ГБПОУ г. Москвы
«Первый московский образовательный комплекс»

Объектно-ориентированное программирование

Лекция 1. Введение

Лекции:

- Тузовский Анатолий Федорович – каф. ОСУ
- Рабочее место к. 307
- Консультация: Четверг с 17-18

Лабораторные занятия:

- Тузовский Анатолий Федорович
- Куркин Александр Александрович

План лекции

- Описание дисциплины
- Понятие подхода к созданию программного обеспечения (ПО)
- Структурный подход к разработке ПО
- Объектно-ориентированный подход к разработке ПО

Цель преподавания данной дисциплины

- На лекциях студенты должны получить знания по следующим темам:
  - объектно-ориентированный подход;
  - объектно-ориентированное программирование;
  - объектно-ориентированная среда .Net Framework;
  - объектно-ориентированный язык программирования C#.
- На лабораторных занятиях студенты должны получить навыки
  - Работы с классами с использованием технологии платформы .Net и языка C#.
  - Разработки ПО с использованием C#.
  - Разработки ПО с использованием среды Microsoft Visual Studio.Net.

Организация преподавания дисциплины

Распределение учебного времени

- Семестр 7 (осенний):
  - Лекции - 16 часов
  - Лабораторные занятия - 104 часов
  - **Всего аудиторных занятий - 120 часов**
  - Самостоятельная работа - 60 часа
  - **Общая трудоемкость - 180 часов**
- Семестр 8 (весенний)
  - Лабораторные занятия - 56 часов
  - Самостоятельная работа - 28 часов
  - **Общая трудоемкость - 84 часа**

Самостоятельная работа (88 часов)

- Изучение материала лекций.
- Выполнение примеров сделанных на лекции.
- Выполнение доп. заданий по ЛР

- Самостоятельная работа может выполняться:
  - на своем компьютере
  - вечером в лабораториях кафедры

Материалы курса

- в локальной сети по адресу

\\sinergy.main.tpu.ru\osu\Info\Дисциплина - ООП

- в Интернет сети

ftp.tpu.ru\OOP\

(id = osu; psw = stud)

  - Лекции – презентации
  - Лабораторные работы
  - Библиотека
  - Курсовой проект

Литература

- в электронном формате
  - в локальной сети
    \\Exilim\Info\ Дисциплина - ООП\Библиотека
  - в сети Интернет
    ftp.tpu.ru\OOP\Library
  - на русском
  - на английском
- книги по технологии .Net и C#
  - в библиотеке НТБ
  - в магазинах

Пояснение цели курса

Широкое распространение программного обеспечения

- Современный мир не может существовать без программ для компьютеров.
  - все государственные службы и инфраструктуры управляются системами, основанными на компьютерах.
  - большинство приборов и оборудования включают вычислительные компоненты и управляющее ПО.
  - отрасли развлечений (включая музыкальную индустрию), компьютерных игр, фильмов и телевидения активно используют ПО.
- В связи с этим жизненно важной для национальных и международных сообществ является технология разработки ***программного обеспечения***

Проблема разработки ПО

- Программные системы становятся все более крупными, более сложными (должны иметь новые возможности, которые ранее были не осуществимыми).
- Должны разрабатываться и устанавливаться (развертываться) быстрее.
- Должны быть более надежными.
- Должны иметь возможность быстро исправляться и развиваться.

Профессиональная разработка ПО

- Множество людей пишут программы –
  - ученые, инженеры, преподаватели составляют программы для обработки своих данных;
  - для любителей программирование является «хобби» - занимаются этим в свободное время (для интереса и развлечения).
- Основная часть ПО разрабатывается ***профессионалами*** (обычно командами) – они создают программы по заказу других людей (потребителей) для зарабатывания денег.
- ***Профессиональное ПО*** предназначено для использования другими людьми (не разработчиками).
- Оно поддерживается в рабочем состоянии (сопровождается), меняется и развивается в

ПО промышленного уровня

- Профессионалы разрабатывают ***ПО промышленного уровня*** для решения некоторых задач клиентов, которые используют их для целей своего бизнеса.
- Для профессионального ПО требуется
  - высокий уровень качества.
  - поддержка работоспособности в течении длительного времени.
- Программная индустрия в значительной степени заинтересована в разработке *ПО промышленного уровня*.

Стоимость, время и качество

- Наиболее важные показатели разработки ПО промышленного:
  1. стоимость разработки;
  2. сроки разработки (время);
  3. качество программного обеспечения.
- ПО должно быть разработано
  - за разумные деньги,
  - в разумные сроки и
  - иметь хорошее качество.
- ПО промышленного уровня
  - является очень дорогим, т.к. является очень трудоемким делом.
  - основными затратами на создание ПО является стоимость рабочей силы (зарплата специалистов).
    - измеряется в терминах затраченных на это человеко-месяцев.

История развития методологии разработки ПО

1. Начальный этап развития программирования.
2. Этап хакеров.
3. Структурный подход к разработке ПО.
4. ОО подход к разработке ПО.

Начальный этап развития программирования

- В 50-е годы фактически не было систематизированного подхода к разработке ПО.
- Программирование для больших ЭВМ (мэйнфреймов), имеющих ограниченные ресурсы
  - несколько десятков килобайт оперативной памяти
  - в качестве устройства данных – устройство чтения с бумажной ленты.
  - в качестве устройств управления и ввода данных использовались телетайпы, которые требовали большие затраты энергии для каждого нажатия

- **Ассемблерный язык рассматривался как решение кризиса разработки ПО.**
- В конце 50-х и начале 60-х годов начали появляться такие компиляторы высокоуровневых алгоритмических языков (FORTRAN, COBOL, BASIC).
- Алгоритмические языки абстрагировали 1 и 0 в
  - символьные имена,
  - высокоуровневые операторы,
  - блочные структуры
  - массивы и записи.
- Появляется этап Хакеров
  - основной показатель – индивидуальная производительность программистов.

Этап хакеров

- Этап хакеров – с начала 60-х до середины 70-х годов.
- Хакеры это программисты, которые могли создать большое количество программного кода, упорно работали и могли поддерживать работоспособность этого кода.
  - были очень способными – получали быстро работающий код на слабых компьютерах;
  - были очень упорными – тратили огромное количество времени на отладку;
  - создавали огромное количество кода (до 100 KLOC/год на языке FORTRAN);
  - часто разрабатывали «красивые» решения проблем.
- *В 60-х годах название “хакер” было положительным (хвалебным).*
- Организованный подход к созданию ПО отсутствует.

К концу 70-х годов слово “хакер” стало отрицательным

- Причина: хакеры переходили к разработке новых проектов и оставляли свой код для поддержки другим специалистами.
- Однако созданные хакерами программы требовали изменений:
  - выявлялась специальные ситуации, которые не обрабатывались программами;
  - мир менялся и программы тоже должны были совершенствоваться.
- Оказалось, что во многих случаях программы легче переписать, чем исправить.
- Появился термин «***поддерживаемый код»*** (unmaintainable code), который можно было просто менять
  - не поддерживаемый код стал называться хакерским.

Структурный подход к разработке ПО (СП)

Структурное подходы к разработке ПО

(Structured Development, SD)

- В конце 60-х годов стал появляться более систематизированный подход к разработке ПО.
- Размеры программ увеличивались и появилась идея, что они должны предварительно **проектироваться**.
- Начали появляться методологии, которые объединяли полученный опыт разработки ПО.
- **Проектирование ПО** стало видом

Структурная разработка

- Структурный подход к разработке ПО является наиболее важным достижением в технологии разработки ПО до 80-х годов.
- Это первый действительно систематизированный подход к разработке ПО.
- Он совместно с языками программирования 3 поколения (3GLs) способствовал:
  - значительному повышению производительности разработки ПО и
  - повышению надежности ПО (от 150 до 15 ошибок на KLOC к 1980 году).

Характеристики структурного подхода

1. Основная идея – п*рограмма состоит из большого количества алгоритмов разной сложности, которые совместно работают для решения имеющейся проблемы*
   – программа = набор функций + данные
2. Используется «**функциональная декомпозиция»**
   – разделение программы на иерархию функций.
3. Выполняется разработка «сверху-вниз».
4. **Появляются этапы анализа и проектирования ПО.**

Функциональная декомпозиция (ФД)

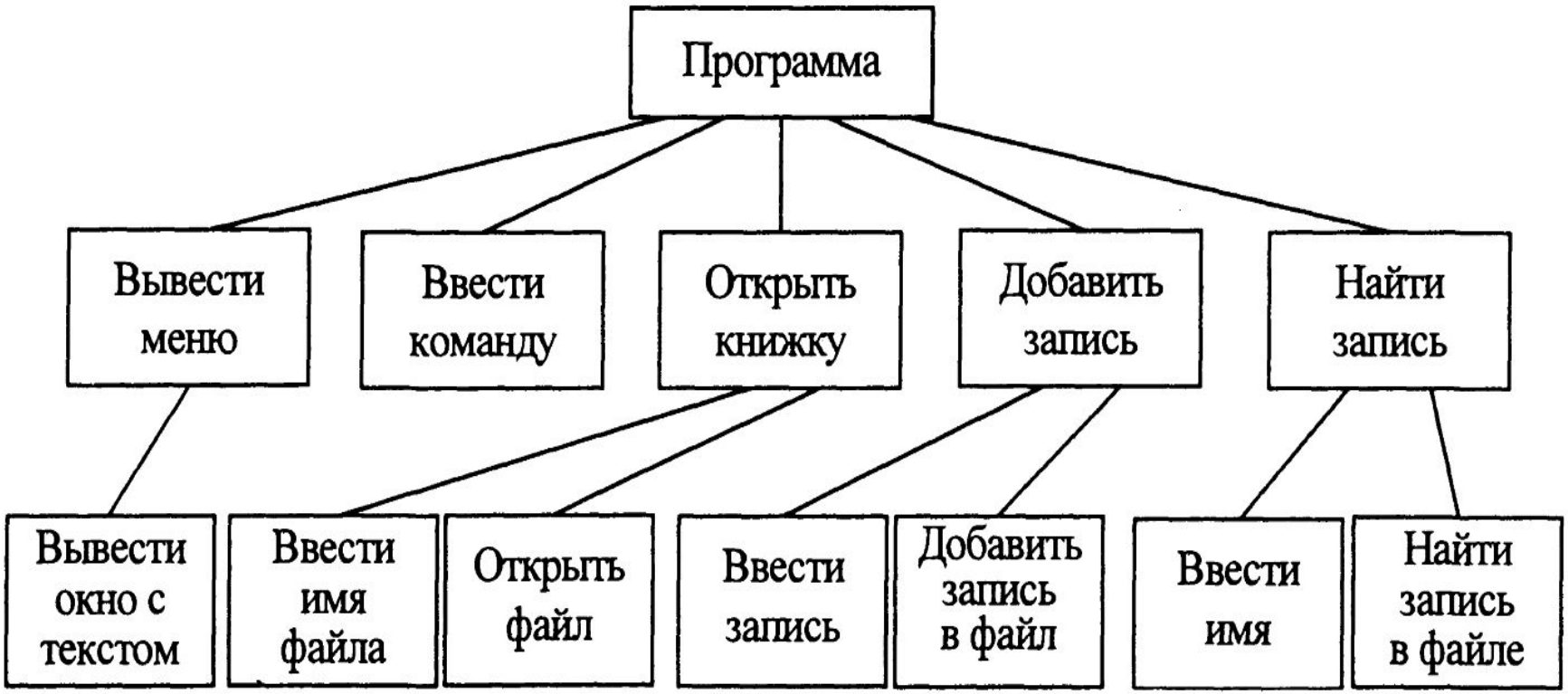
- *Программа рассматривается, как единый алгоритм*
- Базовый принцип ФД – «разделяй и властвуй».
- Программа разделяется на древовидную структуру функций:
  - функции верхнего уровня (расположенные в верхней части) вызывают множество функций нижнего уровня, которые выполнение части функциональности.
  - конечные функции (вершины, листья) являются ***атомарными функциями*** (простейшими)
    - не могут подразделяться далее.

Функциональная декомпозиция (2)

- Такой представление программы намного больше подходит к разработке программ для ***научных задач***, чем, например, к разработке программ для ***информационных систем***.

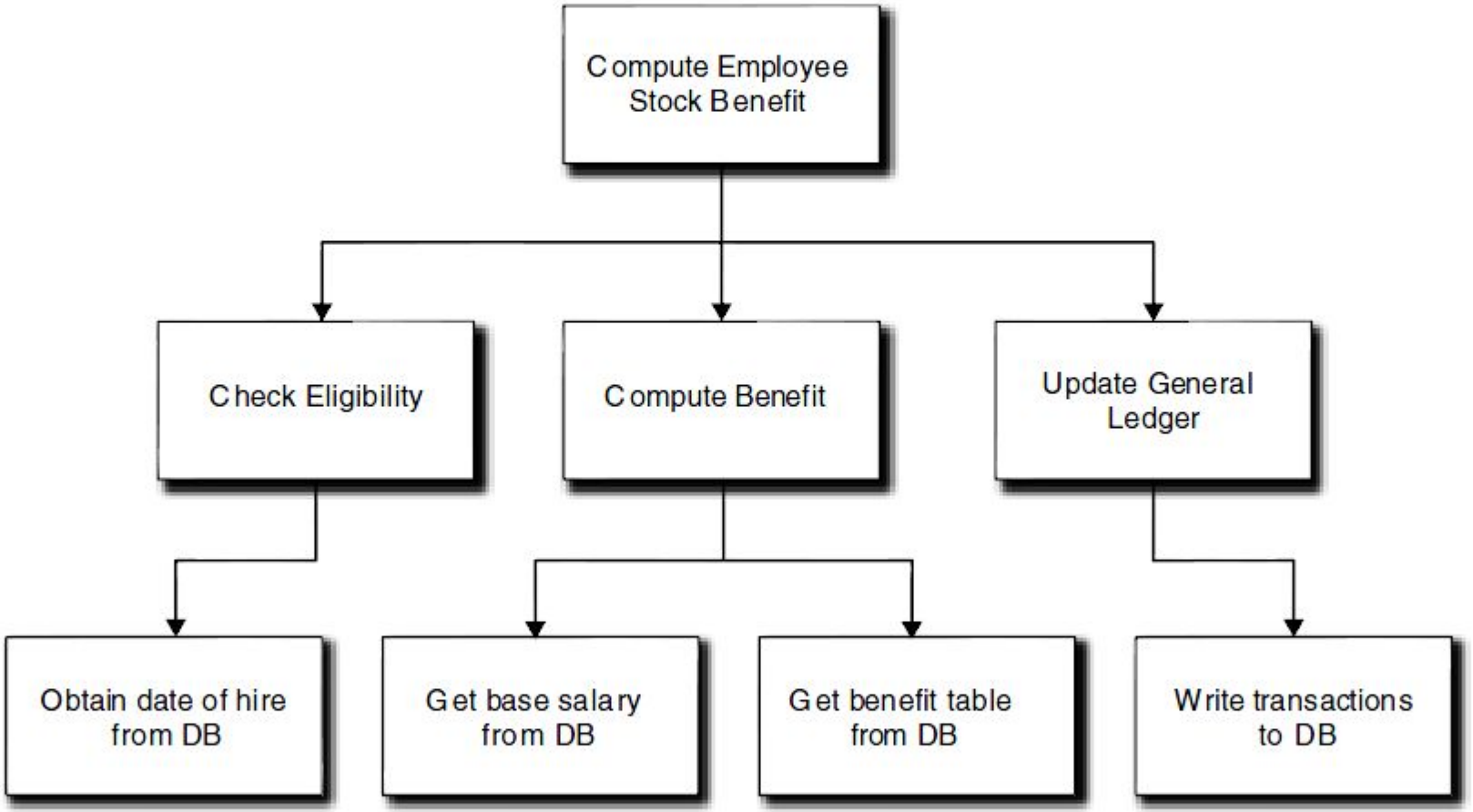
Пример функциональной декомпозиции

- Функциональная декомпозиция системы «Записная книжка»



Пример функциональной декомпозиции

- Пример функциональной декомпозиции задач вычисления выплат сотрудникам от их доли акций компании на более управляемые части.



Достоинства функциональной декомпозиции

- Функциональная декомпозиция была мощной и привлекательной.
- Она была привлекательной, так как объединяла, такие понятия, как:
  - **функциональная изоляция** (например, ветви дерева и детали разделенных функций),
  - **программирование на основе контрактов** (programming by contract) – разделенные функции предоставляют сервисы для клиентов более высокого уровня,
  - **план выполнения нисходящей разработки**, после описания дерева декомпозиции функций,
  - **прикладные программные интерфейсы (API)** в форме сигнатур процедур;
  - **укрупненные средства навигации преимущественного в глубину** (depth-first navigation) для анализ потока управления;
  - **многократное использование** путем вызова одних и тех же базовых функций из разных мест программы (из разных контекстов).

- В общем, это было очень умный способ решения

Недостатки функциональной декомпозиции

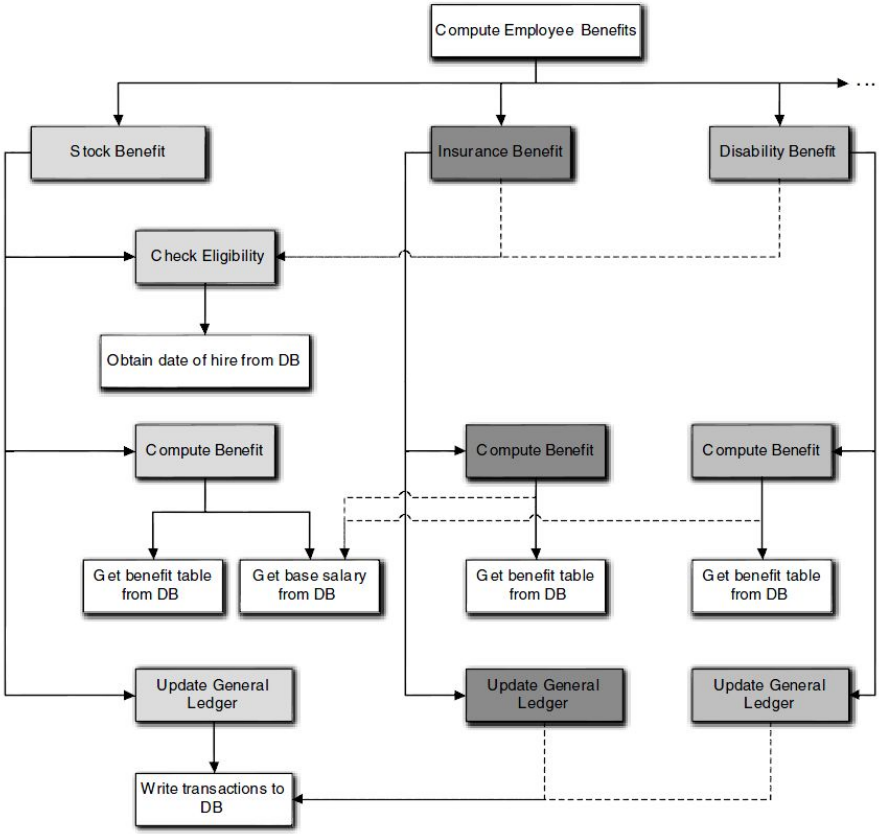
- В конце 70-х годов стало ясно, что SD привел к появлению нового набора проблем, которые никто не ожидал.
- Алгоритмы научных расчетов обычно меняются редко или меняются полностью с использованием более совершенных алгоритмов.
- ***В программировании бизнес-приложений правила, на основе которых работают программы постоянно меняются и программа постоянно развивается.***
- В связи с этим приложения должны модифицироваться в течение всей их жизни, иногда даже в течении начальной их

Недостатки ФД

- Трудность внесения изменений
  - изменение функций верхнего уровня вызывало изменения большого числа функций более низкого уровня;
  - изменение функций нижнего уровня могло вызывать необходимость изменять функции верхнего уровня.
- Проблема избыточности
  - набор простейших функций достаточно ограничен, в связи с этим в разных функциях требовалось использовать сходные простейшие функции;
  - часто один и тот же набор простейших функций был в разных ветвях иерархии;
  - повышалась трудоемкость их кодирования;
  - одинаковые изменения требовалось делать в разных однотипных функциях, что повышало возможность

Возможное улучшение функциональной декомпозиции

- Дерево функциональной декомпозиции стало решеткой.
- Закрашенные задачи являются базовыми элементами для расчета разных доплат (benefits).
- Пунктирные линии показывают перекрестные ссылки из одной ветви декомпозиции к другой, для устранения избыточности.



Проблемы структурного подхода

- Наибольшая трудность для ФД – неявное знание контекста, которое появляется при выполнения обработки вглубь (depth-first processing).
- Высоко-уровневые функции являются просто набором инструкций (указаний) для низко-уровневых функций что-то сделать.
- Чтобы задать инструкцию что-то сделать, высокоуровневая функция должна
  - знать, кто будет это делать;
  - знать что он может это сделать;
  - знать что это следующая работа, которая должна быть выполнена в общем решении.
- Все это нарушало изоляцию функций
- Последнее требование указывает, что вызываемая функция должна понимать высокоуровневый контекст общего решения проблемы.

Объектно-ориентированный подход к разработке ПО (ООПх)

Объектно-ориентированные подходы к разработке ПО

- Данный период начался в начале 80-х годов и оказал влияние на почти все виды деятельности в области разработки ПО.
- ООП, более конкретно дисциплинированный подход к анализу и проектированию ПО, был только одним из множества инноваций.
- Прошло около десяти лет, пока ООА и ООПр развились до такого состояния, что стали универсальными.

Базовая философия ОО подхода

- Достаточно сложный по сравнению с структурным подходом к разработке ПО.
- ***Не очень интуитивно понятный в смысле выполнения вычислений*** (по сравнению с декомпозицией функций)
  - ***требует специального образа мышления***.
- Включает набор различных понятий,

Проблема сопровождения ПО

- В 70-х годах было проведено много статистических исследований того, как разные компании выполняют разработку ПО.
- В результате были полученные очень интересные результаты:
  - **большинство компаний тратили 70% своих усилий на сопровождение ранее созданного ПО;**
  - **работа по модификации существующего ПО часто требует в 5-10 раз больше усилий, чем первоначальная его разработка.**
- Стало понятно, что-то делалось неправильно в разработке ПО.

- Специалисты по ПО пришли к выводу, что ***не возможно исправить (улучшить) структурный подход***.
- ООП является тем подходом, который предназначен для решения проблемы удобства сопровождения.
- ***Основная целью ООА/ООПр – удобство сопровождения ПО.***

Идея ОО подходаПО

- Основная цель ОО подхода – удобство сопровождения ПО
  - в связи постоянным изменением требований.
- Сопровождать ПО становится проще если структура ПО будет имитировать инфраструктуру проблемного пространства.

Объектно-ориентированное прграммирование

- Объектно-ориентированное программирование это подход к разработке программ в виде совокупности объектов, являющихся экземплярами определенного типа (класса), которые взаимодействуют между собой путем обмена сообщениями.
- ООП – это специальные языки и технологии, которые позволяют быстро разрабатывать объектно-ориентированные программы.

Результат использования ООПх

- ***Оказалось, что программы разработанные с использованием ООП более удобными для сопровождения (поддержки).***
- В начале 80-х годов в ОО методологии начали уделять основное внимание тому, как ООА/ООПр могут использоваться для решения недостатков СП.

- ***Скорость (затраченное время) первоначального создания ПО является менее важным, в сравнении с производительностью сопровождения***.
- Другой очень приоритетной целью разработки ПО является его ***надежность*** .
- Одной из основных причин того, что обычная поддержка ПО является трудоемкой, связано с тем, что изменения стали настолько сложными, что они вносили новые дефекты.

Цель ООА/ООПр

- ***Цель ООПх – разработка более понятных, легко поддерживаемых и надежных приложений.***
- Базовым допущением ОО подхода является -- ПО постоянно меняются в течение жизни программного продукта.
- Если это не выполняется. (программирование научных задач), то ОО подход может быть лучшим использовать структурны подход к разработки ПО.

Основные виды деятельности в объектно-ориентированном подходе

- Основными видами работ в объектно-ориентированном подходе (ООП) являются:
  1. Объектно-ориентированный анализ (OOA),
  2. Объектно-ориентированное проектирование (ООПр, OOD),
  3. **Объектно-ориентированное программирование** (ООП, OOP)

Виды требований к ПО

- ***Функциональные*** – какие задачи ПО должно решать
- ***Не функциональные*** – как должно работать ПО
  - безопасность
  - производительность (время отклика)
  - соответствие стандартам
  - поддерживаемые протоколы взаимодействия
  - ...

Объектно-ориентированный анализ

- *Результат ООА – создание **решения для функциональных требований проблемы**, не зависящее от конкретной вычислительной среды.*
- ООА описывает представление потребителя о разрабатываемом решении (приложении
  - абстракция проблемного пространства (ПП) описывается в терминах структуры ПрО потребителя.
- Результат ООА описывается только в терминах потребителя, поэтому ***он может включать только функциональные требования***.
- Решение полученное в ООА не зависит от конкретной вычислительной среды, в которой данное приложение будет реализовано.

Объектно-ориентированное проектирование

- *Результат ООПр – детализация (доработка) решения полученного в результате выполнения ООА* ***для конкретной вычислительной среды***
  - ***для обеспечения не функциональных требований***
- В результате ООПр создается высокоуровневое описание проекта решения, приспособленного к базовым характеристикам используемой вычислительной среды.

Объектно-ориентированное программирование (ООПм)

- ООПм – ***детальное уточнение решения, полученного на этапе ООПр***, которое предоставляет тактическое решение для всех требований с использованием уровня ОО языка программирования.
- ООПм предоставляет решение для всех требований (функциональных и не функциональных) на тактическом уровне
  - например, конкретного языка, сетевых протоколов, библиотеки классов, технологий и т.п.

Последовательность получения решения

- Последовательность формирования решения:

Требования ->

Объектно-ориентированный анализ ->

Объектно-ориентированное проектирование ->

Объектно-ориентированное программирование ->

Компиляция (в макро-язык) ->

Исполняемый код

- При перемещении слева на право
  - уменьшается уровень абстракции и
  - увеличивается учет деталей компьютерной

Спасибо за внимание !