Качество ПО
Метрики ПО

Качество программного обеспечения

- **Качество программного обеспечения** — способность программного продукта при заданных условиях удовлетворять установленным или предполагаемым потребностям (ISO/IEC 25000:2014)[
- **В настоящее время не существует общепринятых критериев качества программного обеспечения**

- Одним из подходов для оценки программных средств является оценка соответствующих атрибутов качества, определённых в серии международных стандартов ГОСТ Р ИСО МЭК 9126 «Информационная технология – Оценка программной продукции».
- Стандарты определяют базовую терминологию и общий подход к проблеме оценки качества программных средств (характеристики качества, метрики для их измерения, методологию оценки), что позволяет уменьшить неопределённость при совместной работе нескольких организаций (заказчики разработки, разработчики, независимые оценщики).

Можно выделить три большие группы факторов, влияющих на качество программного обеспечения:

- ***функциональная*** — связана с полнотой и удобством использования реализованных функций программного средства;
- ***административная*** — связана с квалификацией персонала, организационной структурой и управлением персоналом;
- ***программно-архитектурная*** — связана с процессом разработки программного обеспечения, выбранными методологиями, инструментальными средствами, использованными на различных этапах жизненного цикла программного обеспечения, а также архитектурой программного средства.

Программное обеспечение как продукт имеет некоторые отличия от других промышленных продуктов:

- наращивание объемов выпуска какого-то вида программного продукта происходит практически мгновенно и имеет низкую стоимость, так как производство следующей единицы программного продукта связано только с копированием информации на носитель (компактдиск, дискету или жесткий диск);
- большие ресурсы затрачиваются на стадии планирования, реализации и тестирования;
- сильное влияние человеческого фактора на производство программного продукта, так как производство программного продукта — интеллектуальная и творческая деятельность;
- в жизненном цикле программного продукта, как правило, отсутствует этап утилизации;
- программный продукт не подвержен физическому старению, а только моральному.

Для измерения некоторых показателей качества могут служить:

- тестирование,
- тестирование пользователем (так называемое β-тестирование),
- информация от пользователя о найденных проблемах, получаемая от службы технической поддержки.

- Мероприятия, обеспечивающие приемлемый уровень качества программного средства, можно условно разделить на **административные** и **технологические**.

К **административным** можно отнести следующие мероприятия:

- Проведение **обучения** персонала, переподготовки.
- Тщательное **документирование** всех изменений в структуре ПС. Для этого используются средства поддержки версионности.
- Назначение **ответственных лиц** за каждую доработку программного средства.

- Обеспечение мониторинга качества, например, фиксирование ошибок, поступивших от пользователя ПС. Использование систематических испытательных методов, где испытания будут разработаны параллельно с разработкой программы.
- Введение внутренних стандартов. Такие стандарты обычно содержат соглашения о именовании переменных в программном коде, наименовании файлов данных, процедур и функций.

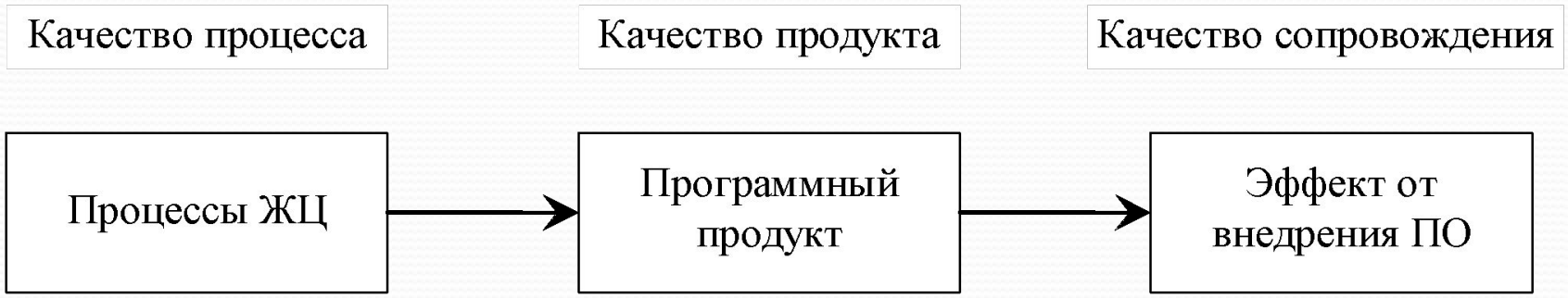
5. Организация отдела тестирования как самостоятельного подразделения.
6. Проведение совместных аттестаций с пользователем

К **технологическим** относятся следующие мероприятия.

1. Выбор стандарта качества и четкое следование ему на всех этапах. Создание модели проекта с регулярными проверками. которые будут выполняться независимыми командами экспертизы. Такая модель может быть построена, например, на основе стандартов качества (например, ISO 9000).
2. Единая среда разработки. Лучшие результаты дают программные продукты разработки, которые поддерживают несколько или все этапы жизненного цикла программного обеспечения.
   На данный момент такими комплексными решениями являются, например, продукты Oracle Designer, продукты фирмы Rational.
3. Использовать формальный язык спецификаций (например. UML, DESIGN IDEF).
4. Выбор надежной СУБД (если программное средство работает с массивами информации и использование СУБД оправдано).
5. Тщательное тестирование программного обеспечения.

6. Широкое внедрение автоматизации тестирования.
7. Использование полностью проверенной программной среды окружения (ОС) и языка программирования, которые минимизируют опасность внесения ошибки.
8. Использование статистических методов для сбора информации о качестве ПС.
9. Изучение результатов испытаний (тестов) и ошибок для использования в постоянном усовершенствовании программы. Источник в случае возникновения отказа должен быть найден и устранен. Недостаточно найти ошибку в программном обеспечении и исправить ее. Изменения должны быть сделаны в процессе разработки ПО.
10. Использование испытательной среды, которая предостережет от передачи пользователю ненадежного программного обеспечения. Создание автоматических средств приемки.

Аспекты качества ПО



Качество продукта достигается процедурами контроля промежуточных продуктов на процессах ЖЦ, проверкой их на достижение необходимого качества, а также методами сопровождения продукта. Поиск и устранение ошибок в готовом ПО проводится методами тестирования, которые снижают количество ошибок и повышают качество этого продукта. Эффект от внедрения ПС в значительной степени зависит от знаний обслуживающего персонала функций продукта и правил их выполнения.

Модель качества

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

- **Переносимость** (мобильность) - способность ПО сохранять работоспособность при переносе из одного окружения в другое (аппаратное, программное окружение)
  - Адаптируемость - способность ПО приспосабливаться 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

Причины недостаточного качества

Функциональность

- Функциональные ошибки - несоответствия требованиям пользователей, функциональной спецификации и т.п.

Надежность

- Нефункциональные ошибки - нарушение правил языка программирования, использования библиотечных функций и сторонних компонентов и т.п.

Эффективность

- Ошибки анализа необходимого количества ресурсов, обычно проявляются только в определенных ситуациях

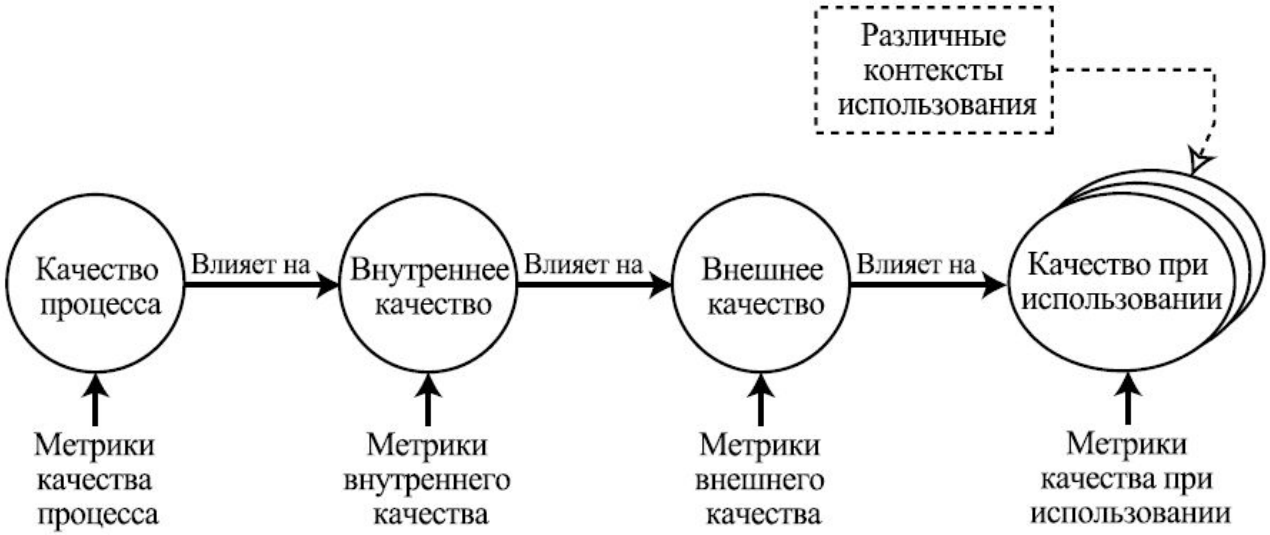
Задачи обеспечения качества

- Обеспечение качества
  - **Измерение (оценка) качества программы**
  - **Применение методов повышения качества**
- Повышение качества
  - Обнаружение ошибок и неудовлетворительных мест в программе
  - Исправление ошибок и другие изменения программы
- Необходимость оценки качества
  - Контроль текущего прогресса
  - Оценка эффективности затрат на повышение качества
  - Выбор наиболее эффективных методов повышения качества
- Основа для измерения и повышения качества - **анализ ПО**

Модель характеристик качества программного обеспечения состоит из нескольких видов атрибутов качества:

- внутренние атрибуты качества (требования к качеству кода и внутренней архитектуре);
- внешние атрибуты качества (требования к функциональным возможностям и т.д.);
- Кроме того, для создания добротного ПО существует качество технологических процессов его разработки.

Для всех этих аспектов качества введены метрики, позволяющие оценить их.



Основные положения по метрической оценке качества ПО

Руководствуются ***стандартами серии ISO/IEC 9126.***

Метрика качества программ - система измерений качества программ. Эти измерения могут проводиться на уровне критериев качества программ или на уровне отдельных характеристик качества. В первом случае система измерений позволяет непосредственно сравнивать программы по качеству. При этом сами измерения не могут быть проведены без субъективных оценок свойств программ. Во втором случае измерения характеристик можно выполнить объективно и достоверно, но оценка качества ПО в целом будет связана с субъективной интерпретацией получаемых оценок.

- Внешние и внутренние метрики задаются на этапе формирования требований к ПО и являются предметом планирования способов достижения качества конечного программного продукта.

Внешние метрики продукта

- надежности продукта, которые служат для определения числа дефектов;
- функциональности, с помощью которых устанавливаются наличие и правильность реализации функций в продукте;
- сопровождения, с помощью которых измеряются ресурсы продукта (скорость, память, среда);
- применимости продукта, которые способствуют определению степени доступности для изучения и использования;
- стоимости, которыми определяется стоимость созданного продукта.

Внутренние метрики продукта

- метрики размера, необходимые для измерения продукта с помощью его внутренних характеристик (количество операций);
- метрики сложности, необходимые для определения сложности продукта (количество циклов, глубина циклов);
- метрики стиля, которые служат для определения подходов и технологий создания отдельных компонентов продукта и его документов.

Метрические шкалы

Зависят от характеристик и особенностей применяемых метрик.

- Номинальной шкале соответствуют метрики, классифицирующие программы на типы по признаку наличия или отсутствия некоторой характеристики без учета градаций.
- Порядковой шкале соответствуют метрики, позволяющие ранжировать некоторое характеристики путем сравнения с опорными значениями, т.е. измерение по этой шкале фактически определяет взаимное положение конкретных программ.
- Интервальной шкале соответствуют метрики, которые показывают не только относительное положение программ, но и то, как далеко они отстоят друг от друга.
- Относительной шкале соответствуют метрики, позволяющие не только расположить программы определенным образом и оценить их положение относительно друг друга, но и определить, как далеко оценки отстоят от границы, начиная с которой характеристика может быть измерена.
- Абсолютная шкала указывает на фактическое значение величины (например, число ошибок в программе равно 10).

Стандартная оценка значений показателей качества

- Модель качества *ПО* имеет следующие четыре уровня представления.
- **Первый уровень** соответствует определению характеристик (показателей) качества *ПО*, каждая из которых отражает отдельную точку зрения пользователя на качество (эффективность)
- **Второму уровню** соответствуют атрибуты для каждой характеристики качества, которые детализируют разные аспекты конкретной характеристики (используемость ресурсов)
- **Третий уровень** предназначен для измерения качества с помощью метрик, каждая из них определяется как комбинация метода измерения атрибута и шкалы измерения значений атрибутов. *Атрибут* качества определяется с помощью одной или нескольких методик оценки на этапах ЖЦ и на завершающем этапе разработки *ПО*. (использование памяти)
- **Четвертый уровень** - это оценочный элемент метрики (*вес*), который используется для оценки количественного или качественного значения отдельного атрибута показателя *ПО*

Оценка качества ПО согласно четырехуровневой модели качества начинается с нижнего уровня иерархии, т.е. с самого элементарного свойства оцениваемого атрибута

При измерении показателей используют следующие типы мер:

меры размера в разных единицах измерения (количество функций, размер программы, объем ресурсов и др.);

меры времени - периоды реального, процессорного или календарного времени (время функционирования системы, время выполнения компонента, время использования и др.);

меры усилий - продуктивное время, затраченное на реализацию проекта (производительность труда отдельных участников проекта, коллективная трудоемкость и др.);

меры интервалов между событиями, например, время между последовательными отказами;

счетные меры - счетчики для определения количества обнаруженных ошибок, структурной сложности программы, числа несовместимых элементов, числа изменений (например, число обнаруженных отказов и др.).

- Если в требованиях к ПО было указано получить несколько показателей, то показатель умножается на соответствующий весовой коэффициент, а затем суммируются все показатели для получения комплексной оценки уровня качества ПО.
- На основе измерения количественных характеристик и проведения экспертизы качественных показателей с применением весовых коэффициентов, нивелирующих разные показатели, вычисляется итоговая оценка качества продукта путем суммирования результатов по отдельным показателям и сравнения их с эталонными показателями ПО (стоимость, время, ресурсы и др.).
- Т.е. при проведении оценки отдельного показателя с помощью оценочных элементов просчитывается весомый коэффициент k - метрика, j -показатель, i -атрибут. Например, в качестве j - показателя возьмем переносимость. Этот показатель будет вычисляться по пяти атрибутам (i=1..5), причем каждый из них будет умножаться на соответствующий коэффициент .
- Когда все атрибуты оценены по каждому из показателей качества, производится суммарная оценка отдельного показателя, а потом и интегральная оценка качества с учетом весовых коэффициентов всех показателей ПО.

Объектно-ориентированные метрики

- Объектно-ориентированные метрики вводятся с целью: улучшить понимание качества продукта, оценить эффективность процесса конструирования, улучшить качество работы на этапе проектирования.
- Класс – представляет собой основной, фундоментальный элемент объектно-ориентированной инфор- мационной системы. Метрики для отдельного класса, иерархии классов и взаимодейст- вия классов являются наиболее ценными для руководителя проекта, который оценивает качество работы.

Объектно-ориентированные метрики

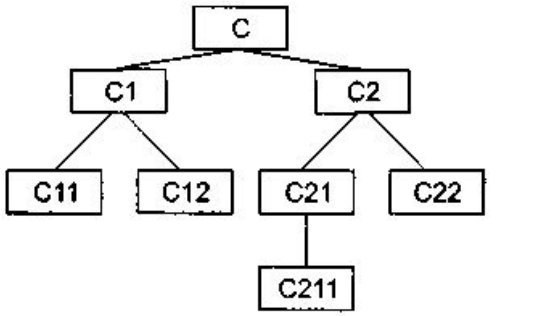
1. метрики Чидамбера и Кемерера

Метрика 1: Взвешенные методы на класс WMC (Weighted Methods Per Class) – относительная мера сложности класса

- в классе *C* определены *n* методов со сложностью $c_1 \ldots, c_i, \ldots, c_n$ $WMC = \sum_{i=1} C_i$

- Упрощенная версия метрики $c_i = 1$, WMC –кол-во методов в классе

Метрика 2: Высота дерева наследования DIT (Depth of Inheritance Tree)



- *DIT* определяется как максимальная длина пути от листа до корня дерева наследования классов

+большая DIT - многие методы могут использоваться многократно.

- большая сложность проекта

Метрика 3: Количество детей NOC (Number of children)

- NOC равно количеству детей, то есть количеству непосредственных наследников класса в иерархии классов.
- *следует строить сбалансированные по высоте и ширине структуры наследования: обычно не выше, чем 7 ± 2 уровня, и не шире, чем 7 + 2 ветви [Г.Буч]*

Метрика 4: Сцепление между классами объектов CBO (Coupling between object classes)

- *CBO* — количество сотрудничеств, предусмотренных для класса, то есть количество классов, с которыми он соединен. Соединение означает, что методы данного класса используют методы или экземплярные переменные другого класса.

Метрика 5: Отклик для класса RFC (Response For a Class)

Множество отклика класса RS — множество методов, которые могут выполняться в ответ на прибытие сообщений в объект этого класса.

$$RS = \{M\} U_{all\_i} \{R_i\}$$

где $\{R_i\}$ — множество методов, вызываемых методом i, $\{M\}$ — множество всех методов в классе

- RFC - количество методов во множестве отклика, **RFC = card{RS}.**

  RFC —количество методов класса плюс количество методов других классов, вызываемых из данного класса.

Наихудшая величина отклика может использоваться при определении времени тестирования.

Метрика 6: Недостаток связности в методах LCOM (Lack of Cohesion in Methods)

- Каждый метод внутри класса обращается к одному или нескольким свойствам (экземплярным переменным). Метрика *LCOM* показывает, насколько методы не связаны друг с другом через свойства (переменные). Если все методы обращаются к одинаковым свойствам, то LCOM = 0.

- *НЕ СВЯЗАНЫ — количество пар методов без общих экземплярных переменных;*
- *СВЯЗАНЫ — количество пар методов с общими экземплярными переменными.*
- I_j *— набор экземплярных переменных, используемых методом* M_j

метрики Чидамбера и Кемерера-6

$$НЕ\_Связаны = card\{I_{ij} \mid I_i \cap I_j = 0\}$$

$$Связаны = card\{I_{ij} \mid I_i \cap I_j \neq 0\}$$

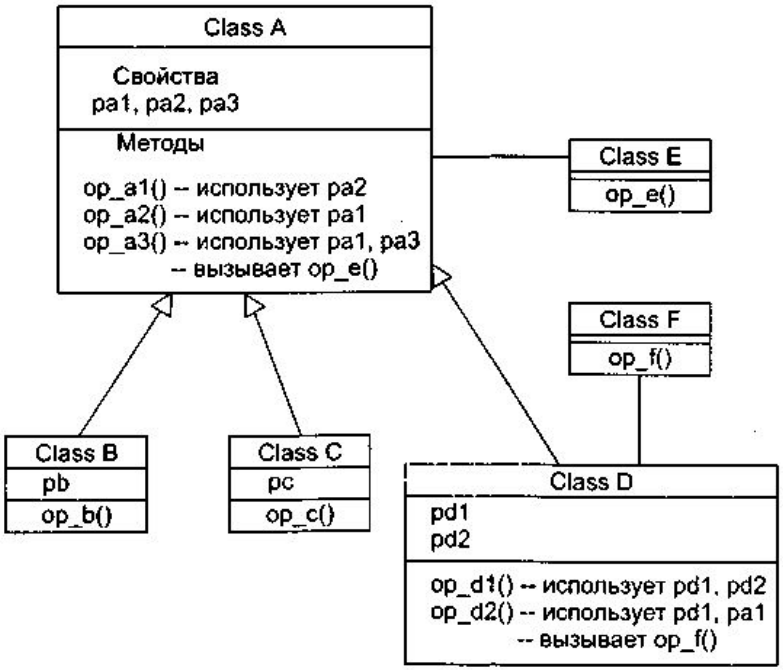
$$LCOM = \begin{cases} \text{НЕ СВЯЗАНЫ - СВЯЗАНЫ, если (НЕСВЯЗАНЫ} > \text{СВЯЗАНЫ);} \\ 0 \text{ в противном случае.} \end{cases}$$

LCOM — это количество пар методов, не связанных по свойствам класса, минус количество пар методов, имеющих такую связь.

LCOM пример

- **Пример 1:** В классе имеются методы: *M1, M2, M3, M4*. Каждый метод работает со своим набором экземплярных переменных:
- $I_1=\{a, b\}$; $I_2=\{a, c\}$; $I_3=\{x, y\}$; $I_4=\{m, n\}$.
- НЕ СВЯЗАНЫ = card $(I_{13}, I_{14}, I_{23}, I_{24}, I_{34}) = 5$;
- СВЯЗАНЫ = card $(I_{12}) = 1$.
- **LCOM = 5-1=4.**

Использование метрик Чидамбера и Кемерера



Для вычисления метрики LCOM надо определить количество пар методов класса:

$$C_m^2 = m!/(2(m-2)!)$$

Имя класса	WMC	DIT	NOC	CBO	RFC	LCOM
Class A	3	0	3	1	4	1
Class B	1	1	0	0	1	0
Class C	1	1	0	0	1	0
Class D	2	1	0	2	3	0