ТиМП

УСЛВИЯ АВТОМАТА - 100% посещаемость, лабы

13.02.2016

Программные средства как продукт технологии программирования

Программа как формализованное описание процесса обработки данных. Программные средства.

Целью программирования является описание процессов обработки данных (по Ожигову). Данные – представление фактов и идей в формализованном виде пригодным для передачи и обработки неким процессом. Информация – смысл который придается данным при их представлении.

Обработка данных – выполнение систематической последовательности действий с данными, а данные представляются и хранятся на носителях данных. Совокупность носителей данных используемых при обработке данных называется информационной средой. Набор данных, содержащийся в какой либо момент в информационной среде, называется состоянием этой информационной среды.

Процесс – последовательность, сменяющих друг друга состояний некоторой информационной среды. Описать процесс – значит определить последовательность состояний информационной среды. Если мы хотим чтоб по заданному описанию требуемый процесс порождался автоматически, необходимо чтоб это описание было формализовано, такое описание процесса называется программой.

Программа составляется на удобном для человека формализованном языке программирования, с которого она автоматически переводится на язык

соответствующего компьютера с помощью другой программы, называемой транслятором.

Для освоения программы пользователем помимо текста требуется дополнительная документация. Программа или логически связанная совокупность программ на носителях данных, снабженная программной документацией, называется программным средством. Программа позволяет осуществлять некоторую автоматическую обработку данных, а программная документация позволяет понять, какие функции выполняет та или иная программа, а так же помогает лучше разобраться в самой программе и так далее.

Не конструктивность, понятие правильной программы

Ошибка – в программе имеется ошибка, если она не выполняет того, что разумно ожидать от нее пользователю. Разумное ожидание пользователя формируется на основе документации. Когда программа не соответствует своей функциональной спецификации, то она называется дефектом программы.

Надежность программного средства

Надежность программного средства - его способность безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью. Отказ – проявление в нем ошибки. Надежность измеряется вероятностью работы без отказов в течение определенного периода времени. При оценке степени надежность необходимо учитывать последние отказы.

Технология программирования как технология разработки надежных программных средств

Технологии программирования мы будем понимать совокупность производственных процессов, приводящую к созданию требуемого

программного средства. а также описанию этой совокупности процессов. Таким образом технологию программирования мы будем понимать как разработку программных средств включая в нее все процессы, начиная с момента зарождения идеи этого средства и в частности связанные с созданием необходимой программной документацией.

20.02.16

Программная инженерия – систематический подход к разработке, эксплуатации сопровождению и изъятию из обращения программных средств. Различия технологий программирования и программной инженерией заключается в способе рассмотрения и систематизации материала. Технологии программирования, акцент делается на изучение процессов разработки и соответственно методы и средства разработки используются здесь. В программной инженерии изучаются методы и инструментарии разработки в первую очередь, то как эти методы образуют процесс.

Технологии программирования и информатизация общества

Лекция 2

Источники ошибок программных средств

Интеллектуальные возможности человека

Три интеллектуальные возможности человека по Дейкстру

1) Способность к перебору – последовательное переключения внимания с одного предмета на другой с последующим узнаванием искомого объекта; Не более 1000 объектов.

2) Способность к абстракции – один из способов преодоления ограничения по перебору

3) способность к математической индукции – позволяет справляться с бесконечными последовательностями. Конченые выводы

Система – совокупность взаимодействующих друг с другом элементов. Программные средства – система. Понять систему (простая система) – осмысленно перебрать все пути взаимодействия между ее элементами. Сложная система – нельзя понять сходу.

Неправильный перевод как причина ошибок программного средства

Модель перевода

Основные пути борьбы с ошибками

1) сужение пространства перебора (упрощение создаваемых систем)

2) Обеспечение требуемого уровня подготовки разработчика

3) Обеспечение однозначности интерпретации информации

4) Контроль правильности перевода

27.02.16 Лекция 3

Общие принципы построения программных средств

Специфика разработки программных средств

Особенности

1) Неформальный характер к требования программных (постановка задачи). Понятие ошибки в нем (оно тоже неформализованное). Но формализован основной объект разработки программных средств. Таким образом содержатся определенный этапы формализации, а переход от неформального к формальному существенно неформален. Требования всегда неформальны (нет четких критериев).

2) Разработка программного средства носит существенно творческий характер. Нет точных временных рамок.

3) Программные средства представляют собой некоторую совокупность текста, а смысл этих текстов выражается процессами обработки данных, и действиями пользователей запускающих эти процессы.

4) Программные средства при своем использовании не расходуется и не расходует используемых ресурсов (не совсем верно (дата центр Яндекса отепляет целый город))

Жизненный цикл программного средства

Жизненный цикл программного средства – весь период его разработки и эксплуатации, начиная от момента возникновения замысла до момента прекращения всех видов его использования (включает все процессы создания и использования программного средства). Процессы разработки включают в себя определенные этапы

1) Стадия разработки программного средства. Состоит из следующих этапов

- Внешнее описание – это описания поведения с точки зрения внешнего по отношению к нему наблюдателя с фиксацией требования по отношению к его качеству. Внешнее описание начинается с описания требований к программному средству со стороны заказчика. Это документ, который формируется заказчиком.
- Конструирование программных средств – охватывает следующий процесс: разработку архитектуры программного средства; разработку структур программ; их детальную спецификацию. Объектная декомпозиция, иерархичная структура, описание взаимодействия модулей и так далее.
- Кодирование – создание текстов программ на ЯП и их отладку с тестированием программного средства.
- Аттестация – оценка качества программного средства. После этого этапа стадия разработки закончена

2) Стадия производства программных изделий

Программное изделие – экземпляр или копия снятая с разработанного программного средства. А изготовление программного изделия – процесс генерации и воспроизведения программ и программных документов с целью их поставки пользователю для применения по назначению, а производство это совокупность работ по обеспечению изготовления требуемого количества программных изделий в установленные сроки. Воспроизведение – снятие копий. Генерация – компиляция, получение программы.

3)Стадия эксплуатации

Охватывает процессы хранения внедрения и сопровождения

Параллельные фазы

- фаза применения – использование программного средства для решения практических задач, путем выполнения ее программ
- фаза сопровождения – процесс сбора информации о его качестве в эксплуатации, устранения обнаруженных в нем ошибок, его доработки и модификации, а также извещение пользователей о внесенных изменениях

Понятие качества программного средства

Качества программного средства – совокупность его характеристик, которая влияет на его способность удовлетворять заданным потребностям пользователей. Важно отметить что повышение качества программного средства по одному из таких свойств часто достигается за счет снижения качества других свойств, либо ценой изменения стоимости срока завершения разработки. Качество программного средства является удовлетворительным, когда оно обладает указанными свойствами такой степени, чтоб гарантировать его успешное использование.

Критерии качества программного средства

1) Функциональность – способность программного средства выполнять набор функций, удовлетворяющих потребностям пользователей.

2) Надежность

3) Легкость применения – это характеристика программного средства которая позволяет минимизировать усилия пользователей по подготовке исходных данных, применению программного средства и оценке полученных результатов. User Experience

4) Эффективность – отношение уровня услуг, предоставляемых программных средством пользователю к объему используемых ресурсов.

5) Сопровождаемость – характеристики, позволяющие минимизировать усилия по внесению изменений, для устранения в нем ошибок и по его модификациям.

6) Мобильность – способность программного средства быть перенесенным из одной среды в другую.

Обеспечение надежности как основной мотив разработки программных средств

Принципы обеспечении надежности. Четыре подхода

1) Предупреждение ошибок. Внимание следующим вопросам – борьба со сложностью; обеспечение точности перевода; преодоление барьера между пользователем и разработчиком; обеспечение контроля принимаемых решений

2) Само обнаружение ошибок – автоматическое обнаружение ошибок

3) Само исправление ошибок – автоматическое исправление ошибок

4) Обеспечение устойчивости к ошибкам – локализация масштаба повреждений

05.03.2016

Методы борьбы со сложностью

1) Обеспечение независимости компонент системы – разбиение систем на такие части, между которыми должны остаться по возможности меньше связи. Модульное программирование

2) Использование в системах иерархических структур – позволяет локализовать связи меду компонентами, допуская их лишь между компонентами принадлежащих смежным уровням иерархии

Обеспечение точности перевода

Четыре задачи

1) Поймите задачу

2) Составьте план

3) Выполните план

4) Проанализируйте полученное решение

Преодоление между пользователем и разработчиком

Включение пользователя в процесс разработки.

Контроль принимаемых решений – в процессе разработки необходимо постоянно принимать решения, и нужно понимать куда мы идем.

Два вида контроля

1) Смежный – проверка полученного документа. лицами не участвующими в его разработке.

2) Сочетание динамических и статических методов контроля – проверка не только документов как таковых, но и процессы обработки данных которые он описывает. Документ – статическая форма, а процесс обработки динамическая форма.

Лекций 4

Внешнее описание программного средства

Назначение внешнего описания программного средства и его роль в обеспечении качества программного средства

Внешнее описание (спецификация требований) – разработка программного средства начинается с этапа формирования требований, на основе которого должен быть получен документ, достаточно точно определяющий задачи разработчиков.

Исходным документом для разработки внешнего описания является определение требований.

Процесс прототипизирования на ранней стадии – пробник

Внешнее описание состоит из двух самостоятельных частей

1) Описание поведения программного средства, которая определяет функции и называется функциональной спецификацией

2) Спецификация качества программного средства

Внешнее описание определяет что должно делать программное средство и какими свойствами должно обладать.

Определение требований программного средства

- задание выражающее в абстрактной форме потребности пользователей. Они в общих чертах определяют замысел программного средства и характеризуют условия его использования. Определение требований – смесь фрагментов на естественном языке, различных таблиц и диаграмм.

Три способа определения требований программных средств

1) Управляемые пользователем – требования определяются заказчиком самостоятельно

2) Контролируемые пользователем – требования формулируются разработчиком при участии представителя пользователей

3) Независимые от пользователя – требования формулируются без участия пользователей

Спецификация качества программного средства

для конкретизации качества программного средства используется стандартизованный набор простых свойств, такие свойства будем называть примитивами качества программных средств.

Критерии:

1) Функциональность. Примитив – завершенность

2) Надежность. Примитивы – завершённость, точность, автономность, устойчивость и защищенность

3) Легкость применения. Примитивы – п-документированность, информативность, коммуникабельность, устойчивость и заищенность

4) Эффективность. Примитивы – временная эффективность, эффективность по памяти и эффективность по устройству

5) Сопровождаемость. Подкритерии. изучаемость (Примитивы с-документированность, информативность, понятность, структурированность, удобность) и модифицируемость (Примитивы – расширяемость, структурируемость и модульность).

6) мобильность. примитивы – независимость, автономность, структурированность и модульность

Определения примитивов

Завершенность – свойство характеризующее степень обладания программными средствами всеми необходимыми частями и чертами.

Точность – мера характеризующая приемлемость величины погрешности результата

Автономность – способность выполнять функции без помощи других компонентов.

Устойчивость – способность программного средства продолжать корректное функционирование, не смотря на задание неправильных входных данных

Защищенность – свойство характеризующее способность противостоять преднамеренным или случайным деструктивном действия

П-документированность – наличие, полнота, понятность, доступность, наглядность документации необходимой для программного средства

Информативность – свойство характеризующее наличие в составе программного средства информации необходимой и достаточной для понимания назначения программного средства, принятых предположений, существующих ограничений входных данных и выходных, а также текущее состояние программ в процессе их функционирования.

Коммуникабельность – свойство характеризующее степень в которой программные средства облегчают задание иди описание входных данных

Временные эффективные – мера, характеризующая способность выполнять определенные функции за заданный промежуток времени

Эффективность по памяти – мера выполнять возложенные функции при определенных ограничениях на используемую память.

Эффективность по устройствам – мера характеризующая экономичность использования устройств машины для решения поставленной задачи

С-документированность – свойство характеризующее с точки зрения наличия документации, отражающей требования к программному средсву и результаты различных этапов разработки

Понятность – степень в которой программные средства позволяют изучающему его лицу понять его назначение, допущения, ограничения, входные и выходные данные

Структурированность – точка зрения организации взаимосвязанных частей в единое целое определенным образом

Удобность (читаемость) – легкость восприятия текста программных средств

Расширяемость – способность программного средства к использованию большого объема памяти для хранения данных или расширение возможностей отдельных компонентов

Модульность – использование программного средства точки зрения организации программ при условии что изменение одной из них оказывает минимальное воздействие на функциональность других.

Независимость от устройств – способность работать на разном аппаратном обеспечении.

12.03.2016

Функциональная спецификация программного средства

Состоит из трёх частей

1) описание внешней информационной среды, в которой должны применятся программы программного средства

Определяются на концептуальном уровне все используемые каналы ввода и вывода и все информационные объекты к которым будут применятся разрабатываемые программные средства, а так же существующие связи между этими информационными объектами

2) определение функций программного средства определенных на множестве состояний этой информационной среды

Вводятся обозначения всех функций, специфицируются все входные данные и выходные данные определенных функций, включая указания этих типов и задания соотношений или ограничений, которым должны удовлетворять эти данные

3) описание нежелательной ситуаций, которые могут возникнуть в результате выполнения программ и реакции на эти ситуации

Должны быть перечислены все существенные с точки зрения внешнего наблюдателя случаи когда программное средство не может выполнить ту или иную функцию

Методы контроля внешнего описания программного средства

1) Статический просмотр – внимательное прочтение текстов внешнего описания программного средства разработчиком (внешнего описания) с

целью проверки его полноты и непротиворечивости и выявление неточностей и ошибок.

2) Смежный контроль спецификации качества

Сверху – это ее проверка со стороны разработчика требований программного средства.

Снизу – это его изучение и проверка разработчиками архитектуры программного средства и текстов программ, а так же проверка разработчиками документации по применению

3) Пользовательский контроль – выражает участие пользователя в принятии решения при разработке внешнего описания и его контроля

4) Ручная имитация – динамический контроль функциональной спецификации

Разработка структуры программы и модульное программирование

Цель модульного программирования

Программный модуль – любой фрагмент описания процесса, оформляемый как самостоятельный программный продукт, пригодный для использования в описаниях процесса. Каждый программный модуль разрабатывается отдельно от всех, и только потом подключается, тем самым строя систему. Рассматривается как средство борьбы со сложностью и как средство борьбы с дублированием. Модульное программирование включает оба метода борьбы со сложностью: обеспечение независимости компонент и использование иерархических систем.

Основные характеристики программного модуля

Хороший модуль снаружи проще чем внутри

Хороший модуль проще использовать, чем построить

Формальные (конструктивные характеристики)

1) Размер модуля – измеряется количеством содержащимся в нем строк, модуль не должен быть слишком маленьким или слишком большим. От 10 до 100 строк

2) Прочность модуля – мера его внутренних связей, чем выше прочность, тем больше связей он может спрятать от внешней по отношению к нему части программы. Для оценки степени прочности предлагается использовать 7 классов модулей

1. Самой слабой степенью прочности обладает модуль прочный по совпадению. Используется при повторении
2. Функционально прочный модуль – модуль выполняющий одну какую либо определенную функцию. При реализации этой функции модуль может использовать другие модули
3. Информационно прочный модуль – модуль выполняющий несколько операций на одной и той же структурой данных, которая считается неизвестной вне этого модуля
4. и другие ☺

3) Сцепление с другими модулями – мера его зависимости по данным от других модулей. Чем слабее – тем сильнее его независимость от других модулей (тем лучше). Худшим видом является сцепление по содержимому – в одном модуле переменная а в другом модуле эта переменная используется. изменение переменой приведет к неисправности второго модуля. Сцепление

по общей области – сцепление модулей, когда несколько модулей используют одну область памяти

4) Рутинность модуля – его независимость от истории обращений к нему. Модуль называется рутинным, если результат обращения к нему зависит только от значений параметров. Но модуль называется зависимым от предыстории, если результат обращения к нему зависит от внутреннего состояния модуля.

Рекомендации

1) всегда стоит использовать рутинные модули, если это не приводит к плохим сцеплениям модулей.

2) Зависящие от предыстории модули следует использовать, когда необходимо обеспечение параметрического сцепления

Методы разработки структуры программ

Модульная структура программы должна включать совокупность спецификаций модулей, образующих эту программу

Спецификация программного модуля содержит синтаксическую спецификацию его входов и функциональную спецификацию

26.03.2016

Метод восходящей разработки

Строится модульная структура программы в виде дерева. Основная концепция: поочередно программируются модули программы начинается с самого нижнего уровня (листьев) в таком порядке, чтоб для каждого

программируемого модуля были запрограммированы все модули к которым он обращается.

Метод нисходящей разработки

Сначала строится модульная структура программы, затем начинается программирование модулей с головного модуля (верхнего уровня), переходя к программированию какого либо другого модуля только в том случае, если запрограммирован тот модуль, к которому он обращается. Ставятся заглушки (Mock или Stub).

Конструктивный подход к разработке

Это модификация нисходящей разработки, где модульная древовидная структура программы формируется в процессе программирования модуля (главного сначала).

Архитектурный подход

Представляет собой модификация восходящей разработки. Модульная структура программы формируется в процессе формирования модуля. Повышение уровня использования языка программирования. Означает что для заданной предметной области выделяются типичные функции (нижнее уровни), каждая из которых может использоваться для решения разных задач, и, используя эти модули мы формируем наше программное средства.

Контроль структуры программы

Три контроля

1) Статический контроль. Оценка структуры программы с точки зрения хорошо ли программа разбита на модули.

2) Смежный контроль. Смежный контроль сверху – контроль со стороны разработчиков архитектуры и внешнего описания программного средства. Смежный контроль снизу – контроль спецификации модулей со стороны разработчиков этих модулей.

3) Сквозной контроль – мысленное прокручивание структуры программы при выполнении заранее разработанных наборов тестов. Вид динамического контроля.

Разработка программного модуля

Порядок разработки программного модуля

1) Изучение и проверка спецификации модуля, выбор языка программирования.

2) Выбор алгоритма и структуры данных. Реализованы ли алгоритмы (тогда лучше использовать их)

3) Программирование модуля.

4) Шлифовка текста модуля. Сделать читабельным для других пользователей.

5) Проверка модуля. Ручная проверка внутренней логики модуля до начала его отладки.

6) Компиляция модуля

Структурное программирование

1) Конструкции структурного программирования – следование, разветвление и повторение.

Пошаговая детализация и понятие о псевдокоде

Метод пошагово детализации модуля – процессы разработки текстового модуля разбивается на ряд шагов. На первом шаге описывается общая работа модуля в линейной текстовой форме (ориентирована на восприятие человека). На каждом следующем этапе производится уточнение и детализация всех необходимых понятий.

Контроль программного модуля

Применяется три метода

1) Статическая проверка текста модуля

2) Сквозное прослеживание

3) Доказательство свойств программного модуля

Проектирование программного обеспечения при объектом подходе

Инкапсуляция – сокрытие реализации от внешнего мира. Доступ к полям через методы класса. А в методах описана вся логика ограничения этих полей.

Наследование – заимствование у других классов полей и методов.

Полиморфизм – реализация одного метода для разных классов

Перегрузка – переопределение метода

Основной задачей логического проектирования при объектом подходе является разработка классов для реализации объектов полученных при объектной декомпозиции. что предполагает полное описание полей и методов каждого класса.

Физическое проектирование при объектном подходе включает объединение классов и других программных ресурсов в программные компоненты, а так же размещение этих компонентов для конкретных вычислительных устройствах.

Разработка структуры программного обеспечения при объектном подходе

Большинство классов можно отнести к определенному типу, называемому стереотипом.

- Классы сущности – классы предметной области

- Граничные классы (интерфейсы) – наследники реализуют методы. Используются для представления сущностей реального мира или внутренних элементов систем. Для выявления класса сущностей используют описание вариантов использования, концептуальную модель и диаграммы деятельности. Далее полученный список классов фильтруют.

- Управляющие классы – обеспечивают взаимодействия между действующими лицами и внутренними элементами системы. Если количество классов кандидатов и других ресурсов велико, то их целесообразно определить в группы (пакеты). Пакеты при объектном подходе называют совокупность описания классов и других программных ресурсов в том числе и самих пакетов.

Диаграмма пакетов показывает из каких частей состоит программная система и как эти части связаны друг с другом. Виды взаимосвязи

1) Объекты одного класса посылают сообщения объектам другого класса

2) Объекты одного класса обращаются к компонентам объектов другого класса

3) Объекты одного класса используют другие списки параметров других объектов

Диаграмма последовательности этапа проектирования – отображает взаимодействие объектов упорядоченное по времени. Объекты изображаются в виде прямоугольников в которых может быть имя объекта. каждое сообщение представляется виде линии со стрелкой. 09.04.16. Сообщению присваивают имя, но можно указать любую дополнительную информацию (параметры, внешние факторы).

Синхронные процессы – последовательные.

Асинхронный – параллельный

Диаграмма последовательности позволяет отображать также параллельные процессы, то есть асинхронные сообщения, которые не блокируют работу вызывающего объекта.

Диаграмма кооперaций – альтернативный способ представления взаимодействия объектов в процессе реализации сценария. В отличие от диаграмм последовательностей, диаграммы кооперaций показывают потоки данных между объектами, то позволяет уточнить связи между ними.

Уточнение отношений класса

На этапе проектирования помимо ассоциации и обобщения существуют два типа отношений: агрегация и композиция.

Агрегация – ассоциация между целым и его частью (частями). Пример колесо часть автомобиля, а если как товар то ассоциация.

Композиция – более сильная разновидность агрегации, где подразумевается что объект часть может принадлежать одному целому. Уничтожение целого уничтожает часть.

Интерфейсы – класс, содержащий только объявление операций (в UML).

Признаки видимости полей и методов (модификатор доступа)

18.04.2016

Спецификатор

<признак> <видимости> <имя аргументы> : <тип возвращаемого>

Ответственность класса – краткое неформальное перечисление основных функций объектов класса.

Диаграмма состояния объекта

Состояние объекта – ситуация в жизненном цикле объекта, во время которой он удовлетворяет некоторому условию, осуществляет определенную деятельность или ожидает некоторого события. Изменение состояния связанное с нарушением условия или с завершением деятельности или наступления определенного события называется переходом.

Диаграмма состояний показывает состояние объекта, возможные переходы, а так же события или сообщения вызывающие каждый переход.

Условие записывается в виде логического выражения, переход если результат выражения истина. Объект одновременно не может перейти одновременно в два состояния. Условия перехода ля любого события должны быть взаимно исключающими. Если события не указаны, то переход осуществляется по завершению деятельности связанной с данным состоянием.

Тестирование программных продуктов

QA – тестер.

Процесс разработки программного средства включает три стадии тестирования

1) Автономное

2) Комплексное

3) Системное

2 похода к формированию тестов

структурный

функциональный

Для повышения качества соблюдают следующие принципы

1) Предполагаемы результаты должны быть известны до тестирования

2)Избегать тестирование автором программы

3) Необходимо досконально изучить результаты каждого теста

4) Проверка действия программы на неверные данные

5) Проверка программы на неожиданные побочные эффекты на неверных данных

Формирование тестовых наборов

Удачным следует считать тест, который нашел хотя бы одну ошибку.

16,05,2016

Гибкие методологии разработки

Agile методология