Лекция ТП(06.09.2019)
ТП - совокупность методов средств и процедур в разработке ПО
Система принципов для создания экономии выгодного прог продукта, который работает на заданной платформе. ТП соответствует методике(система принципов и способов организации процесса разработки). Две базовые методологии-структурное проектирование и объектно ориентированное проектирование.

1. Последовательная декомпозиция задачи с выделением отдельным процессов исходных и результирующих данных. Нисход и восход проектирование
2. Применение обьектных методов где элементы предметной области определяются через абстракции наделяемые соответствоваемым поведением.

Каждый технолог операция имеет входные данные(док и рабочие материалы, на основании которых создается программный продукт). Результат выведенный в доступной стандартном виде. Должно быть управление(набор инструкция нормативов стандартов, на основании которых понятно как должен работать процесс). Механизм(исполнитель операций)

Типы ПО

1. Автономное ПО - установлено на одном компьютере;
2. Встроенное ПО - часть уникального приложения использующего конкретную аппаратуру;
3. ПО реального времени - выполняются функции в течении минимального промежутка времени с минимальным временем отклика;
4. Сетевое ПО - взаимодействие через сеть.

Жизненный цикл ПО - период времени с момента появления идеи создания ПО до момента завершения поддержки ПО фирмой разработчка.
Стандарт- ISO/1Ec12207:1195

Процессы ПО:

1. Основные процессы (заказчик поставщик и разработчик): приобретение, поставка, разработка. Эксплуатация, сопровождение. Заказчик определяет потребности, которые должны быть реализованы. Далее заказчик соотносит желаемое с действительным. В разработке разработчик должен выполнить реализацию в соответствии с утвержденными требованиями. Использование ПО.
2. Вспомогательные процессы: документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, аудит, разрешение проблем.

3. Организационные процессы: управление, усовершенствование...

Стратегии конструирования ПО:

1. Однократный проход (водопадная стратегия):

1.1) Системный анализ - анализ предметной области подлежащей автоматизации, поиск и оценка аналогов. Появляется аналитическая записка

1.2) Анализ требований - определение форматов входных и выходных документов и основного функционала ПО. Появляется ТЗ(требования к используемому программному и аппаратному обеспечению. Ограничения, хар-ки входных и выходных потоков, требования по персоналу, временные хар-ки.

1.3) Проектирование

1.4) Кодирование

1.5) Тестирование - определяется правильность ПО, верификация, отладка

1.6) Сопровождение

Основные особенности - каждый след этап после завершения предыдущего.

2. Спиральная модель БОЭМА

Создается версионно

Планирование, анализ риска, констрирование, оценивание заказчиком

3. Инкрементная модель
4. Модель быстрой разработки приложений (RAD)

Каждый этап не более чем 60-90 дней, Коллектив из 2-3 человек. Бизнес-моделирование, моделирование данных, моделирование обработки, автоматическая генерация приложения, тестирование и объединение.

Схема процессов XP

Аналитик рассматривает пользовательскую историю с учетом возможностей разработчика

Программные продукты

1. Системные - ОС, оболочки, утилиты
2. Прикладные

2.1) Для разработчиков - CASE-средства, IDE...

2.2) ДЛя непрограммистов - проф и тд

3. Гибридные - автоматизированные системы реального времени

Эксплуатационные требования

1. Правильность
2. Универсальность
3. Надежность

4. Проверяемость
5. Точность результатов
6. Защищенность
7. Программная совместимость
8. Аппаратная совместимость
9. Эффективность
10. Адаптируемость
11. Повторная видимость
12. Реентерабельность

Требование к ТЗ:
1. Введение
2. Основания для разработки
3. Требования к программе или программному изделию
4. Требования к программной документации
5. Технико-экономические показатели
6. Стадии и этапы разработки
7. Порядок контроля и приемки

13.09.2019
Особенности этапа проектирования(итерационный процесс, при помощи которого требования к программной системы транслируется в инженерное решение, дается концептуальное представление с последующим уточнением в конечном итоге осуществляется трансляция на выбранный ЯП)
2 стадии проектирования
1. Предварительное проектирование (требования к ТЗ, результатом является определение архитектуры ПО, выделение обрабатываемых и результирующих данных)
2. Детальное проектирование (разрабатывается структура программы, форматы данных, алгоритмы работы программы)

Если есть визуальная составляющая-осуществляется проектирование пользовательского интерфейса, определяются хар-ки и формы.

Архитектура ПО. Архитектура-строение программного продукта, те представление как система с взаимодействующими элементами. В качестве элементов могут выступать системы более низкого уровня, программы и модули. Выбор осуществляется на основании принятия след решений:
1. Организация программной системы
2. Выбор структуры элементов системы
3. Поведение этих элементов
4. Построить структуру из элементов
5. Определить сущность элемента

Выделяют два вида архитектур ПО:

1. Однопользовательское (программы, пакет программ, программный комплекс, программные системы)
2. Многопользовательское (системы клиент-сервер)

Структуры для описание архитектуры:
1. Логическая
2. Модульная
3. Процессная
4. Физическая

Типы пользовательского интерфейса:
1. Примитивный
2. Меню
3. Со свободной навигацией
4. Прямого манипулирования

Выбор методологии проектирования:
1. Структурная декомпозиция
2. Объектная декомпозиция

Модели системного структурирования(разбиение системы на отдельные компоненты:
1. Клиент-сервер
2. Трех-уровневая структура
3. Модель абстрактной машины

Части модуля связанны по данным

Связность модуля:
1. Коммуникативная связность
2. Последовательная
3. Функциональная

Сложность или стоимость может определяться след способом:
1. Мера длины по Холстеду
2.

20.09.2019 Лекция ТП
Высота структуры определяет насколько далеко находится друг от друга основной модуль системы и модуль, выполняющий внутренние функции. Ширина определяет кол-во модулей систем, которые могут быть либо должны быть запущены одновременно.
Существует две методологии проектирования структуры ПО:
1. Структурная
2. Объектно-ориентированная

1. Методы нисходящего проектирования, восходящего, расширения ядра. Все методы базируются на том что существует иерархия модулей, которые связаны по управлению. Модули более высокого уровня обращаются к модулям низкого уровня. Любая система - древовидная структура. В случае нисходящего проектирования средством разбиения задачи на более мелкие являются функциональные характеристики. На начальном этапе программный продукт рассматривается как единый блок, способный принимать данные и генерировать требуемый результат. Затем разрабатывается укрупнённая структура. Затем действия применяются рекурсивно к каждому новому блоку, до тех пор пока разбиение не стане не целесообразным, те будут получены неделимые компоненты (если он выполняет какую-то одну известную функцию, либо алгоритм его работы достаточно просто описать). Метод них проектирования положен в основу стандарта ISO12207-2003. Особенность - программирование, проектирование и отладка могут выполняться параллельно, что позволяет минимизировать трудоемкость, избежать существенных переделок. Нисходящее проектирование реализуется через две стратегии: пошаговое уточнение Дейкстра, стратегия анализа сообщений Йоден и Майерс. Основное внимание уделяется проектированию корректной структуры программы. 2 легко осуществить проверку корректности каждого имеющегося процесса. Основной недостаток пошагового-в случае обнаружения неточностей на заключительных этапах требуется пересмотрение всех вышестоящих ходов. Чаще всего реализуется с использованием псевдокода либо при помощи управляющих конструкций.

Метод анализа сообщений предполагает, что разбиение на элементы осуществляется на основе данных, которые необходимо обработать. Используется для структуризации обработки информации и на основе информации циркулирующей в системе. Сток связан с внешней средой по выходным данным. Преобразователь основная часть программы требуемая для реализации основного алгоритма функционирования. Каждый из элементов показывается как вершина, дуги определяют передачу данных между отдельными процессами.

2. Особенностями восходящего проектирования является то что элементы верхнего уровня связаны с взаимодействием с пользователем либо с другими внешними программами проектируются в последнюю очередь. Подъем будет до тех пор пока не будет получена конечная программная система. Метод целесообразно использовать в следующих случаях: существуют разработанные модули как модули нижнего уровня, когда известно что простые модули могут потребоваться во многих частях программы.
3. Выделение некоторого подмножества вспомогательных функция, формируемых из технического задания. Затем создаются функции, они

составляют ядро программы. Затем расширение ядра.

Проектирование структуры

Каждый процесс в дальнейшем может быть уточнен и все процессы разобьются на исполнительные и управляющие. Исполнительные процессы наделяются конкретными алгоритмами работы с целью воздействия на некоторое хранилище данных. Управляющие процессы определяют схему перехода между отдельными вычислительными процессами с возможным изменением состояния ПО. Для управляющих процессов строят конечный автомат либо граф. Для проектирования структуры существует широкий спектр методологии с возможностью описать поведение системы.

Методология SADT

Отражается основные функции. Основные особенности

1. Универсальность
2. Простота
3. Поддержка коллективной работы
4. Использование на ранних стадиях разработки
5. Сочетается с другими структурными методами проектирования

Реализуется через стандарт IDEF0.
«M есть модель системы S, если M может быть использована для получения ответов на вопросы относительно S с точностью A»

Это древовидная структура, где верхняя диаграмма является наиболее … Входная стрелка процесса всегда слева, выходная стрелка - справа. Стрелка сверху - стрелка управления(правила управления). Стрелка снизу - механизм(устройства или человеческий ресурс, выполняющий данные процесс). Снизу так же есть стрелка вызов - возможность передачи воздействия от текущего процесса к другому процессу системы. Детализация выполняется до тех пор пока на диаграмме имеется а) более 3-5 блоков б) какой-либо из блоков имеет более 5 входов воздействия.
Связи
Потребительская связь - результаты одного блока - механизм для другого блока
Функциональная связь - выход одной функции является воздействием другой
Логическая связь - между двумя процессами, которыми могут выполняться параллельно с целью завершения общего процесса
Коллегиальная - между процессами использующими общее управление
Ресурсная - между процесса с общими ресурсами
Временная - между процессами которые должны выполняться одновременно до или после запуска той или иной функции.

Диаграммы потоков данных

Описывается асинхронный процесс преобразования информации с момента входа в систему до момента выхода пользовател. Исходя из нотации выделяются след типы узлов.
Внешняя сущность - заказчик, поставщик, клиент, банк, ВУ
Функция или процесс - преобразователь входных данных в выходные в соответствии с алгоритмом. Имеет номер и имя, и кто будет выполнять процесс
Хранилище данных - абстрактное устройство для хранения информации. Тип устройства способ помещения и извлечения не детализируется. Имеет уникальное имя и может иметь номер
Поток данных - дуга соединяющая процессы

Лекция ТП (27.09.2019)
Две стратегии:

1. Функциональная - предполагает функциональное взаимодействие между системой
2. Декомпозиция в соответствии с функцией - рекомендуется использовать только на начальных этапах разработки, когда необходимо определить модель системы. Позволяет собрать исходную информацию о системе на основе перехода от ручных операций к автоматизированным (PPP декомпозиций)
3. Декомпозиция с уже известными стабильными системами - создание модулей на каждую подсистему либо компонент. Обобщенная модель системы есть комбинация моделей модули. Такая декомпозиция позволяет использовать компоненто ориентированный подход.
4. Декомпозиция основанная на отслеживании жизненного хода цикла - моделирование систем, непрерывно преобразующие свои входы в конечный продукт. Выполняется в соответствии с преобразованием входа
5. Декомпозиция по физическому процессу - выделение функциональных стадий, этапов завершения и шагов выполнение. Результат - последовательное описание системы не учитывающий ограничений, которые могут накладывать функции друг на друга.

Диаграммы активности

Входит в стандарт SADT (IDEF3). Является способом описаания процессов, позволяющим оценить последовательность выполнения операций необходимых для решения задач. Задача описывается в терминах работ. ДА позволяет определить сценарий выполнения работ. Основные элементы - работы которые на вход способны получать данные и лог элементы. Которые принято называть перекрестками. Лог элементы позволяют выполнять ход работ. Три типа элементов:

1. Перекресток ИЛИ
2. Перекресток И
3. Перекресток XOR

Синхронные и асинхронные элементы.

Проектирование данных. Диаграммы «сущность-связь». Соотношение хранимых и обрабатываемых в системе данных. Чаще всего используется нотация IDEFX. Согласно данной нотации выделяется два типа элементов: ER (enter relation). В качестве сущностей рассматриваются классы однотипных объектов, информация о которых должна быть учтена в модели, каждая сущность имеет имя. Имя определяется как существительное в ед числе. Внутри сущности описываются характеристики одного из объектов. Является экземпляром сущности. Характеристики принято называть атрибутами сущности. Ключевые и не ключевые атрибуты. Первичные и вторичные ключи.
Первичные ключи обязательно подчеркивается. Есть понятие внешнего ключа (ключи, физически принадлежащий другой таблице и переносимый в данную таблицу по связям). Связи это дуги, определяющие отношение одной сущности к другой либо к самой себе. Связи именуются глаголами по отношению к родительской сущности к дочерней. Связи бывают различными по мощностями:

1. Один ко многим
2. Один к одному
3. Многие ко многим

Сущность связь создается на основе интервьювирования с заказчиком.
Сущность - существительные, появляющиеся в интервью

Для описания поведения системы

1. Диаграммы переходов состояний - графическая форма конечного автомата. При получении тех или иных сигналов. Под сигналом понимайся любые данные приходящие из вне системы. Получив управляющее воздействие, система обязана выполнить некоторую операцию, затем перейти в новое состояние, либо остаться с текущем. ДС определяется набором вершин, которые являются состояниями системы, и набором дуг. Определяющих правила перехода между состояниями системы. Состояния именуются. Именование дуг именуется дробью, в знаменатели - сигнал или событие, которое выводит систему из текущего состояния, знаменатель - действие которое необходимо выполнить для перехода системы из одного состояния в другое. Состояния могут быть определены как составные состояния. На диаграмме в вершине определяется значок. Будет выделено как минимум 3 элемент:

Entry - действие которое необходимо выполнить при входе
Do - действия которое необходимо выполнять при нахождении системы в данном состоянии
Exit - действие которое необходимо выполнить перед выходом из системы

После выполнения детализации необходимо определить спецификацию каждой выполняемой функции. Если говорить о спецификации состояния система, то могут использоваться

- ПЕРТ-диаграммы - диаграмма анализа и корректировки плана. Вершина обозначает событие, а дуга выполняемую работу. ПОд событием понимается окончания одной и начало другой. Основным недостатком является невозможность определения совместных и параллельных работ. Отображает лишь необходимость и вероятность перехода между работами. В случае необходимости отображения параллельных моделей - сеть Петри
- Сеть Петри - модель системы разворачивающуюся во времени. Позволяет исследовать потоки данных циркулирующих в системе и динамику передачи управления между элементами системы. Основные элементы - вершины, определяющие выполняемую работу, дуги определяющие связь, переходы позволяющие регулировать передачу информации, и фишки - управляющие элементы влияющие на движение потоков данных. Наличие фишки является признаком наступления данного события и выполнение текущей работы, передаются по дугам. Если на пути встречается переход, то он работает как логическое И.

Лекция ТП 04.10.2019

Метод структурного программирования. Представление структур и данных единым набором основных конструкций. Существует 4 типа конструкций

1. Конструкция последовательности - используется когда два или более компоненты программы или данных следуют друг за другом строго последовательно.
2. Элемент выбора - наличие буквы О (результирующий компонент в определенный момент времени может получить толко одно из имеющихся значений, распределение значений без приоритета
3. Конструкция повторения - когда один элемент может повторяться от 0 до неограниченного количества раз.
4. Элементарная конструкция - простой компонент который не разбивается на элементы

Чаще всего при проектировании структур требуется выделять входные и выходные структуры данных. Выходные ставятся в соответствии с входными.

FLOW-формы

Форма записи алгоритма: следование, уловие или выбор, циклы

Диаграммы Несси-Шнейдермана

Условие записывается треугольником

Структурный схемы Константайна

1. Элемент структуры - мб определен как модуль подсистема или детализированный модуль, библиотека, область данных

На связях мб указан тип взаимодействия
Типы вызовов модулей

1. Последовательный вызов
2. Параллельный вызов
3. А вызывает В как сопрограмму

Связи бывают условными и циклическими

Диаграмма Варение-Орда

Строить модель программ и структур исходя из поведения входные и выходных данных. На практике используется крайне редко, если не нужно описать иерархию. Вложенность определяется количеством скобок.

HIPO-диаграммы

Лекция ТП (09.10.2019)

Объектный подход - принцип декомпозиция и струтуркная организация жлементов. Система представляет собой структуру из модулей связанных отношениями. В случае структурного подхода выполняется функциональная декомпозиция. Система представляется в виде иерархии взаимосвязанных функций. В случае объектной декомпозиции система разбивается на набор объектов, соотвтсвущих объектам реального мира, взаимодействие путем передачи сообщений. Второе отличие - объединение объекта как атрибутов так и поведения элементы. Третье отличие - структурная организация внутри модуля системы(структурная - набор функций, при ОО подходе иерархия выстраивается с помощью композиции и наследования)
Каждый элемент системы обладает собственным поведением,
Концептуальной основой является объектная модель. Основные элементы:

1. Абстрагирование
2. Инкапсуляция
3. Модульность (каждый модуль реализует собственную абстракцию)
4. Иерархия (Упорядочивание системы абстракций и расположение по уровням, иерархия классов, иерархия объектов)

Три доп элемента:

1. Типизация (ограничения на класс объекта и препятствующая взаимозаменяемости различных классов
2. Параллелизм
3. Устойчивость (свойство объекта существовать во времени)

Важным качеством является согласованность модели деятельности организации и модели проектирования систем начиная со стадии

формирования требований до стадии реализации. Объектный подход применяется на всех циклах. Методология объектно ориентированного анализа и проектирования. С точки зрения ОО анализа и проектирования любая сложная система представляет собой совокупность моделей, каждая из которых способна определить либо структурный либо поведенческий аспект модели. Для построения моделей применяется специальная нотация, называемая языком моделирования UML нотация. Является стандартизированной. UML входит в состав большинства средств разработки и проектирования ПО. UML пригоден при проектировании любого вида системы. Сам язык не зависит от моделируемой реальности, однако лучше применяется когда моделирование основано на рассмотрении прецедентов использования. Язык определен словарем и набором правил, позволяющих комбинировать слова. Модели языка UML разделяются на два класса:

1. Структурная модель (описывают структуру сущности предметной области, включая классы, атрибуты, интерфейсы. Применяется как логическое представление, представление реализации)
2. Модели поведения (динамические модели)

Функионирование сущности во времени, методы взаимодействия, изменение состояния сущности. Динамические модели учитывают масштабируемость системы, адаптируя под него процесс функционирования. Определяют топологию взаимосвязей между компонентами системы
Все модели подразделяются на два уровня:

1. Концептуальные модели (верхний уровень абстракции, описывающих с точки зрения понимания процесса)
2. Физический уровень (нижний уровень описания системы, определяет реальные сущности, реализующие структуру и поведение системы)

Взаимосвязь позволяет наиболее точно определить систему любой сложности

Три разновидности блоков в UML:

1. Предметы - основные элементы
    1.1) структурные элементы 0 статическая часть модели:

класс(описывает множество объектов), **интерфейс(**набор операций класса или компоненты, поведение предмета видимое извне), **кооперация**(определяет взаимодействие и является совокупностью ролей для обеспечения коррективного поведения, как структурное так и поведенческое поведение), **автор**(каждая роль требует от системы). **прецедент**(описание последовательности действий в интересах конкретного автора, при этом должен производить видимый для автора результат), **компонент**(физическая часть системы, соответсвующая набор интерфейса, исходные файлы, библиотечные файлы, исполняемые

модули), **узел**(физический элемент который существуетт во время работы системы и представляет собой некоторый ресурс на котором система размещает элемент с памятью либо без памяти)

1.2) предметы поведения: **взаимодействие**(поведение включающее в себя набор сообщений, которыми могут обмениваться объекты в некотором контексте для достижения целей) и **конечный автомат**(поведение определяемое последовательностью состояний в которых может находиться объект)

1.3) группирующие предметы - организационные группы в которых расположены элементы(основной элемент - **пакет**(распределение элементов по группам)

1.4) поясняющий предмет - набор документов, поясняющий элементы модели

2. Отношения - определено 4 вида отношений

2.1) зависимостей - пунктиром (семантическое отношение между двумя сущностями , изменение одной влияет на другую)

2.2) ассоциации - одиночная линия не имеющая направления (структурное отношение описывающее совокупность связей, показывает что объекты одноф сущности свзяаны с объектами другой сущности, так что можно перемещаться между объектами), мб реализована в двух видах - **агрегация**(особая разновидность представляющая связь часть - целое, когда один класс является коллекцией других, агрегация по ссылке зависит от времени существования класса) и **композиция** (физическое включение, является агрегацией по значению, четкое выражено владение)

2.3) обобщения

2.4) реализации - семантическое отношение которое позволяет реализовать функции через интерфейсы

3. Диаграммы

3.1) диаграмма вариантов использования - определение требований для системы

3.2) диаграмма классов

3.3) диаграммы поведения системы - диаграммы взаимодействия, процесс обмена соощений(диаграмма кооперацй, поведение системы в рамках различного использования(диаграмма деятельности), диаграмма состояния(моделирует поведение при переходе из одного состояния в другое)

3.4) диаграмма реализации

Диаграмма вариантов использования

Используется на начальных этапах разработки с целью определения спецификации и требований. Диаграмма описывает функциональное значение системы. Цель диаграммы - определить какие сущности каким образом будут взаимодействовать с системой. Правила:

1. Каждый вариант должен относится хотя бы к одному автору
2. Каждый вариант использования должен иметь инициатора

3. Каждый вариант использования должен приводить к результату

Определяем общие границы, формулируем общие требования и разрабатываем исходную концептуальную модель. В результате может быт ьразработна документация. Основаные компоненты - авторы и прецеденты, связанные набором отношений. Прецеденты определяют последовательность действий, взаимодействие с автором. При этом у автора есть внешняя сущность, использующая функционал системы, по сути один и тот же пользователь в разные моменты может играть различные роли в системе.

Виды отношений

1. Ассоциативное отношение - отношение медлу автором и прецедентом, устанавливает роль которую играет автор при использовании данного прецедента
2. Отношение расширения - отношение базового с другим элементов, когда базовый может расширять функциональность за счет доп условий
3. Отношение обобщения - указание фактов что некоторый вариант использования А мб обобщен до варианта испольщования В
4. Отношение включения - указывает на то что поведение для одного включается в качестве составного компоненты в другой вариант использования