Лекция 2

Основными требованиями к программному продукту являются эксплуатационные требования.

Они определяют характеристики разрабатываемого ПО, предъявляемого в процессе его функционирования.

Требования:

- Правильность. Т.е. функционирование в соответствии с ТЗ
- Универсальность. Обеспечение правильной работы ПО при любых допустимых входных данных, и защита от неправильных
- Надежность. Полная повторяемость результата даже в случае различного рода сбоев.
- Проверяемость. Возможность проверки получаемых результатов
- Точность. Получение результата с погрешностью не превышающие допустимую
- Защищенность. Обеспечение конфиденциальности информации
- Программная и аппаратная совместимость. Совместимость со всем
- Эффективность. Использование минимальное количество ресурсов, в частности технических средств для достижения результат
- Адаптируемость. Возможность быстрой модификации с целью приспособления к изменяющимся внешним условиям
- Повторная входимость. Возможность повторного запуска без перезагрузки системы
- Реентерабельность. Возможность параллельного использования несколькими процессами
- Рентабельность. Определяет, что задача, решаемая с помощью компьютера гораздо выгоднее, чем без него

Сложность многих программ не позволяет сразу определить четкие требования к ним. Чаще всего для перехода от идеи создания до ее реализации необходимо выполнить исследование предметной области, результаты которого заносятся в ТЗ. Техническое задание является начальным документов для разработки, входит в этап пред проектного проектирования и исследования и оформляется в соответствии с гост 19.201. При выполнении пред проектного исследования должны быть преобразованы нечеткие знания в точные требования, при этом разрешается 2 вида неопределенностей:

1) Могут быть неизвестны методы решения формулируемой задачи.
2) Неизвестная структура автоматизированных процессов. Требует досконального изучения существующих бизнес процессов

При снятии неопределенности второго типа:

а) Определить структуру и взаимосвязи

б) Распределить структуру между человеком и системой

в) Четко разделить. Выделить для решения задачи ресурсы

Разработка ТЗ

Это документ, в котором сформулированы основные цели разработки, требования ПО, сроки, этапы разработки, регламентирован процесс испытаний. В разработке ТЗ принимает участие представитель заказчика и исполнителя. Основными факторами, определяющими характеристики исходного ПО:

1) Исходные данные
2) Среда функционирования
3) Взаимодействие с другим ПО

ТЗ включает в себя:

- Введение. Наименование, краткая характеристика продукта, область применения, основное назначение. Должно быть доказано, что разрабатывать данный продукт необходимо. Для обоснования актуальности может быть приложен документ с информацией о аналогах и прототипах, создаваемой системы, с указанием причины несостоятельности аналогов.
- Основания для разработки. Определяются документы на основании которых ведется разработка, указывается предприятие, создавшее документы.
- Назначение разработки. В данном разделе указывается, какие функциональные и эксплуатационные характеристики, выгодные для пользователя.
- Требования к программному изделию.
    - Требование к функциональным характеристикам. Должны быть указаны все функции, состав и характеристики данных и результатов. Могут быть определены критерии эффективности.
    - Требования к надежности. Требуется указать как ведет себя система в нештатных ситуациях.
    - Требования к условиям эксплуатации.
    - Требования к составу и параметрам технических средств. Необходимы технические средства, требования по памяти, частоте и т.д.
    - Требования к программной совместимости. Указывается язык программирования, среда программирования, требования к ОС, требования к пользовательскому интерфейсу, требования по защите информации.
- Требования к программной документации. Обязаны определить перечень документов, с которыми будет поставляться программа.
- Технико-экономические показатели.
- Стадии и этапы разработки. Разрабатывается календарный график.
- Порядок контроля и приемки. Указываются виды проводимых испытаний и общие требования

Особенности этапа проектирования

Проектирование – итерационный процесс при помощи которого требования к ПО транслируются в инженерное представление программной системы. Сначала выполняется концептуальное проектирование с последующим уточнением, позволяющие сформировать этапы. Обычно проектирование выполняется в несколько ступеней:

- Предварительное проектирование. Позволяет сформировать архитектуру программы и определить набор обрабатываемых данных.
- На этапе детального проектирования, наборы обрабатываемых данных преобразуются в структуру данных, а описанные в архитектуре командные циклы в алгоритм работы программы.

По архитектуре ПО классифицируется:

- Однопользовательское ПО.
    - Программы. В любой непонятной ситуации пиши программное обеспечение
    - Пакеты программ.
    - Программные комплексы.
    - Программные системы.
- Многопользовательское ПО.
    - Система клиент-сервер

Программные архитектуры различаются в зависимости от используемого пользовательского интерфейса:

- Примитивный – один сценарий работы
- Меню – реализует множество сценариев работы, иерархическая цепочка
- Со свободной навигацией. Множество интерфейсов.
- Интерфейс прямого манипулирования. Все действия над графическими объектами

Стандарты:

- Проектирования
    - Конфигурация рабочих мест
    - Фиксация решений
    - Набор моделей
    - Совмещение работы
- Оформление документации
    - Комплектность
    - Согласование и утверждение
- Интерфейс пользователя
    - Оформление экранов
    - Оформление текстов помощи
    - Перечень стандартных сообщений

На этапе детального проектирования осуществляется 3 основных вида деятельности:

- Выполнение структурирование системы.
    - Модель хранения данных. В соответствии с данной моделью, разрабатываемое ПО предполагает набор подсистем, которое разделяет и обрабатывает данные и размещает в некоторой памяти. Чаще всего такие данные организуются в виде БД, а подсистема обеспечивает манипуляцию.
    - Модель клиент-сервер. Реализуется система, где данные могут располагаться в различных подсистемах, связь между подсистемами осуществляется при помощи локальной сети. Конфликты по доступу решаются либо сервером, либо доступом к сети.
    - Модель абстрактной машины. Представляется в виде многослойной структуры, каждый уровень которого, использует средства нижестоящего слоя.
- Моделирование управления системы
    - Модель централизованного управления. Одна из подсистем выступает в качестве системного контроллера и в ее обязанности входит управление работы другими подсистемами. Различают 2 модели:
        - Модель «Вызов-возврат». Есть основная программа, она вызывает подпрограммы, возврат осуществляется, когда завершается подпрограмма.
        - Модель «Системный контроллер». Предполагает, что существует программа, которая в любой момент времени может вызвать какой-либо из процессов системы. Процессы взаимодействовать не могут, только через системный контроллер.
    - Модель событийного управления. Предполагается, что любая система работает на основании возникающих внешних событий.
        - Широковещательная модель. Работает на основе системы очереди.
        - Модель управления прерываниями. Относительно вектора прерывания, выбирается обработчик прерывания. Все события сопоставляются определенному прерыванию. Все прерывания разбиваются на типы или группы. Типы прерываний образуют вектор прерываний. В векторе для каждого прерывания имеющего особый код ставится соответствующий обработчик.
- Декомпозиция системы на подсистемы. 1 – модель потоков данных. В основе лежит разбиение модуля по функциям. 2 – модель объектов основана на сцеплении сущностей, имеющих собственные наборы данных и наборы операций.

  Программный модуль – любой фрагмент описания процесса, который оформляется как самостоятельный программный продукт, пригодный для дальнейшего использования. Один и тот же модуль может входить в состав нескольких программ. В этой связи модуль рассматривается как средство борьбы со сложностью программ.

Модульность – свойство системы подвергаться декомпозиции на ряд внутренне связанных и слабо зависящих друг от друга модулей. Оптимальный модуль должен удовлетворять 2м критериям: снаружи он должен быть проще чем изнутри, и модуль должно быть проще использовать, чем построить.

Основной характеристикой модуля является информационная закрытость. Предполагает, что модуль внутри должен быть сложнее чем снаружи. Пользователю модуля должны быть известны подробности доступа функции модуля без подробностей их реализации. Модуль состоит из двух частей:

1) Интерфейс. Должна доступна пользователю. То, что известно снаружи.
2) Внутренняя структура модуля. Должна быть скрыта

Для организации связи между модулями и оценки степени их связи существует понятие связность. Связность – внутренняя характеристика модуля, определяющая мера зависимости ее частей. Чем выше связность, тем лучше результат проектирования. Для измерения связности используют понятия Сила Связности(СС). Существуют 7 сил связности. Каждая из них имеет вес, чем больше вес – тем больше связность.

Степени связности:

1) Связность по совпадению (0). Присутствует в модуле, где отсутствуют явные связи. Существуют, когда в программной системе изредка вызывают функции друг друга
2) Логическая связность (1). Возникает в модуле, где все функции взаимосвязаны на одной и той же операции.
3) Временная связность (3). Части модуля могут быть не связаны, но выполняются в одно и то же время.
4) Процедурная связность (5). Части модуля связаны порядком выполнения действий. Реализует некоторый сценарий поведения. Все связаны общим процессом
5) Коммуникативная связность (7). Части модуля связанны по данным, обрабатывают одну и туже структуру. Довольно технологично, так как в случае изменения структуры меняется один модуль.
6) Последовательная связность (9). Все функции модуля выполняются друг за другом. Выходные данной одной части используются как входные части другой части модуля.
7) Функциональная связность (10). Части модуля вместе реализуют одну функцию, без привлечения внешних обработчиков. Функционально связанный модуль содержит элементы, участвующие в выполнении одной и только одной проблемной задачи.

Сцепление модулей

Мера взаимосвязи модулей, их внешняя характеристика. Чем меньше сцепление, тем лучше.

Выделяют 6 типов сцепления (каждый из них имеет собственный вес):

1) Сцепление по данным(1). Предполагает, что модуль А вызывает модуль В, при этом все входные и выходные параметры вызываемого модуля — это простые элементы данных (скаляры).
2) Сцепление по образцу(3). В качестве данных возвращается структура данных.
3) Сцепление по управлению(6). Предполагает, что модуль А явно управляет функционалом модуля В. Решается установкой флагов/переключателей.
4) Сцепление по внешним ссылкам(5). Есть некий модуль А и В, они работают с общей областью.
5) Сцепление по общей области(7). В одном случае скаляр, в другом структура.
6) Сцепление по содержимому. Когда один модуль ссылается на содержимое другого модуля.

Рутинность – независимость модуля от предыдущих обращений к нему. Модуль называют рутинным, если результат обращения к нему зависит только от значения передаваемых в него параметров. Модуль называют зависящим от предыстории (нерутинным), если результат обращения к нему зависит от внутреннего состояния модуля.