

ТЕХНОЛОГИЯ И ПРОЦЕСС РАЗРАБОТКИ ПО (Л-7)



к.п.н., доцент Касаткин Д.
А.

e-mail: kasatkinda@cfuv.ru



Обеспечения качества ПО

- Актуальный вопрос современной индустрии ПО – **обеспечение качества**
- Тенденции в образовании: от теории и технологий программирования к программной инженерии
- Наша цель: рассмотреть теоретические и практические вопросы обеспечения качества ПО на различных этапах жизненного цикла
- Все представляемые методы являются формальными – могут быть представлены с помощью некоторого формализма





Качество ПО

Стандарт ISO 9126 учитывает точки зрения

- Разработчиков – *внутреннее качество ПО*
- Руководства и аттестации ПО – *внешнее качество ПО*
- Конечных пользователей – *качество ПО при использовании.*
- Качество ПО включает
- 6 факторов
- 27 атрибутов – для качественной оценки факторов метрики или показатели – для количественной оценки атрибутов
- ГОСТ Р ИСО/МЭК 9126



Функциональность

- **Функциональность** – способность ПО в определенных условиях решать задачи, нужные пользователям
- Функциональная пригодность – способность решать нужный набор задач
- Точность – способность выдавать нужные результаты
- Способность к взаимодействию, совместимость – способность взаимодействовать с нужным набором других систем
- Соответствие стандартам и правилам – соответствие ПО имеющимся стандартам, нормативным и законодательным актам, другим регулирующим нормам
- Защищенность – способность предотвращать неавторизованный и не разрешенный доступ к данным, коммуникациям и др



Надежность

- **Надежность** – способность ПО выполнять свои функции в заданных условиях
- **Зрелость** – величина, обратная частоте критических отказов, вызванных ошибками в ПО
- **Устойчивость к отказам** – способность поддерживать заданный уровень работоспособности при внутренних и внешних отказах
- **Способность к восстановлению** – способность восстанавливать определенный уровень работоспособности и целостность данных после отказа
- **Соответствие стандартам надежности**



Удобство сопровождения

- **Удобство сопровождения** – удобство проведения всех видов деятельности, связанных с сопровождение программ
- Удобство проведения анализа – удобство проведения анализа ошибок, дефектов и недостатков, а также удобство анализа необходимости изменений и их возможных последствий
- Удобство проверки – показатель, обратный трудозатратам на проведение тестирования и других видов проверки того, что внесенные изменения привели к нужным результатам
- Удобство внесения изменений – показатель, обратный трудозатратам на выполнение необходимых изменений
- Стабильность – показатель, обратный риску возникновения неожиданных эффектов при внесении необходимых изменений
- Соответствие стандартам удобства сопровождения



Эффективность

- **Эффективность** (производительность) – свойство ПО при заданных условиях обеспечивать необходимую работоспособность по отношению к выделяемым ресурсам
- **Временная эффективность** – способность ПО решать определенные задачи за отведенное время
- **Эффективность использования ресурсов** – способность решать нужные задачи с использованием заданных объемов ресурсов определенных видов (ресурсоемкость)
Соответствие стандартам производительности



Удобство использования

- **Удобство использования** – способность ПО быть удобным в обучении и использовании
- *Понятность* – показатель, обратный к усилиям, которые затрачиваются пользователями на восприятие основных понятий ПО и осознание способов их использования для решения своих задач
- *Удобство обучения* – показатель, обратный к усилиям, затрачиваемым пользователями на обучение работе с ПО *Удобство работы* – показатель, обратный трудоемкости решения пользователями задач с помощью ПО
- *Привлекательность* – способность ПО быть привлекательным для пользователей
- Соответствие стандартам удобства использования



Переносимость (мобильность)

- **Переносимость** (мобильность) – способность ПО сохранять работоспособность при переносе из одного окружения в другое
- (аппаратное, программное окружение)
- Адаптируемость – способность ПО приспособливаться к различным окружениям без специальных действий
- Удобство установки – способность ПО быть установленным или развернутым в определенном окружении
- Способность к сосуществованию – способность ПО сосуществовать в общем окружении с другими программами, разделяя с ними общие ресурсы
- Удобство замены другого ПО данным – возможность применения данного ПО вместо других программных систем для решения тех же задач в определенном окружении
- Соответствие стандартам переносимости



Оценочные характеристики качества:

- Для оценки различных свойств процесса создания программного продукта, а также и самого продукта, **применяются количественные характеристики, называемые мерами**
- Путем непосредственного измерения **определяются опорные свойства.**
- Остальные свойства оцениваются путем вычисления функций от опорных значений. Такие функции **называются метриками.**



Размерно-ориентированные метрики

- Размерно-ориентированные метрики Основаны на **LOC-оценках**, т.е. на **количестве строк в текстах программ (Lines Of Code (LOC))**. К числу размерно-ориентированных метрик относятся:
 - производительность
 - качество
 - удельная стоимость
 - документированность



метрики производительности и качества

- Метрики производительности и качества рассчитываются в виде следующих отношений:

Производительность = [число строк кода(LOC)] / [Затраты]

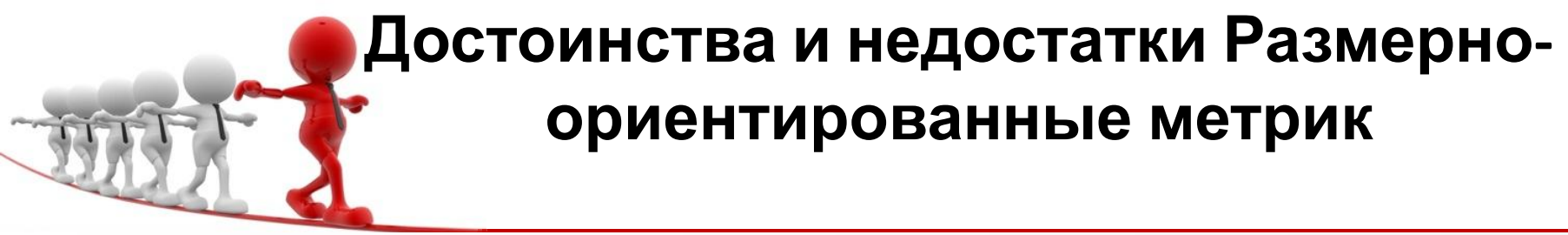
- где затраты измеряются в человеко-месяцах
(работа одного человека в течении месяца)

Качество = [число ошибок] / [число строк кода(LOC)]

Метрики стоимости и документированности

Удельная Стоимость = [Стоимость в тыс. рублей] / [число строк кода(LOC)]

Документированность = [число страниц документации] / [число строк кода(LOC)]



Достоинства и недостатки Размерно-ориентированные метрик

Достоинства:

- основаны на объективных данных
- просты и легко вычислимы

Недостатки:

- зависят от языка программирования
- трудновыполнимы на начальной стадии проекта
- не приспособлены к непроцедурным языкам программирования



Функционально-ориентированные метрики (FP-оценки)

Исходят не из размера программного продукта, а из его функциональности.

Оценивают:

- характер пользовательского интерфейса
- сложность выполняемой обработки
- распространенность используемой конфигурации
- степень сложности инсталляции
- условия эксплуатации
- степень модифицируемости



FP-оценки

Вместо количества строк в текстах используется количество функциональных указателей (Function Points)

следующая формула $FP = UI * (0.65 + 0.01 * E[F(i)])$

где =

- **UI** - оценка сложности пользовательского интерфейса,
- **F(i)** ("эф итое") – коэффициенты регулирования сложности, основанные на эмпирической оценке ряда системных параметров и принимающие целые значения в диапазоне от 0 до 5.
- **E[F(i)]** - сумма всех коэффициентов по i параметру.



FR-оценки

К числу параметров, учитываемых коэффициентами регулирования сложности относятся:

- объем используемых средств передачи данных
- степень распределенности обработки
- степень распространенности используемой аппаратной платформы
- степень жесткости требований к производительности
- частота выполнения транзакций



FR-оценки

Кроме того учитываются:

- процент информации, вводимой в режиме on-line
- сложность обработки данных, наличие значительной логической и математической обработки
- легкость инсталляции
- степень переносимости
- степень модифицируемости



Область применения метода функциональных указателей ((Function Points) – **коммерческие информационные системы**

- **Для продуктов с высокой алгоритмической сложностью** (*системного и встроеного ПО, ПО реального времени*) используется **метод указателей свойств (Features Points)**.
При расчете **указателя свойств** учитывается **количество используемых в ПО алгоритмов**

Функционально-ориентированные метрики

- Функционально-ориентированные метрики **аналогичны соответствующим размерно-ориентированным метрикам с точностью до замены** =
- параметра длины на количество функциональных указателей
- или указатель свойств в зависимости от выбранного метода FP-оценки



Достоинства и Недостатки Функционально-ориентированных метрик

Достоинства:

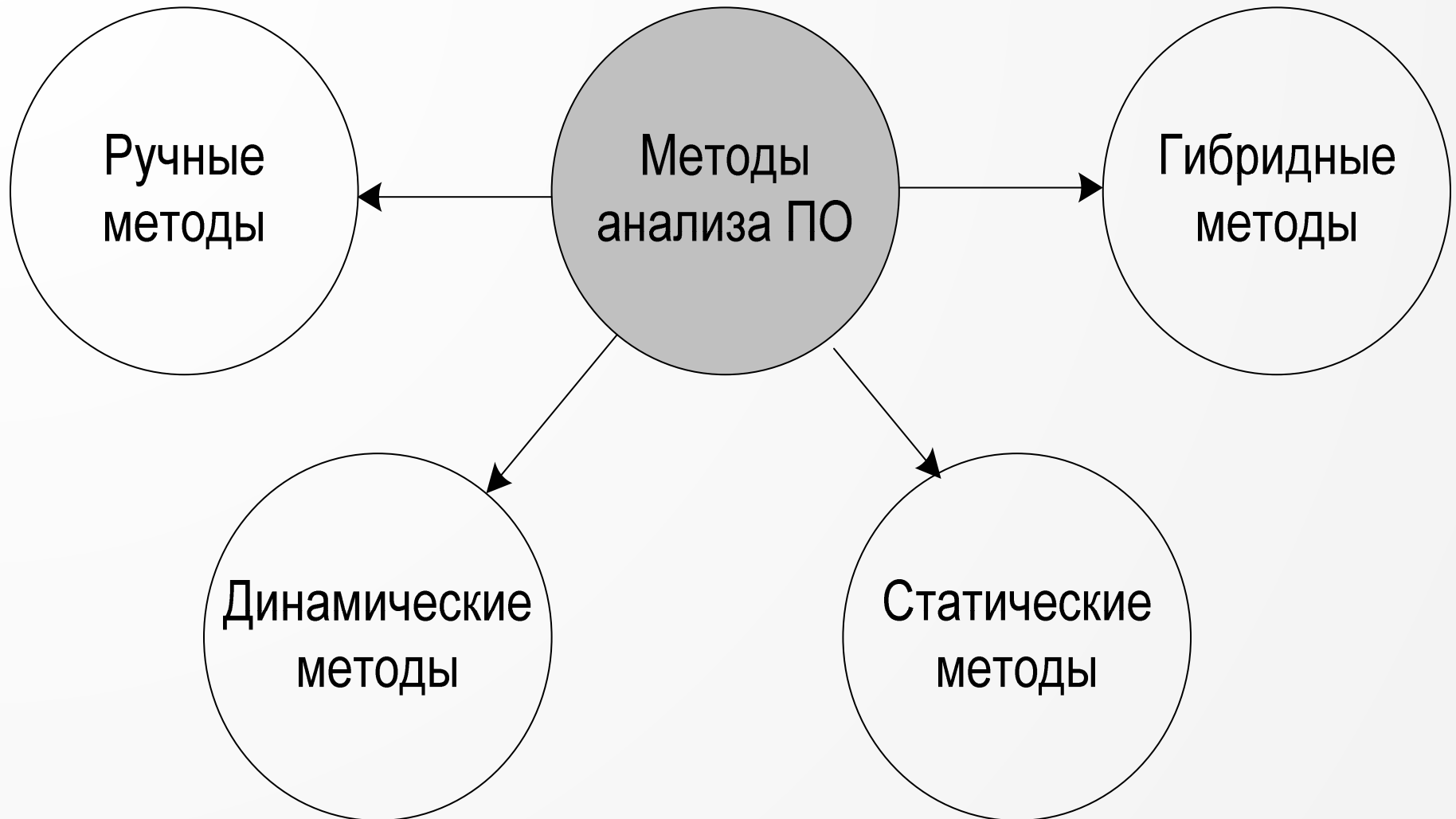
- не зависят от выбора языка программирования
- вычисляются на любой стадии проекта

Недостатки:

- используют не прямые, а косвенные измерения
- основаны на субъективных оценках



Методы анализа ПО



Ручные методы



- Персональные проверки
- Аудит кода
- Парное программирование
- Ручная верификация

НЕ НАШИ МЕТОДЫ!!!



Динамические методы

- **Динамические методы** используют результаты выполнения программы
Тестирование
- Модульное
- Системное
- Нагрузочное
- Мониторинг
- Профилирование
- Анализ трасс выполнения



Статические методы

- **Статические методы** используют различные артефакты получаемые в процессе проектирования ПО (требования, спецификации, исходные код программы)
- Методы формальной верификации
- Дедуктивная верификация
- Верификация на основе проверки моделей
- Статический анализ исходного кода



Гибридные методы

- **Гибридные методы** используют несколько разных методов
- Создание тестов на основе статического анализа
- Статический анализ для автоматического формирования моделей, для которых применяются формальные методы проверки моделей
- Уточнение результатов статического анализа с помощью методов проверки моделей
- Комбинирование результатов статического анализа и тестирования для повышения точности результатов



Методы повышения качества

- Методы повышения надежности
- Динамические, на основе тестирования, анализа трасс выполнения и др.
- Статические, на основе статического анализа и верификации
- Методы улучшения функциональности
- Динамические, на основе тестирования, анализа трасс выполнения и др.
- Статические, на основе методов формальной верификации



Методы оценки качества

- Методы оценки надежности
- Динамические, на основе прогнозных моделей
- Статические, на основе метрик сложности и обнаружения дефектов
- Архитектурные, на основе анализа архитектуры ПО и надежности отдельных компонентов
- Методы оценки функциональности
- Динамические, на основе тестирования программы
- Статические, на основе методов формальной верификации
- Методы оценки эффективности
- Динамические, на основе профилирования
- Статические, на основе анализа возможных путей выполнения



Надежность ПО

- Надежность ПО является одной из важнейшей характеристик качества
- **Надежность ПО** – вероятность его работы без отказов в течении периода времени, рассчитанная с учетом стоимости каждого отказа (Майерс)
- Надежность ПО должна учитывать не только частоту проявления ошибок, но и серьезность их последствий для пользователя системы.
- Оценивать и повышать надежность можно на любой стадии проектирования, на основе одного или нескольких представлений программы, при этом можно говорить только о надежности исполняемой программы



Требования к надежности ПО

Для каждой программы можно определить необходимый уровень надежности

Назначение	Область применения	Требования к надежности
Программа, демонстрирующая возможности транзакций	Курс лабораторных работ по предмету «Базы данных»	Время работы – единицы минут, корректно работает при заранее определенных действиях пользователя
Программа расчета заработной платы	Финансовый отдел предприятия	Время работы – десятки часов, устойчива к любым действиям пользователя, обеспечивает восстановление данных после сбоя.
Программа управления срабатыванием подушки безопасности	Автомобиль	Время работы – десятки тысяч часов, устойчива к любым внешним воздействиям, определяет случаи перехода в нерабочее состояние.
Программа управления опасным объектом	Промышленность, авиация и космос, ВС	???



Причины ненадежности

- Основными источниками ненадежности аппаратных систем являются внешние факторы, обычно неподвластные человеку:
- скачки напряжения питания; электромагнитное излучение; радиация;
- ...
- **Источником ненадежности программ являются ошибки, сделанные разработчиками программ, на разных стадиях проектирования**
- Будем считать программу правильной, если она не содержит ошибок разработчиков, такая программа не дает неверных результатов
- **Абсолютно надежна**



Источники ошибок в ПО

- Что такое ошибка в программе ?
- Если программа не соответствует
- **Спецификации** – в ней то же могут быть ошибки
- Неформальным требованиям пользователя – пользователь может не учесть всех возможных ситуаций или неправильно сформулировать свои требования, у программы может быть много пользователей с различными требованиями
- Непредусмотренные входные данные и воздействия
- Ошибки окружения программы – некорректная работа другого ПО и аппаратуры
- Является ли луна вражеским объектом ? Одна из первых компьютерных систем противовоздушной обороны США (60-е годы) в первое же дежурство подняла тревогу, приняв восходящую из-за горизонта Луну за вражескую ракету, поскольку этот «объект» приближался к территории США и не подавал сигналов что он «свой»



Определение надежной программы

- В программе имеется ошибка, если она не выполняет действия, которые ожидает от нее некий абстрактный пользователь(эксперт), в том числе и при недопустимых внешних воздействиях и входных данных, а также при отказах другого ПО и сбоях и отказах аппаратуры.
- Наличие ошибки – функция самой программы и нереализованных ожиданий ее пользователей (Майерс)
- Из этого определения следует:
- Программа не имеющая ошибок может давать неверные результаты, однако стремится минимизировать возможный ущерб
- Такой программы не существует



Ошибки в программах

- **Ошибки имеются практически во всех программах** Для программ на языке C в среднем
- **0,25 ошибок на 1 KLOC**
- Примерно 45% ошибок являются критическими
- В ядре ОС Android (765 KLOC) найдено 359 ошибок*

- * Coverity Scan: 2010 Open Source Integrity Report



Последствия ошибок в программах

- Переоблучение больных из-за ошибки в программе управления радиотерапевтической установкой
- Печально известная ошибка в линейном ускорителе Therac-25 стала причиной гибели нескольких больных, получивших смертельные дозы радиации во время лечения, проводимого с июня 1985-го по январь 1987 года в нескольких онкологических клиниках в США и Канаде. Эти дозы, как было оценено позже, более чем в 100 раз превышали те, что обычно применяются при лечении. Частично причиной этих несчастий стала ошибка типа race condition.



Последствия ошибок в программах



Авария при запуске французской ракеты «Ариан-5» (1996) на 37-й секунде полёта компьютер, находившийся на борту ракеты, получил от датчиков системы управления неверную информацию о пространственной ориентации ракеты. Исходя из этой информации, компьютер начал корректировать траекторию полёта для того, чтобы компенсировать несуществующую на самом деле погрешность. Ракета стала отклоняться от курса, что привело к возрастанию нагрузок на её корпус. В результате чрезмерных нагрузок верхняя часть ракеты отвалилась, и по команде земли



Последствия ошибок в программах

- Неудача при запуске первого американского спутника к Венере

Единственная ошибка в программе на Фортране – вместо требуемой в операторе запятой программист поставил точку. В результате

Потеря связи с космической станцией «Фобос-1»

Произошла из-за ошибочной команды, переданной с Земли на бортовой компьютер

Ошибка не учета отрицательной высоты

- При полетах над Мертвым морем американских самолетов произошла ошибка деления на ноль что привело к перезагрузке системы
- Падение спутников системы ГЛОНАСС
- Три спутника навигационной системы ГЛОНАСС упали в Тихий океан недалеко от Гавайских островов вскоре после их запуска. Причина аварии была признана ошибка в программировании, которая привела к тому, что в ракету залили неправильное количество топлива.



Фобос-Грунт



«... никаких фатальных ошибок и дефектов при создании станции обнаружено не было". Причиной возникновения "нештатной ситуации", установили специалисты, стал "перезапуск двух полуккомплектов устройства ЦВМ22 БВК, выполнявших на этом участке полета управление КА "Фобос-Грунт"...



Рекомендуемая литература

- Paul Ammann, Jeff Offutt. Introduction to Software Testing. -- Cambridge University Press, 2008
- Cem Kaner, Jack Falk, Hung Q. Nguyen. Testing Computer Software. -- Wiley, 1999
- Ю.Г. Карпов. Model Checking. Верификация параллельных и распределенных программных систем -- БХВ-Петербург, 2010
- Doron A. Peled. Software Reliability Methods. -- Springer, 2001
- Nielson F., Nielson H.R., Hankin C. Principles of Program Analysis. Springer, 2005
- M.R. Lyu Handbook of Software Reliability Engineering. McGraw-Hill publishing, 1995
- Г.Майерс. Надежность программного обеспечения, 1980
- В. Кулямин. Методы верификации программного обеспечения – http://www.sciinnov.ru/icatalog_new/entry_62322.htm