют имитационное моделирование (simulation), когда

- структура моделируемой системы непосредственно представлена структурой модели
- процесс функционирования ее элементов выражен в виде правил и уравнений
- процесс в системе в целом имитируется на компьютере

Принцип имитационного моделирования:

- Модельное время продвигается дискретными шагами
- На каждом шаге вычисляются новые значения переменных
- Если в модели есть случайные величины, то производится много экспериментов, в которых их значения генерируются в соответствии со своими заданными законами распределения, затем результаты экспериментов усредняются
  - Отсюда устаревшее название: метод Монте-Карло



Вычисление интеграла методом Монте-Карло

Виды имитационных моделей

1. Дискретно-событийные модели
   - СМО: клиент-сервер, склад, магазин, сеть связи, ...
   - Реагирующие системы (продвигаемые событиями - event-driven)
2. Модели системной динамики
   - Макроэкономика, большие системы
3. Многоагентные модели
   - Толпа людей в аэропорту, эпидемия, потребительский рынок, ...

Недостаток - большая вычислительная трудоемкость имитации – отступает сейчас на второй план

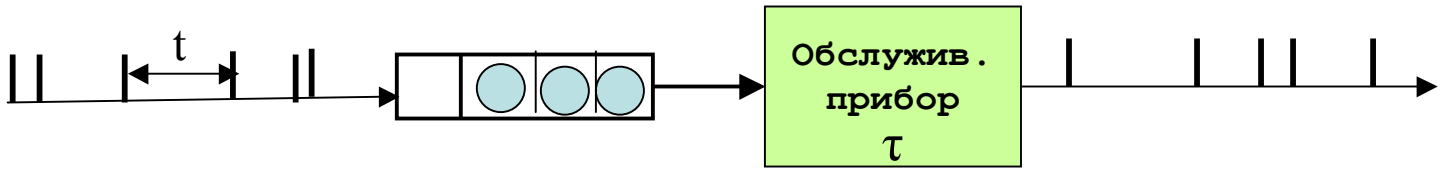
Инструментальные системы имитационного моделирования:

GPSS, Arena, DYNAMO, MATLAB, Simulink, PowerSim, AnyLogic, ...

Дискретно-событийные модели

- Система с дискретными состояниями
- События происходят в случайные моменты времени и приводят к изменению состояния системы
- Модельное время продвигается неравномерными шагами – каждый раз до ближайшего очередного события
- Имитируется большое число событий: 10^6 и более
- Накапливается статистика: среднее время нахождения в определенном состоянии, его дисперсия и т.д.

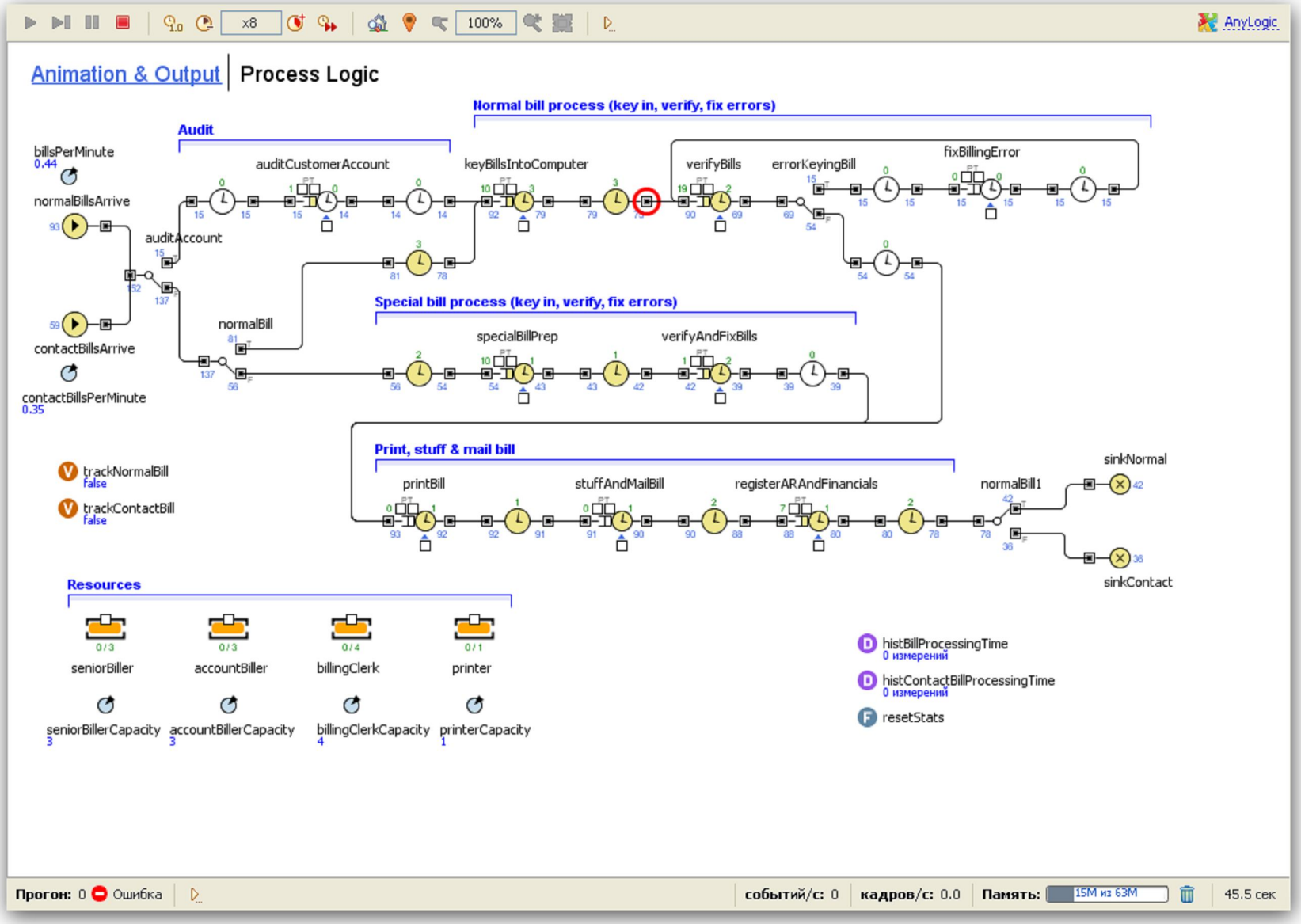
В СМО события - это появление заявок, начало и конец обслуживания



Основная структура данных – список будущих событий

- Преимущество перед аналитическими моделями СМО: гораздо проще учесть приоритеты заявок, любые виды распределений вероятностей, неоднородные потоки, нестационарные режимы и другие усложнения модели

Пример схемы модели СМО в AnyLogic



Многоагентные модели

- Динамика системы определяется не глобальными правилами и законами, а как результат активности множества *агентов* (автономных сущностей), взаимодействующих как друг с другом, так и со средой
  - ✓ Это принцип, аналогичный системам частиц в компьютерной графике
- Агентами могут быть люди, компании, активы, проекты, транспортные средства, города, сообщения и т.д.
- В отличие от заявок в СМО, агенты – активные сущности, обладающие индивидуальным поведением
- При разработке агентной модели инженер вводит параметры агентов, определяет их поведение, помещает в окружающую среду, устанавливает возможные связи и ограничения и запускает моделирование
- Индивидуальные поведения агентов образуют в совокупности глобальное поведение моделируемой системы

Пример: модель поведения пассажиров на станции метро в AnyLogic

Модели системной динамики
(Форрестер, 1970)

- Модель системы состоит из множества переменных, которые взаимодействуют друг с другом средством петель обратной связи
- Для основных фазовых переменных (так называемых *системных уровней*) пишутся дифференциальные уравнения вида:

$$\frac{dy}{dt} = y^+ - y^-$$

где $y+$ – положительный темп скорости переменной y, включающий в себя все факторы, вызывающие рост переменной y; y— отрицательный темп скорости, включающий в себя все факторы, вызывающие убывание переменной y :

$$y\pm = g(y1, y2, \ldots, yn) = f(F1, F2, \ldots, Fk) = f1(F1)\, f2(F2)\, \ldots fk(Fk)$$

где $F_j = g_j(y_{i_1}, \ldots, y_{i_m})$ – факторы, причем $m = m(j) < n,\ k = k(j) < n$ (число уровней)

Пример модели макроэкономики *(не для запоминания к экзамену)*

Системные уровни

1) население P;
2) основные фонды K;
3) доля фондов в сельском хозяйстве X;
4) уровень загрязнения Z;
5) количество невозобновляемых природных ресурсов R

Факторы

1) относительная численность населения P_P;
2) удельный капитал K_P;
3) материальный уровень жизни C;
4) количество пищи на человека) F;
5) удельный капитал в сельском хозяйстве X_P;
6) относительное загрязнение Z_S;
7) доля остающихся ресурсов R_R

Модель макроэкономики

Система дифференциальных уравнений для системных уровней:

$$\begin{cases} \frac{dP}{dt} = P(B - D) \\ \frac{dK}{dt} = K_+ - \frac{K}{T_K} \\ \frac{dX}{dt} = X_+ - \frac{X}{T_X} \\ \frac{dZ}{dt} = Z_+ - \frac{Z}{T_Z} \\ \frac{dR}{dt} = -R_- \end{cases}$$

где:

$B = B(C, F, P_P, Z_S) = C_B \cdot B_C(C) \cdot B_F(F) \cdot B_P(P_P) \cdot B_Z(Z_S)$ – темп рождаемости,

$D = D(C, F, P_P, Z_S) = cD \cdot DC(C) \cdot DF(F) \cdot DP(PP) \cdot DZ(ZS)$ – темп смертности,

$K_+ = K_+(P, C) = P \cdot K_C(C)$ – скорость производства основных фондов,

$X_+ = X_+(F, Q) = X_F(F) \cdot X_Q(Q) \,/\, T_X$ – прирост доли сельскохозяйственных фондов,

$Z_+ = Z_+(P, K_P) = P \cdot Z_K(K_P)$ – скорость генерации загрязнения,

$T_Z = T_Z(Z_S)$ – характерное время естественного разложения загрязнения,

$R_- = R_-(P, C) = P \cdot R_C(C)$ – скорость потребления ресурсов

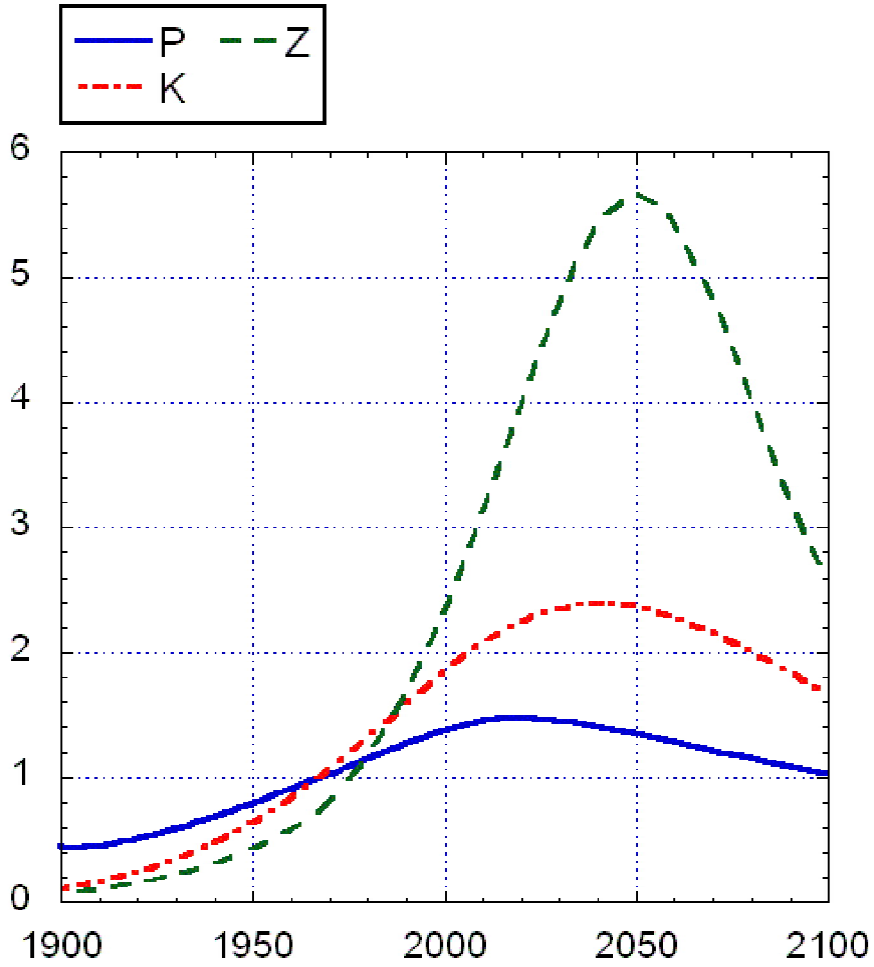
Все функции, обозначенные буквами с подстрочными символами (B_C, B_F, K_C и т.д.) - таблицы с линейной интерполяцией; они задавались экспертами в данной области

Уравнения для вспомогательных переменных:

$P_P = \frac{P}{P_N}$ относительная плотность населения

$K_P = \frac{K}{P}$ - удельный капитал, и т.д.

Некоторые результаты моделирования



Система уравнений решается классическим численным методом: одношаговая разностная схема первого порядка

P – население; K - основные фонды; Z - уровень загрязнения – все при условии сохранения параметров на уровне 1970 г.

Виды моделей программ и их представление в виде графов

Модели ПП описывают структуру будущей программы и/или структуру ее поведения

Структура – это внутреннее устройство чего-либо;
это состав целого из частей и отношения (связи) между частями

- Формально описанная структура - это частный случай математической модели
- Структуру обычно описывают в виде графа: вершины соответствуют компонентам (частям), дуги – связям
- Графовое и соответствующее *графическое* представление моделей способствует их наглядности, понятности (ср. чертежи в традиционных отраслях промышленности)

Основные традиционные виды моделей программ:

- Функциональная структура (или структурная схема)
- Алгоритмическая модель
- Информационная модель
- Событийная структура (модель состояний и переходов)

NB: В языке UML есть и другие виды моделей (диаграмм) - производные от этих базовых

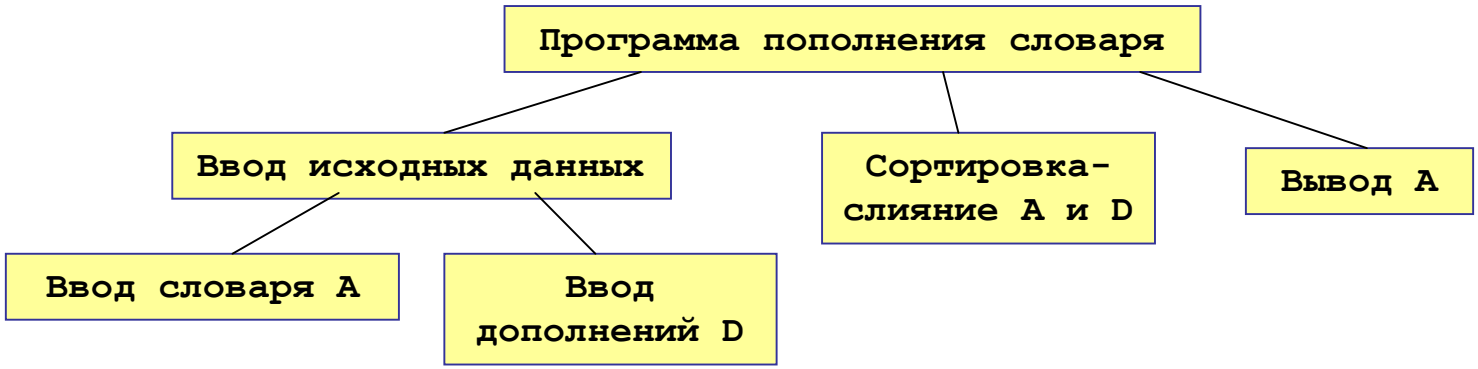
Функциональная структура

Описывает состав ПП из частей:

- *функциональных блоков* (ФБ) (подсистем, модулей, фрагментов кода)
- классов (объектно-ориентированная модель)

Виды отношений между частями:

а) часть-целое (отношение эквивалентности), тогда граф - дерево иерархии ФБ:



б) отношение вызывающий - вызываемый (для подпрограмм), тогда граф – дерево, гамак или более сложного вида

в) отношение наследования – для классов ОО модели

Функциональная структура - наиболее универсальная модель; она служит для *функциональной декомпозиции* проекта – разделения на части с целью лучшей модульности ПП и планирования разработки

Алгоритмическая структура

Описывает алгоритм как последовательность шагов, т.е. отношение предшествования во времени ФБ (отношение частичного порядка)

Изображается *блок-схемой алгоритма* (Flow Chart, или схема потока управления) или в виде *псевдокода*

- Блок-схемы нагляднее текстов, но устарели ввиду их громоздкости и сложности автоматической обработки
- Они вытеснены *псевдокодом* - текстовой нотацией, где управляющие конструкции (разветвление, цикл и т.д.) заимствуются из языка программирования (Паскаля, Си), а содержание ФБ записывается неформально, на естественном языке, как и в блок-схемах
- Есть и специально разработанные псевдокоды, напр. язык PDL (Program Design Language) фирмы IBM:

```
Start;
  Ввод файла дополнений D;
  If D пуст Then Stop;
  Ввод файла словаря A;
  Сортировка-слияние A и D;
  Вывод файла словаря A;
Stop
```

Алгоритмическая модель – основа для технического задания на программирование нетривиальных алгоритмов и для их документирования

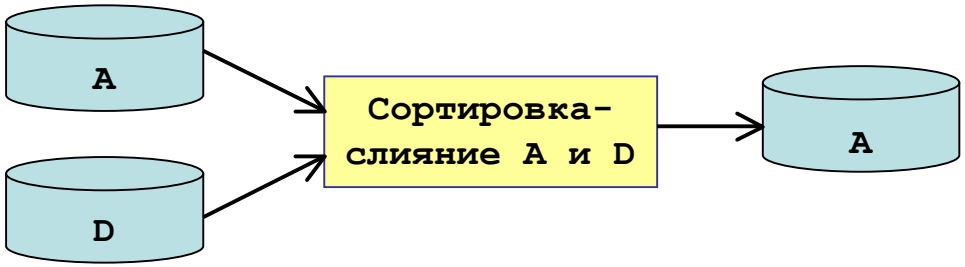
Информационная модель

Ее компоненты - порции информации: структуры данных, записи, файлы и, возможно, ФБ, их обрабатывающие

Два основных варианта:

А) Схема потока данных (Data Flow Diagram, DFD) описывает отношения на двух множествах: порций данных и ФБ

- Изображается двудольным графом; дуги – отношения двух типов: «быть входными данными» и «быть выходными данными»
- Дуги заменяют собой ФБ ввода / вывода или пересылки данных

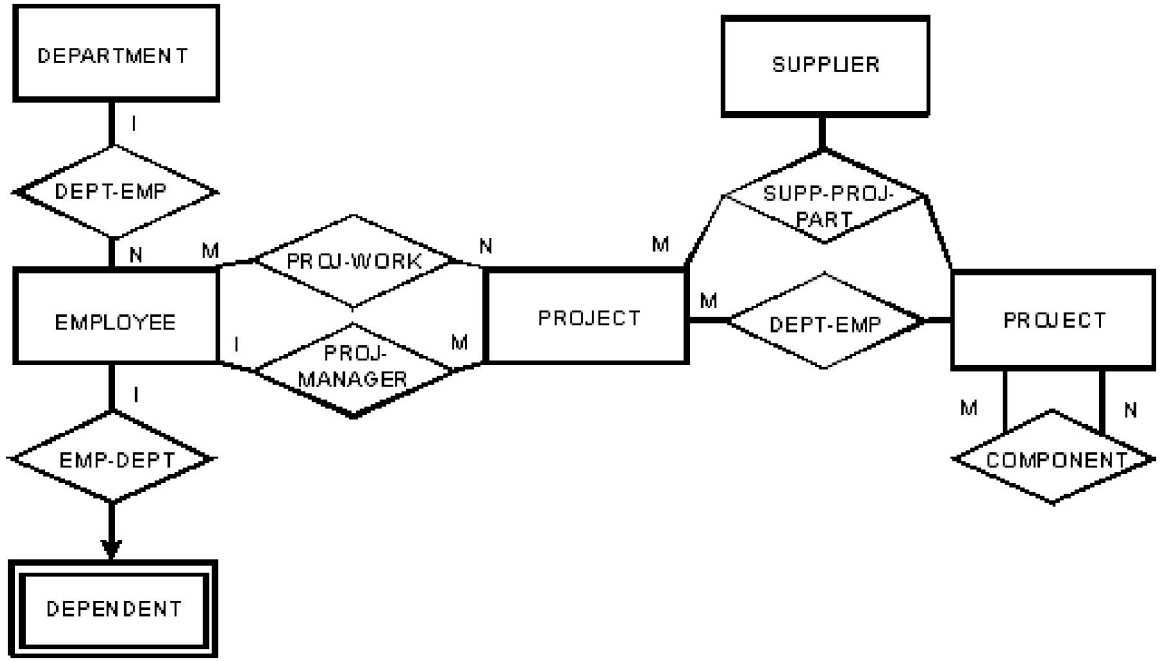


- Схемы потока данных хорошо подходят для описания тех *бизнес-процессов,* где вычисления просты, а ввод-вывод – интенсивный: напр., для автоматизированного документооборота (Doc Flow)
- В различных нотациях применяются при проектировании ERP-систем (см. лекцию 7)

Информационная модель (2)

Б) Инфологическая (информационно-логическая) модель описывает отношения между объектами предметной области ПП

Чаще всего это модель «сущность-связь» (Entity-Relationship, ER-model) – исходная для построения схемы базы данных



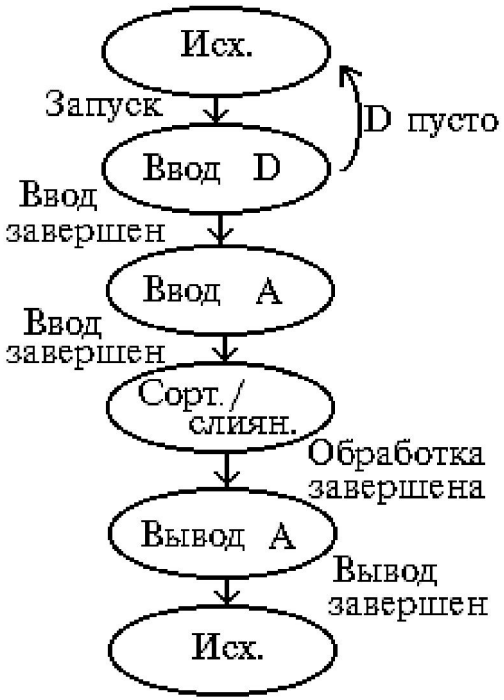
Пример: фрагмент ER-модели базы данных проектной организации

Такая модель используется, напр., в популярной CASE ERwin

Событийная структура

- непроцедурная модель поведения программы, аппаратуры или объекта управления

Компоненты – состояния системы, отношения – переходы между ними
Типичная модель – конечный автомат, изображается диаграммой состояний и переходов (State-Transition Diagram, STD), причем дуги нагружены событиями, вызывающими переходы и возможно, другими условиями..



- Состояния изображаются кружками или овалами, переходы - дугами
- В этой модели хорошо описываются реакции на внешние события и нестандартные ситуации
  - Из состояния "Ввод X" может исходить еще одна дуга - переход по ошибке чтения файла (возможная реакция - вывод сообщения и повторная попытка чтения)

- Событийная модель удобна для описания *реагирующих (responsive)* систем, к которым относятся многие системы реального времени
- В них состояния соответствуют ожиданию внешнего события, вызывающего переход в другое состояние

Заключение

- Разработка требований к ПП и управление ими – трудоемкий и ответственный процесс
- Проектирование ПП – это построение и исследование моделей ПП совместно с моделями объектов управления
- Для сложных объектов трудно строить и решать (выполнять) аналитические модели; альтернатива – имитационные модели
- Модели ПП – это графы их статической структуры или структуры их поведения, графическое представление которых – диаграммы (схемы) разного вида

Следующая лекция – 5 апреля

Дополнительные вопросы

1. Упорядочите перечень проектных документов по их важности на этапе сопровождения. Если ограничиться только одним из них, без какого документа невозможно какое-либо сопровождение ПП вообще?
2. Каков состав и функции современных Help-систем ? Что должна содержать минимальная Help-система прикладного ПП ?
3. Какие новые возможности электронный учебник по работе с ПП может дать по сравнению с бумажным? Предложите его возможную структуру.
4. Проиграйте примеры имитационных моделей в системе AnyLogic: http://www.xjtek.ru/anylogic/demo-models/logistics/
5. Какой, по-вашему, метод имитационного моделирования применяется в стратегических играх типа Civilization или SimCity?
6. В чем заключается эквивалентность ориентированного графа бинарному отношению на множестве его вершин ? Чему в этом смысле эквивалентен двудольный граф ?
7. Попробуйте восстановить семантику предметной области из схемы на слайде 34. Что означают пометки 1, N, M на дугах связей ?