Технология разработки программного обеспечения

Востриков Александр Владимирович

avostrikov@hse.ru

к.т.н., доцент департамента компьютерной инженерии

ауд. 904

Роли в команде

- Генеральный директор
- Ведущий программист
- Программист
- Тестировщик
- Технический писатель

Формирование оценки за дисциплину

Итоговая оценка за дисциплину *К* в модуле по 10-балльной шкале формируется как взвешенная сумма:

$К = 0{,}7 \cdot Тек + 0{,}3 \cdot Экз$,

при этом $0{,}7 \cdot Тек$ включает в себя:

- 0,2 ·Контрольная работа;
- 0,5 · Соблюдение сроков / активность на практических занятиях.

Экзамен проводится в виде защиты проектов / теоретических вопросов по дисциплине.

Контроль за ходом работы

Asana — мобильное и веб-приложение для управления проектами в небольших командах.

Регистрация в системе тут: asana.com.

Ликбез выложен https://www.youtube.com/watch?v=Jx8xhbFrOqI

Функциональные возможности ПО:

- гибкая система доступа, основанная на ролях;
- система отслеживания ошибок;
- календарь;
- учёт временных затрат;
- настраиваемые произвольные поля для временных затрат, проектов и пользователей;
- создание записей об ошибках на основе полученных писем;
- возможность самостоятельной регистрации новых пользователей.

Штрафы

- Опоздание сдачи этапа работы на 1 неделю:

Вычет 50% от набранного балла.

- Опоздание сдачи этапа работы на 2 недели:

Вычет 75% от набранного балла.

- Опоздание сдачи этапа работы на 3 недели и более:

Вычет 100% от набранного балла.

Контрольные точки работы

- 1 модуль. Контрольная работа. Презентация должна содержать в себе постановку задачи (1 балл), обоснование актуальности проекта (1 балла), разработанное техническое задание (1 балл), бизнес-план проекта (2 балла), оценку текущего состояния проекта (1 балл). Дополнительные баллы проставляются за ответы на вопросы преподавателя (2 балла) и студентов (1 балл), качество выполнения презентации (1 балла).
- 2 модуль. Контрольная работа. Работающее ПО, презентация должна содержать в себе демонстрацию разработанного ПО (2 балла), расчет финансовых показателей проекта (1 балл), соответствие выполненных работ плану выполнения проекта (2 балла), оценку текущего состояния проекта и перспектив его развития (1 балл). Дополнительные баллы проставляются за ответы на вопросы преподавателя (2 балла) и студентов (1 балл), качество выполнения презентации (1 балла).

Рекомендуемая литература

- Орлов С.А. Технологии разработки программного обеспечения: Разработка сложных программных систем: Учебное пособие. – 3-е изд. – СПб.: Питер, 2004. – 526 с.
- Брукс Ф. Мифический человеко-месяц / Символ, С-Пб.: 2000.
- Липаев В.В. Системное проектирование сложных программных средств для информационных систем / Синтег, М.: 1999.
- Рейнвотер Дж. Как пасти котов. Наставление для программистов, руководящих другими программистами / СПб.: Питер. 2006. С. 256.
- Йордон Э. Путь камикадзе / Лори, М.: 2003.
- Глаголев В. Разработка технической документации. СПб.: Питер, 2008. – 192 с.
- ГОСТ 34.601-90
- ГОСТ Р ИСО/МЭК 12207 (ISO/IEC 12207).
- Благодатских В.А., Волнин В.А., Поскалоф К.Ф. Стандартизация разработки программных средств. М.: Финансы и статистика, 2007. – 288 с.

Разделы дисциплины

- Введение
- Жизненный цикл ПО
- Начальная стадия ЖЦ (Анализ и планирование)
- Инженерия требований
- Управление программными проектами
- Управление ресурсами
- Управление проектами
- Инструментальная поддержка процесса разработки
- Обеспечение качества ПО
- Оценка качества ПО
- Методы обеспечения качества ПО
- Документирование ПО
- Заключение
- Качество процесса разработки
- Комплексные средства управления

Жизненный цикл ПО

- Фазы жизненного цикла ПО
- Стратегии конструирования ПО

- Однократные (водопадные) стратегии

1) Классическая каскадная модель

- Инкрементные стратегии

1) Инкрементная модель
2) RAD

- Эволюционные стратегии

1) Прототипироание
2) Спиральная модель
3) Экстремальное программирование
4) Модель SCRUM

- Смешанные подходы

1) Rational Unified Process (RUP)

Начальная фаза ЖЦ (анализ и планирование)

- Инженерия требований
- Сбор требований
- Анализ Требований
- Документирование требований
- Планирование и управление требованиями

Управление программными проектами

- Процесс проектирования программного продукта
- Управление ресурсами

1) Роли в программном проекте
   - Управление задачами
   - Этапы программного проекта
   - Наблюдение за проектом
   - Системы управления проектами и ресурсами

Инструментальная поддержка процесса разработки

- Версионирование проекта
  - Ветки, теги
  - Основные операции
  - Системы контроля версий
  - Поддержка нескольких версий ПО
- Управление дефектами и изменениями
  - Свойства дефекта
  - ЖЦ дефекта
  - Промышленные системы управления дефектами
- Сборка программных проектов
  - Основные задачи и проблемы
  - Управление зависимостями
  - Автоматизация сборки программных проектов
- Выпуск программного продукта
  - Дистрибутив
  - Альфа и бета-версии. Релиз программного продукта
  - Сопровождение программног продукта
- Управление рисками

Качество ПО

- Характеристики качества ПО
- Стандарты качества ПО
- Оценка качества ПО
- Метрики ПО
- Аудит ПО
- Повышение качества программных систем
- Рефакторинг программных систем
- Реинжиниринг ПО
- Формальная верификация ПО
- Статический анализ
- Тестирование ПО

Тестирование ПО

- Основные принципы тестирования ПО
  - Структурное тестирование
  - Функциональное тестирование
- Организация процесса тестирования
  - Модульное тестирование
  - Системное тестирование
    1) Тестирование восстановления
    2) Тестирование безопасности
    3) Стресс-тестирование
    4) Тестирование производительности
  - Регрессионное тестирование
- Тестирование приложений GUI
- Автоматизация тестирования ПО

Документирование ПО

- Виды программных документов
- Стандарты документирования
- UML как средство документирования
- Автоматизация документирования
- Промышленные системы документирования (DocBook, DITA)
- Документирование больших программных проектов

Лицензирование ПО

- Классификация ПО
- Виды лицензий ПО
- Свободные лицензии ПО

Заключение

- Качество процесса разработки
- Комплексные средства управления разработкой ПО

Программное обеспечение компьютерных систем

ПО и его классификации

ПО – совокупность программ, выполняемых вычислительной системой. К ПО относится область деятельности по его проектированию и разработке:

- технология проектирования программ;
- методы тестирования программ;
- анализ качества работы программ;
- документирование программ.

Сфера применения ПО

- ПО современных компьютеров включает миллионы программ – от игровых до научных. ПО по назначению делится на:

- **Базовое** (системное) ПО;
- **Рабочее** (прикладное) ПО;
- **Инструментальное** ПО (средства разработки ПО – СУБД реляционные (Oracle, MySQL), объектно-ориентированные, иерархические, сетевые).

Классификация ПО по способу распространения

- **Коммерческое** ПО;
- **Бесплатные** программы;
- **Условно-бесплатные** (их можно получить и опробовать бесплатно, но для систематического пользования нужно платить);
- **Пиратские** (ворованные) копии программ (не имеют документации).

Пакеты прикладных программ

- **ППП** – комплект программ, предназначенных для решения задач в определенной области
- Выделяет следующие виды ППП:

- **проблемно-ориентированные** (где возможна типизация функций управления) – ППП автоматизации бухучета, управления персоналом;
- **Автоматизации проектирования** (в работе конструкторов – разработка чертежей);
- **Общего назначения** (графические редакторы, СУБД);
- **Офисные**;
- **Системы искусственного интеллекта** (экспертные системы, поддержка общения на естественном языке).

Разработка ПС

Стадии разработки ПО, регламентированные ГОСТ

В РФ ЖЦ разработки ПО установлен стандартом ГОСТ 19.106-78 «Общие требования к программным документам, выполненным печатным способом» (09.1981) и содержит следующие стадии и этапы:

1. **ТЗ**
2. **Эскизный проект (ЭП)**
3. **Технический проект (ТП)**
4. **Рабочий проект (РП)**
5. **Внедрение**

Техническое задание

На стадии ТЗ выполняются следующие работы, входящие в состав соответствующих этапов.

1. **Обоснование необходимости разработки программ**: постановка задачи, сбор исходных материалов, выбор и обоснование критериев эффективности и качества.
2. **Выполнение НИР**: определение структуры входных и выходных данных, предварительный выбор методов решения задач, обоснование целесообразности применения ранее разработанных программ, определение требований к техническим средствам, обоснование возможности решения поставленных задач.
3. **Разработка и утверждение ТЗ**: определение требований к ПО, разработка технико-экономических показателей, определение стадий, этапов и сроков разработки ПО и документации на него, выбор языков программирования, согласование и утверждение ТЗ.

Эскизный проект

Результатом выполнения данной стадии является полное описание архитектуры ПО. Как правило, это описание делается на нескольких уровнях иерархии. На верхнем уровне детализации выделяются основные подсистемы, устанавливаются связи между основными подсистемами, прописываются функции подсистем. Затем процедура декомпозиции выполняется для каждой подсистемы, выделяются модули, составляющие эту подсистему. В итоге получается иерархически организованная система, состоящая из уровней (связь модулей). Единицы, выделяемые на различных уровнях, определяются разработчиком. Результаты ЭП отображаются в документе Пояснительная записка к ЭП, оформленному по ГОСТ 19.404-79.

Технический проект

Содержанием работ по этой стадии является проектирование структуры ПО. Результатом – реализующий заданный и утвержденный в ТЗ комплекс программ. Форма представления результата – пояснительная записка к техническому проекту согласно ГОСТ 19.105-78. Разработка структуры ПО заключается в выделении всех программных компонентов по функциональным признакам, определение функциональных спецификаций модулей, структуры входных и выходных данных, определение операционной среды, аппаратных средств.

Рабочий проект

Содержанием работ на этой стадии является описание ПО на выбранном проблемно-ориентированном языке (кодирование), разработка, отладка, согласование и утверждение порядка и методики испытаний, разработка программных документов, проведение тестирования, проведение приемо-сдаточных испытаний. Результат – ПО в форме программной документации или в форме программного изделия.

Качество ПО

Качество ПО – способность ПО подтвердить свою спецификацию при условии, что спецификация ориентирована на характеристики, которые желает получить пользователь.

Одной из важнейших проблем обеспечения качества программных средств является формализация характеристик качества и методология их оценки. Методологии и стандартизации оценки характеристик качества программных средств на различных этапах жизненного цикла посвящен международный стандарт ISO 14598.

Рекомендуется следующая общая схема процессов оценки характеристик качества программ:

Функциональная пригодность – наиболее неопределенная характеристика программного средства.

Оценка корректности программных средств состоит в формальном определении степени соответствия комплекса реализованных программ исходным требованиям контракта, ТЗ и спецификаций на программное средство. Путем верификации должно быть определено соответствие исходным требованиям всей совокупности компонентов комплекса программ, вплоть до модулей и текстов программ с описанием данных.

Оценка способности к взаимодействию состоит в определении качества совместной работы компонентов программных средств и БД с другими прикладными системами и компонентами на различных вычислительных платформах, а также взаимодействия с пользователями в стиле, удобном для перехода от одной вычислительной системы к

Оценка защищенности программных средств включает определение полноты использования доступных методов и средств защиты программного средства от потенциальных угроз и достигнутой при этом безопасности функционирования информационной системы. Наиболее широко и детально задачи оценки комплексной защиты информационных систем изложены в ISO 15408:1999-1-3 «Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий».

Оценка надежности – измерение количественных метрик к таким показателям как завершенность, устойчивость к дефектам, восстанавливаемость.

Потребность в ресурсах памяти и производительности компьютера в процессе решения задач значительно изменяется в зависимости от состава и объема исходных данных. Для корректного определения предельной пропускной способности информационной системы нужно измерить экстремальные и средние значения длительности исполнения программ.

Оценка практичности программных средств проводится экспертами и включает определение понятности, простоты использования, изучаемости и привлекательности программного средства.

Сопровождаемость можно оценивать полнотой и достоверностью документации на ПО. Она должна определять стратегию, стандарты, процедуры, распределение ресурсов и планы создания, изменения и применения документов на программы и данные.

Оценка мобильности – подготовленность ПС к переносу на другую платформу.

Надежность ПО

- **Надежность ПО** – способность ПП безотказно выполнять определенные функции при заданных условиях с большой вероятностью.
- **Степень надежности** характеризуется вероятностью работы ПП без отказа в течение определенного периода времени.

Источниками ненадежности являются непроверенные сочетания исходных данных, при которых отлаженные программы дают неверные результаты и отказы.

Отказ – нарушение работоспособности ПО, являющееся следствием таких явлений, как нарушение кодов записи программ в памяти, стирания или искажения данных в оперативной или долговременной памяти, нарушения нормального хода вычислительного процесса.

Сбой – самоустраняющийся отказ, не требующий внешнего вмешательства. Основное различие между сбоем и отказом – по временному показателю длительности восстановления. Если длительность восстановления больше какого-то порогового значения, то аномалию в работе ПО относят к отказам, иначе – к сбоям.

Жизненный цикл ПО

Жизненный цикл (ЖЦ) программной системы – это последовательность этапов, через которую проходит она в ходе своего существования. На данный момент существует несколько моделей жизненного цикла программных систем. В общем случае ЖЦ состоит из следующих этапов:

- анализ;
- проектирование;
- разработка;
- тестирование;
- развертывание;
- использование.

Варианты жизненного цикла программ

Варианты жизненного цикла программ

Также различают различные виды жизненных циклов и проектирования по виду сборки готового продукта. Это проектирование **нисходящее** (сверху вниз), **восходящее** (снизу вверх), **расширения ядра** (проектирование отдельных основных модулей, иерархическое проектирование дополнительных модулей, наращивающих функциональность) и **смешанное**.

При **проектировании сверху вниз** проводится общий анализ системы: определяются входные и выходные данные, требования к ним. Далее производится декомпозиция системы на отдельные подсистемы, определяются функциональность подсистем, связи между ними, наборы данных проходящих по этим связям, требования к данным. Далее процесс проектирования повторяется к отдельным подсистемам.

При **восходящем проектировании** на начальном этапе проектируются самостоятельные системы. После этого проектировщик пытается объединить их между собой таким образом, чтобы получить связанные подсистемы, обладающие большей функциональностью. Подобное объединение производится до тех пор, пока не будет получен модуль, обладающий требуемой функциональностью.

Метод расширения ядра заключается в том, что проектировщик выделяет наиболее важный, с его точки зрения, модуль и производит его проектирование. Далее проектируются модули, которые расширяют функциональность имеющегося в заданном направлении, производится их объединение. Таким образом, на каждом этапе проектирования в наличии имеется система с ограниченной функциональностью.

При **смешанных методиках** возможны комбинации указанных подходов. Так, например, в начале работ могут быть определены основные универсальные блоки, которые потребуются при создании системы. После их проектирования начинается проектирование сверху вниз, причем при проектировании функциональности блоков учитывается возможность включения в них уже разработанных универсальных.

Распределение работ

Существует эмпирический закон, который гласит, что в процессе создания ПО 30% времени тратится на анализ требований и проектирование, 40% времени – на кодирование и еще 30% - на доводку и тестирование. В конкретных случаях это отношение может меняться, однако для общего случая вполне верно. Можно даже сказать, что уровень программистов и разработчиков можно определить по тому, насколько соблюдается данное соотношение. При равномерно сбалансированной команде соотношение будет соблюдаться.

Анализ

Этап анализа требований посвящен работе с заказчиком. На данном этапе заказчик предъявляет требования к создаваемой системе по ее функциональности, качеству, времени и стоимости разработки. Требования к системе должны быть четко оговорены, задокументированы и подписаны обеими сторонам. Это делается для того, чтобы ни одна из сторон не могла в последствии отказаться от набора и характеристик требований: разработчик не может сдать не полный комплект работы, а заказчик не может претендовать на выполнение дополнительных работ или коррекцию их качества. Четкое формулирование требований избавляет как заказчика, так и разработчика от большого количества проблем, которые могут возникнуть впоследствии. Слишком общая постановка задачи приводит к множественности ее трактовки, слишком узкая постановка задачи лишает разработчиков маневра.

Двумя крайними случаями при анализе требований могут быть либо постановка задачи «Сделайте нам работающую систему», либо навязывание разработчикам приемов и методов, а также строгого собственного видения функциональности системы. В первом случае реакцией заказчика на готовый продукт может быть: «Это совсем не то, что мы ожидали. Мы думали, что вы профессионалы и можете разработать нормальную систему». Во втором случае может выясниться, что система не может быть разработана при существующих требованиях по соображениям технического (например, завышенные требования к производительности) либо экономического (например, быстрое создание сложной высококачественной хорошо задокументированной и передаваемой системы за минимальные деньги) плана.

Анализ. Разработка требований и внешнее проектирование ПО

1. Общая схема создания ПО

Процесс создания программ можно представить как последовательность действий:

1. Постановка задачи
2. Алгоритмизация решения задачи
3. Программирование

Постановка задачи

- это точная формулировка решения задачи на компьютере с описанием входной и выходной информации.

К основным характеристикам функциональных задач, уточняемым в процессе ее формализованной постановки, относятся:

- Цель и назначение задачи, ее место и связи с другими задачами;
- Условия решения задачи с использованием средств вычислительной техники;
- Содержание функций обработки входной информации при решении задачи;
- Требования к периодичности решения задачи;
- Ограничения по срокам и точности выходной информации;
- Состав и форма представления выходной информации;
- Источники входной информации для решения задачи;
- Пользователи задачи

Алгоритм

- Система точно сформулированных правил, определяющая процесс преобразования допустимых исходных данных в желаемый результат за конечное число шагов.

Обязательные свойства алгоритмов:

- Дискретность
- Определенность
- Выполнимость
- Массовость

В алгоритме обязательно должны быть предусмотрены все ситуации, которые могут возникнуть в процессе решения.

Программирование

Программа – реализованной алгоритм на языке программирования.

Наиболее часты программисты делятся на **системных и прикладных**.

В условиях создания больших по масштабам и функциям обработки программ существует квалификация – **программист-аналитик**, который анализирует и проектирует комплекс взаимосвязанных программ для реализации функций предметной области.

В процессе создания программ на начальной стадии работ участвуют и специалисты – **постановщики задач**.

Большинство информационных систем основано на работами с БД. Если БД является интегрированной, обеспечивающей работу с данными многих приложений, возникает проблема организационной поддержки БД, которая выполняется **администратором БД**.

Основным потребителем программ служит **конечный**

2. Разработка требований к ПО

Наиболее оптимальной является совместная работа проектировщиков и пользователей по выработке требований.

Можно установить 2 фазы по выработке требований:

1) **Фаза планирования**. Определяется реализуемость, устанавливаются цели, оцениваются затраты, обеспечивается ориентация для разработки проекта.
2) **Фаза выработки требований пользователя.** Вырабатываются требования к входным данным, информационным потокам, выходным данным, документации, среде, вычислительным ресурсам.

3. Цели разработки ПО

Цели разработки обычно включают следующую информацию:

- Краткое описание ПО
- Определение круга пользователей
- Подробное описание функциональных задач
- Документация. Определяются типы документации и предполагаемый круг читателей для каждого типа.
- Эффективность. Временные характеристики, использование ресурсов.
- Совместимость
- Конфигурация
- Безопасность
- Обслуживание
- Установка
- Надежность. Сбои и их последствия

4. Разработка внешних спецификаций проекта

Внешнее проектирование – это процесс описания планируемого поведения разрабатываемого ПО с точки зрения потенциальных пользователей (вопрос устройства интерфейса ПО).

Разработка внешних спецификаций разбивается на 2 части:

- Предварительный внешний проект
- Детальный внешний проект

Предварительный внешний проект содержит описание основных компонентов и внешних функций, составляющих отдельные компоненты проекта. Неопределенным остается точный синтаксис, семантика, выходные результаты.

Детальный внешний проект должен включать следующую информацию:

1. Описание входных данных (точное описание синтаксиса и семантики всех данных, вводимых пользователем).
2. Описание выходных данных (точное описание результатов функций).
3. Характеристики надежности (описание влияния всевозможных отказов).
4. Эффективность
5. Замечания по программированию

Проектирование и разработка

Пользовательский интерфейс является своеобразным коммуникационным каналом, по которому осуществляется взаимодействие пользователя и компьютера.

Лучший пользовательский интерфейс – это такой интерфейс, которому пользователь не должен уделять много внимания или почти не замечать его.

Общие принципы проектирования пользовательских интерфейсов

- Программа должна помогать выполнить задачу, а не становиться этой задачей.
- При работе с программой пользователь не должен ощущать себя дураком.
- Программа должна работать так, чтобы пользователь не считал компьютер дураком.

Графический интерфейс пользователя

Графический интерфейс пользователя (GUI) является обязательным компонентом большинства современных программных продуктов, ориентированных на работу конечного пользователя. К графическому интерфейсу предъявляются высокие требования как с чисто инженерной, так и с художественной стороны разработки, при его разработке ориентируются на возможности человека.

Наиболее GUI реализуется в интерактивном режиме работы пользователя для программных продуктов, функционирующих в среде Windows, и строится в виде системы спускающихся меню с использованием в качестве средства манипуляции мыши и клавиатуры. Работа пользователя осуществляется с экранными формами, содержащими объекты управления, панели инструментов с пиктограммами режимов и команд обработки.

Разработка

Собственно **разработка ПО** заключается в детальном проектировании отдельных работ и их реализации. Обычно считается, что данный этап является основным с точки зрения реализации проекта в целом. Однако скорее можно сказать, что этап важен с точки зрения качества и функциональности продукта.

С точки зрения менеджера проекта при разработке программного обеспечения следует помнить, что кризисную ситуацию чаще проще предотвратить, чем найти пути выхода из нее. Это означает, что в ходе разработки следует внимательно относиться к выполнению всех сроков, требований по качеству и функциональности продукта, объему его функций. Менеджер проекта интересуется деталями хода проекта до наступления контрольных точек, чтобы к их наступлению все работы были завершены.

Также менеджеру следует помнить, что увеличение длительности рабочей недели не всегда положительно сказывается на производительности. Йордон приводит интересную статистику. При увеличении длительности рабочей недели до 60 часов производительность продолжает расти, от 60 до 100 часов в неделю производительность стабилизируется и начинает падать, при рабочей неделе более 120 часов производительность становится отрицательной, то есть программисты больше исправляют собственные ошибки, чем пишут новый код.

С точки зрения разработчика следует помнить, что качество программного кода закладывается именно в момент его написания. Чем меньше ошибок допустит сам программист, чем больше времени он потратит на проверку кода и его проработку, тем меньше шансов найти в нем какой-либо изъян. Зачастую здесь может помочь «метод пристального взгляда», когда уже по окончании разработки кода программист повторно просматривает собственный код, причем возможно даже в распечатках.

Выражаясь метафорично, основным девизом этапа разработки является «Контроль над всем». Менеджер должен контролировать соответствие разработки календарному плану и разработанным требованиям. Программисты должны следить за качеством и безопасностью разрабатываемого ими кода. Тестировщики должны отслеживать соответствие продукта его функциональности и требованиям. Составители документации и инженерные психологи должны контролировать удобство использования системы.

Тестирование ПО

Роль этапа **тестирования** зачастую незаслуженно принижается. Однако ошибки в программном коде – явление не только обычное, но и системное. В конце 80-х – начале 90-х был проведен ряд исследований в области эффективности труда программистов. Так, было выявлено, что программист со стажем более 10 лет совершает в среднем 131,3 ошибки на 1000 строк кода. 50% из них выявляется на этапе компиляции. На этапе тестирования выявляется половина из оставшихся ошибок. Для устранения оставшихся ошибок требуется от 10 до 40 человеко-часов на заключительных этапах. При этом устранение ошибки в готовом продукте обходится иностранным компаниям примерно в 4000$. Коэффициент обнаружения ошибок пользователями в готовом продукте к количеству ошибок, обнаруженных при тестировании составляет в среднем 0,91. То есть, если на тестирование поступает некачественный продукт, есть много шансов за то, что он таким и останется.

Для менее опытных программистов принята следующая статистика. Один программист делает в ходе работы от 0,5 до 3 ошибок на строчку кода. В результате устранения указанных в ходе работы компилятора ошибок в коде остается порядка 1 ошибки на 10 строчек кода. Отладка программы сокращает количество ошибок до 1 на 100 строк кода. Тестирование кода специально выделенным сотрудниками с последующей коррекцией кода снижает число ошибок до 1 на 1000 строк. Эти цифры принято брать за основу при планировании разработки.

По данным Microsoft устранение одной ошибки, требующей создания пресс-релиза и пакета обновлений, обходится фирме примерно в 100 000$.

Виды тестирования

Существует несколько видов тестирования. Начальное тестирование проводится непосредственно разработчиками для того, чтобы убедиться, что ПО работает в соответствии с документацией. Функциональное тестирование проводится с целью более глубокой проверки соответствия функциональности ПО. Нагрузочное тестирование проверяет работоспособность ПО при повышенных нагрузках. Тестирование пользовательского интерфейса может быть разделено на две части: проверка функциональности интерфейса и проверка удобства интерфейса. Наиболее общим является регрессионное тестирование, состоящее из всех тестов, которые программа проходила до сих пор. Подобное тестирование необходимо так как изменение функционирования некоторого модуля может повлиять на работу других, связанных с ним, модулей, зачастую – весьма многочисленных.

Под регрессионным тестированием понимается последовательный прогон всех тестов, которые запускались для данного продукта до сих пор. Задачей регрессионного тестирования является проверка функциональности уже включенных модулей. Модули, которые не прошли тестирование, отправляются на доработку.

Кроме того, тестирование делят на тестирование методом черного, серого и белого ящиков. В первом случае тестировщик не имеет никакой информации о структуре тестируемой системы и описывает только симптомы неисправности, при этом он может высказывать предположения о причинах возникновения неисправности. В последнем случае тестировщик обладает полным исходным кодом программы и может указывать конкретные строки кода, вызвавшие сбой. Тестирование методом серого ящика является промежуточным вариантом – тестировщик имеет представление о составе модулей тестируемой системы, однако не имеет информации об их внутренней структуре или особенностях реализации.

Функциональное тестирование преследует своей целью проверку корректности работы приложения. В таком виде тестирования основными задачами является проверка устойчивости, корректности и, возможно, безопасности системы.

Нагрузочное тестирование служит для проверки функционирования системы в экстремальных условиях: нехватка оперативной или внешней памяти, потери канала связи или соединения с другим приложением, работа с поврежденными файлами и так далее. При этом программа должна проявлять устойчивость и безопасность.

Тестирование функциональности пользовательского интерфейса проводится с целью определения соответствия действий, привязанных к элементам управления, требованиям, заявленным в документации. То есть любая функция должна быть доступна именно тем методом, который был заявлен.

Тестирование удобства пользовательского интерфейса проводится скорее инженерными психологами, чем тестировщиками. Здесь психологи проверяют насколько приятно и удобно пользоваться данной программой, очевиден ли ее интерфейс, нельзя ли его улучшить.

При исправлении ошибки следует помнить несколько стандартных подходов.

- Повторный просмотр кода резко снижает количество ошибок, обнаруженных на этапе тестирования.
- Обычно программисты делают примерно одни и те же «любимые» ошибки. При обнаружении одной из них проверьте код на предмет обнаружения других аналогичных. Здесь, как и везде, действует принцип 80/20 – 80% ошибок встречаются в 20% кода.
- Копирование кода является методом повышения производительности и количества ошибок.
- Нет ничего постояннее, чем временное. Если вы не запишите себе в список задач доработку временного кода, есть большая вероятность получить от отдела тестирования запись в свой список ошибок.

Развертывание ПО

Развертывание приобретает высокую актуальность для больших систем. В простейшем случае программа сдается заказчику на некотором носителе и не требует специальных действий для работы с ней. Однако для сложных систем для запуска комплекса может потребоваться целый ряд работ. Некоторые информационные системы, например, могут потребовать разворачивания высокопроизводительного кластера. Сама инсталляция аппаратного и программного обеспечения для его работы может быть весьма трудоемкой задачей. Далее может потребоваться установка и настройка специального программного обеспечения: серверов приложений и БД, клиентских программ, заполнение баз начальными данными. При развертывании принципиально нового программного обеспечения может потребоваться обучение персонала навыкам работы с ним. В результате необходимо будет провести комплекс учебных мероприятий для начала функционирования системы.

Например, сложные системы АСУ ТП обычно доводятся непосредственно у заказчика, так как различные предприятия обладают собственной спецификой. При внедрении нового оборудования и программных систем перед началом работ производится обучение персонала. В конечном итоге должна быть произведена приемка продукта с подписанием соответствующих актов.

Развертывание сложной системы может оказаться трудоемким и затратным процессом, который необходимо предусмотреть еще на этапах анализа требований и проектирования. Так, например, может оказаться, что заказчиком не предусмотрены средства для развертывания системы и весь проект окончится провалом. Кроме того, может выясниться, что развертывание комплекса в существующих условиях просто невозможно. Также может выясниться, что развертывание комплекса может быть слишком затратным мероприятием, стоимость которого превышает саму стоимость разработки.

Сопровождение ПО

Сопровождение ПО также следует предусмотреть еще на этапе проектирования, так как его стоимость может существенно повлиять на оценку прибыльности проекта. В ходе использования системы осуществляется информационная поддержка пользователей, возможно их обучение, устранение обнаруженных неисправностей, изменение функциональности и состава системы в соответствии с вновь появившимися требованиями заказчика. Если оговаривается соответствующий сервис, создается служба технической поддержки, которая осуществляет обслуживание комплекса, отвечает на возникающие вопросы. При обнаружении неисправностей или несоответствия заявленной функциональности производится замена соответствующих модулей вновь разработанными. В случае, если заказчик решает изменить функциональность системы, производится ее модернизация.