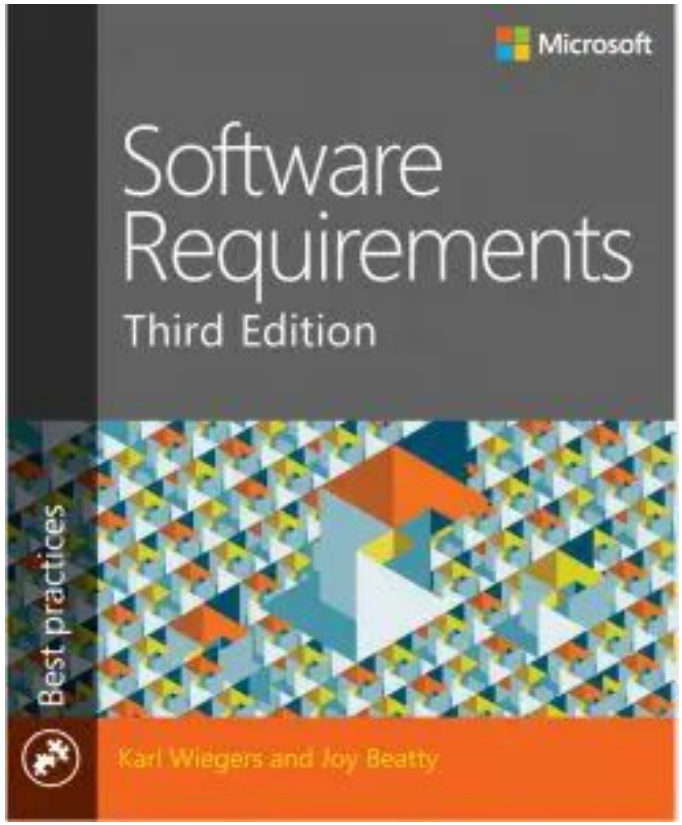
ТЕМА 7
Написание качественных требований

Writing Quality Requirements

După - Karl Wiegers and Joy Beatty



- ***Из предидущих лекций***
- ***Мы уже знаем***,
- что ***Понятие Пользователь включает в себя все заинтересованные стороны (стейкхолдеры):***

- владелец системы,
- Тот кто финансирует систему,
- - "конечный пользователь системы ",
- Администратор проектной компании,
- Менеджер проекта
- Аналитики разработчики требований,
- Программисты
- .....

- *А также мы знаем*
- *что: Спецификации требований к информационной системе и программному обеспечению (SRS) — разрабатываются с участием этих заитересованных сторон (стейкхолдеров), в успехе проекта*

- ***Необходимо отметить,*** что:
- ***Многие спецификации требований*** к ***Системе и Программному обеспечению (SRS) часто заполнены требованиями с ошибками,***
- ***Немногие разработчики программного обеспечения научились составлять качественные требования,***
- ***Есть мало доступных примеров хороших требований, из которых можно извлечь опыта.***
- Немногие проекты имеют идеальные требования для их применения по обмену

- ***Немногие компании готовы размещать спецификации своих продуктов в открытом доступе.***
- ***Не существует формализованного метода написания хороших требований — во многом это вопрос опыта,***

- ***Понятия ошибок***
- ***Гораздо дороже исправлять дефектные требования на более поздних этапах.***
-

- ***Типы ошибок. Примеры***
- ***Большинство ошибок вызвано:***

- Пропусками (Omissions)

-искаженными фактами

-Несоответствием (противоречиями)

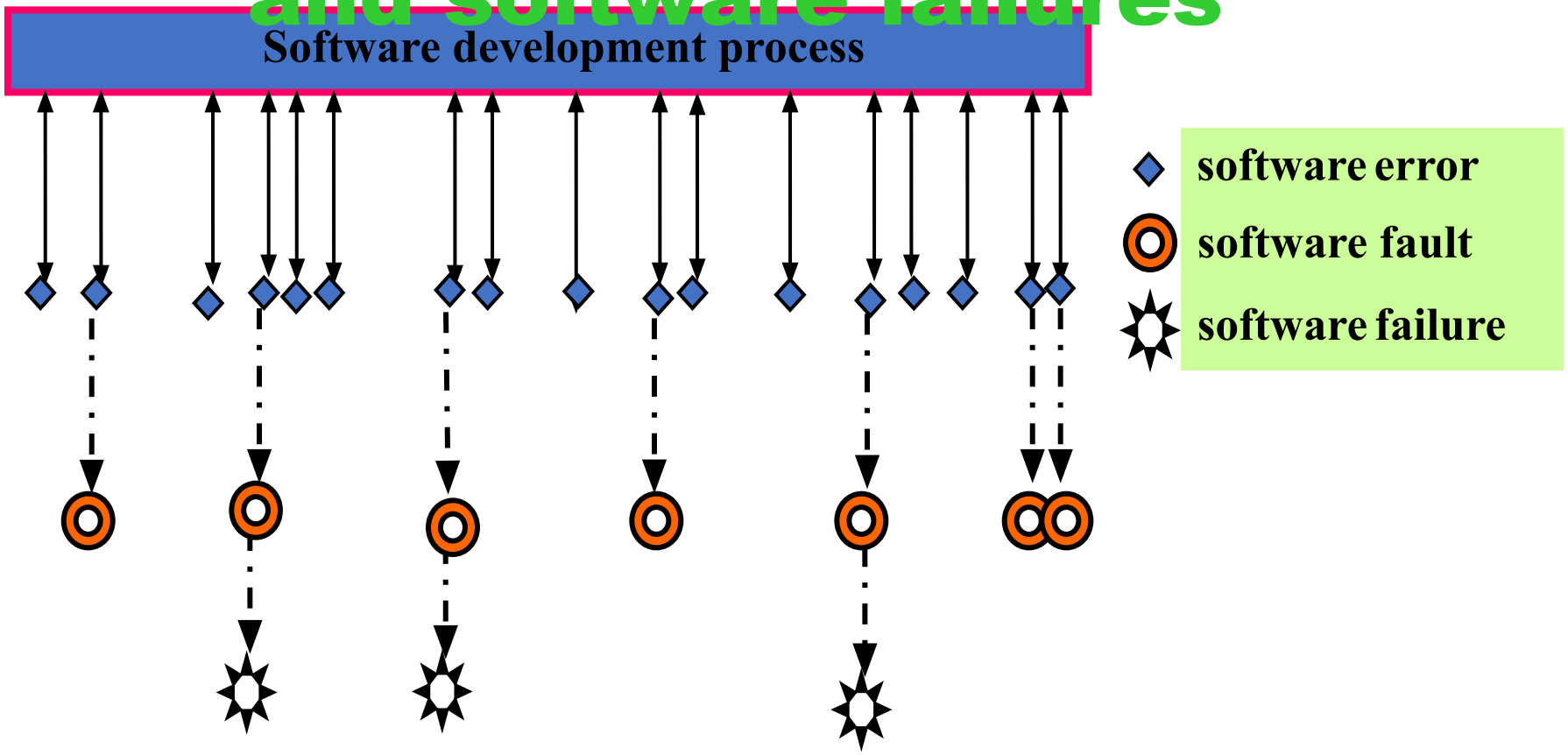
-неоднозначность (множеством значений)

- ***Основные определения***
- ***- Программная ошибка (software error)- совершается программистом:***

- Синтаксическая (грамматическая) ошибка
- Логическая ошибка (умножение вместо сложения)

- ***Программный дефект (software fault)*** - Часть программного обеспечения может не выполняться
- ***Программный сбой (software failure)*** - Программная ошибка становится программным сбоем, тогда когда она активна.

Software errors, software faults and software failures

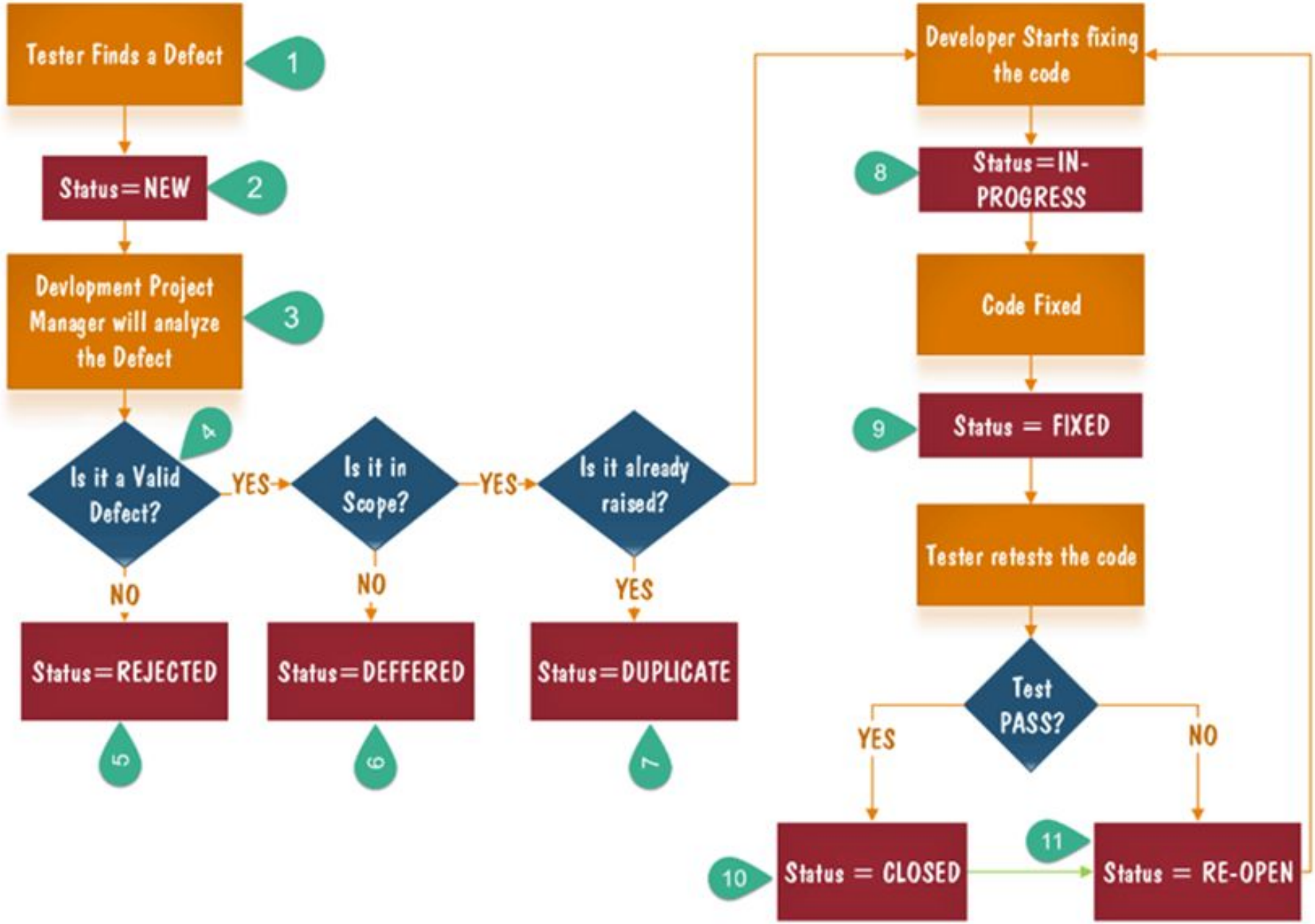


- **Жизненный цикл дефекта?**
- ***Defect Life Cycle*** или ***Жизненный цикл ошибки программного обеспечения*** — это определенный набор состояний, через которые проходит дефект или ошибка на протяжении присутствия в ***программном обеспечение***.
- ***Цель жизненного цикла дефекта*** состоит в том, ***чтобы легко координировать и сообщать о текущем статусе дефекта, который изменяется,*** различным правопреемникам, а также ***сделать процесс устранения дефекта систематическим и эффективным.***
- ***Статус дефекта***
- **Defect Status** или ***Статус ошибки в жизненном цикле дефекта — это текущее состояние, в котором дефект или ошибка находится в настоящее время.***
- Цель статуса дефекта — точно передать текущее состояние или развитие дефекта или ошибки, чтобы лучше отслеживать и понимать фактическое развитие жизненного цикла дефекта.
- Количество состояний, через которые проходит дефект, варьируется от проекта к проекту.

- диаграмма жизненного цикла, покрывающая все возможные состояния:
- ***New:*** Когда новый дефект регистрируется и публикуется впервые. Ему присвоен статус НОВЫЙ.
- ***Assigned:*** Как только ошибка публикуется тестировщиком, руководитель тестировщика утверждает ошибку и назначает ошибку команде разработчиков.
- ***Open:*** Разработчик начинает анализировать и работает над исправлением дефекта
- ***Fixed:*** Когда разработчик вносит необходимое изменение кода и проверяет изменение, переводит статус ошибки как Fixed.

- ***Pending retest***: после устранения дефекта разработчик передает тестеру конкретный код для повторного тестирования кода. Поскольку тестирование программного обеспечения остается незавершенным с конца тестировщика, присвоенный статус **pending retest.**
- ***Retest:*** На этом этапе тестировщик выполняет повторное тестирование кода, чтобы проверить, исправлен ли дефект разработчиком или нет, и меняет статус на **Re-test.**
- ***Verified:*** тестер повторно тестирует ошибку после того, как она была исправлена разработчиком. Если в программном обеспечении не обнаружена ошибка, то ошибка фиксируется и присваивается статус ***verified.***
-

- ***Reopen:*** если ошибка сохраняется даже после того, как разработчик ее исправил, ***тестер изменяет статус на reopened***. Ошибка снова проходит жизненный цикл.
- ***Closed:*** Если ошибка больше не существует, тестер присваивает статус ***Closed.***
- ***Duplicate:*** Если дефект повторяется дважды или дефект соответствует одной и той же концепции ошибки, статус меняется на ***duplicate***.
- ***Rejected:*** Если разработчик считает, что дефект не является подлинным дефектом, он меняет дефект на *rejected*.
- ***Deferred:*** Если текущая ошибка не имеет первостепенного приоритета и ожидается, что она будет исправлена в следующем выпуске, то статус Deferred присваивается таким ошибкам
- ***Not a bug:*** Если это не влияет на функциональность приложения, багу присваивается статус ***Not a bug***.


Tester Finds a Defect
1
Status = NEW
2
Devlopment Project Manager will analyze the Defect
3
Is it a Valid Defect?
4
YES
Is it in Scope?
YES
Is it already raised?
NO
Status = REJECTED
5
NO
Status = DEFFERED
6
YES
Status = DUPLICATE
7
Developer Starts fixing the code
8
Status = IN-PROGRESS
Code Fixed
9
Status = FIXED
Tester retests the code
Test PASS?
YES
NO
10
Status = CLOSED
11
Status = RE-OPEN

- Некоторые примеры дефектов
- программного обеспечения

- Каждую неделю появляются новые истории о сбоях программного обеспечения во множестве отраслей;
- вызывая хаос, останавливая бизнес или даже вызывая гибель людей.
- Каждый год ***Tricentis*** собирает новости со всего мира, кульминацией которых является Tricentis Software Fail Watch,
- ***Tricentis*** - лидер в области тестирования программного обеспечения
- https://www.peerspot.com/products/tricentis-tosca-reviews

- ***Инфузионные насосы отзывают из-за фатального дефекта CareFusion*** — производитель медицинского оборудования, который за последние годы несколько раз отзывал продукцию в экстренных случаях.
- В 2015 году ***насос Alaris Pump*** компании CareFusion был отозван из-за программной ошибки, из-за которой насос, предназначенный для автоматической доставки лекарств и жидкостей больничным пациентам, задерживал инфузии.
- Последствия, которые варьировали от удержания препарата в критических точках до случайной передозировки, - могут быть смертельными.

- ***Программная ошибка в истребителях F-35 вызывала проблемы с обнаружением целей***
- Серьезная проблема с программным обеспечением в самолете F-35 Joint Strike Fighter привлекла внимание специалистов.
- Авиаинженеры выявили программную ошибку, из-за которой самолеты при полете парами неправильно обнаруживали цели.
- Поскольку каждый самолет в строю обнаруживает цель под разными углами, кажется, что программное обеспечение не может расшифровать, есть ли только одна цель или несколько.
- Как выразилось одно информационное агентство, у F-35 «двоится в глазах».

- ***Ошибка в программе помогла ограбить банк***
- Эта история состоит из двух частей:
- программная ошибка.Первая часть: группа хакеров-воров взломала банковскую систему Бангладеш для кражи средств.
- Группа, использовавшая орфографическую ошибку, успешно перевела 81 миллион долларов в четырех транзакциях.
- По данным властей Бангладешского банка, принтер настроен для автоматической печати завершенных транзакций.
- Проблема в системе (будь то случайная или созданная ворами) ***прервала автоматический процесс печати,***
- дело было обнаружено только через несколько дней,, что дало ворам достаточно времени, чтобы замести следы.

- ***Потеря данных в Gitlab*** (GitLab is an open source code repository and collaborative software development platform for large DevOps and DevSecOps projects. GitLab is free for individuals.)
- Два года назад известная платформа для совместной работы над кодом GitLab столкнулась с серьезной потерей данных в результате одного из самых серьезных сбоев в ИТ-мире.
- GitLab изначально использовал один сервер базы данных, но решил протестировать решение с использованием двух серверов.
- Они планировали скопировать данные из производственной среды в тестовую среду.
- При этом автоматические механизмы начали удалять из базы учетные записи, которые были определены как опасные. В результате возросшего трафика процесс копирования данных начал замедляться, а затем и вовсе остановился из-за несоответствия данных.
- Во избежание повреждения информация из производственной базы данных была удалена в процессе копирования.

- ***British Airways "Техническая проблема«***
- Британская авиакомпания British Airways в 2018 году сообщила о проблеме с ИТ-системой, которая привела к задержке сотен рейсов в Великобритании, а десятки рейсов были полностью отменены.
- Этот сбой затронул три аэропорта Великобритании и тысячи пассажиров, которым пришлось перебронировать билеты или пройти регистрацию с использованием ручных систем.

- ***Amazon AWS Outage***
- . AWS-ul Amazon, considerat a fi unul dintre ***cele mai fiabile servicii de găzduire,*** a suferit o întrerupere gravă pe coasta de est a SUA în 2017.
- Infrastructura AWS acceptă milioane de site-uri, ceea ce înseamnă că, atunci când serverele companiei cad, aceasta cauzează o mulțime de probleme pe internet. Nu a fost o surpriză faptul că „dificultățile tehnice majore” ale ASW au dus la probleme fără precedent pentru sute de site-uri web.
- Multe companii de diferite dimensiuni și din diferite industrii își stochează datele în centrele de date ale AWS. Printre ele sunt Netflix, Slack, Business Insider, IFTTT, Nest Trello, Quora și Splitwise.
- Mulți dintre ei au fost afectați de întreruperea menționată mai sus.

- ***Сбой Amazon AWS***. *(Amazon Web Services (AWS) – это самая распространенная в мире облачная платформа с широчайшими возможностями, предоставляющая более 200 полнофункциональных сервисов для центров обработки данных по всей планете)*
- Amazon AWS, считающийся одним из самых надежных хостинговых сервисов, в 2017 году потерпел серьезный сбой на восточном побережье США.
- Инфраструктура AWS поддерживает миллионы сайтов, а это значит, что когда серверы компании выходят из строя, это вызывает массу проблем в Интернете. Неудивительно, что «серьезные технические трудности» ASW привели к беспрецедентным проблемам для сотен веб-сайтов.
- Многие компании разного размера и отрасли хранят свои данные в центрах обработки данных AWS.
- Среди них Netflix, Slack, Business Insider, IFTTT, Nest Trello, Quora и Splitwise.Многие из них пострадали от вышеупомянутого сбоя.

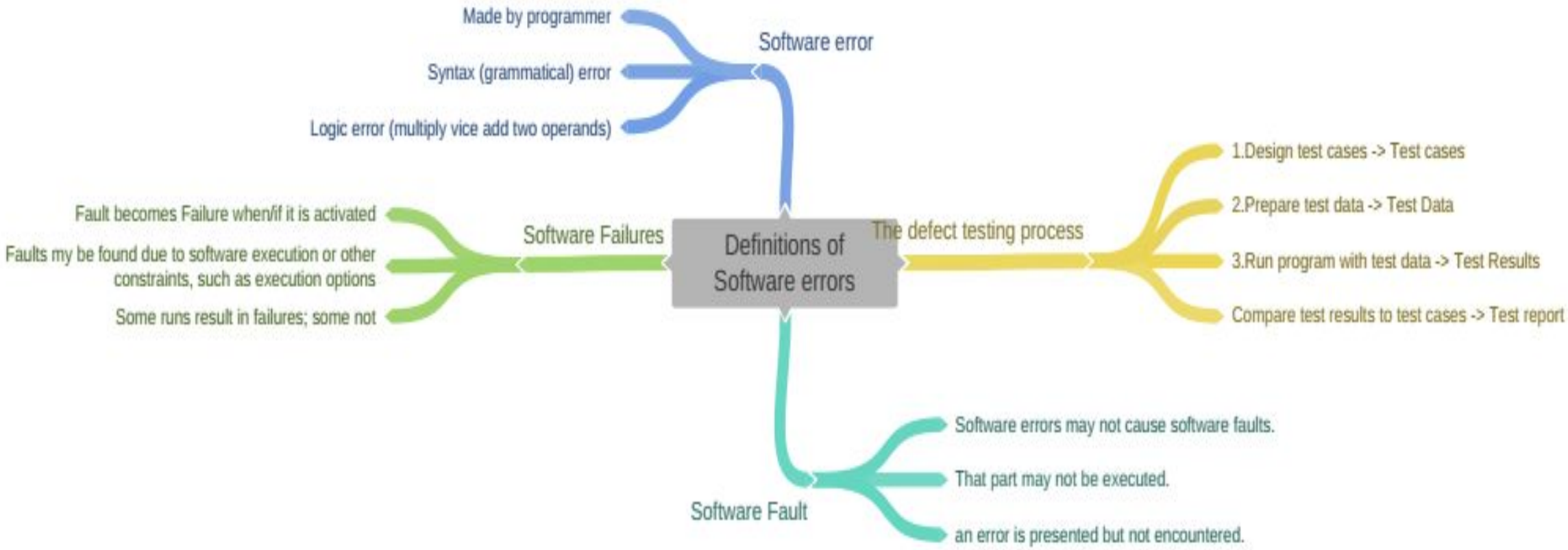
- ***Проблема безопасности с Google Plus***
- Уязвимость в Google+ раскрыла личную информацию почти 500 000 человек, использовавших социальную сеть в период с 2015 года по март 2018 года, специальный API, который можно было использовать для получения доступа к закрытой информации.
- Программная уязвимость позволяла сторонним разработчикам видеть имя, адрес электронной почты, статус занятости, пол и возраст пользователей сети.
- Ошибка была обнаружена в марте 2018 года и немедленно исправлена.

- ***Утечка данных пользователей из Facebook***
- Facebook, чья способность управлять личной информацией уже была поставлена под сомнение, подтвердил, что почти 50 миллионов учетных записей могут быть подвержены риску.
- Хакеры воспользовались уязвимостью в системе, которая позволила им получить доступ к учетным записям и, возможно, личной информации пользователей Facebook.
- Атака была обнаружена 25 сентября 2018 года. По данным источников The New York Times, 3 программных недостатка в системах сети позволили хакерам получить доступ к учетным записям пользователей, в том числе генерального директора Facebook Марка Цукерберга.
- Представители сети заявили, что хакеры воспользовались уязвимостью в коде «Просмотреть как» — функции, позволяющей проверить внешний вид профиля с точки зрения других людей.

- ***Lista cazurilor poate fi continuată în fiecare zi!!!***

Причины программных ошибок

согласно Вигерс Карл, Битти Джой
Разработка требований к программному обеспечению.



software error - программная ошибка
software failures - программные сбои
software fault – программные дефект

The Nine Causes of Software Errors

The nine causes of software errors are:

1. **Faulty requirements definition**
2. **Client-developer communication failures**
3. **Deliberate deviations from software requirements**
4. **Logical design errors**
5. **Coding errors**
6. **Non-compliance with documentation and coding instructions**
7. **Shortcomings of the testing process**
8. **User interface and procedure errors**
9. **Documentation errors**

E binesă fii familiarizat cu acestea

Mai mult, unele din ele le puteți defini în propriile cuvinte și puteți oferi un exemplu!?

- Программная ошибка означает воспроизводимый дефект или их комбинации в Программном обеспечении, которые приводят к отказу Программного обеспечения при использовании в соответствии с Документацией.
- Не являются Ошибкой программного обеспечения, вызванные:

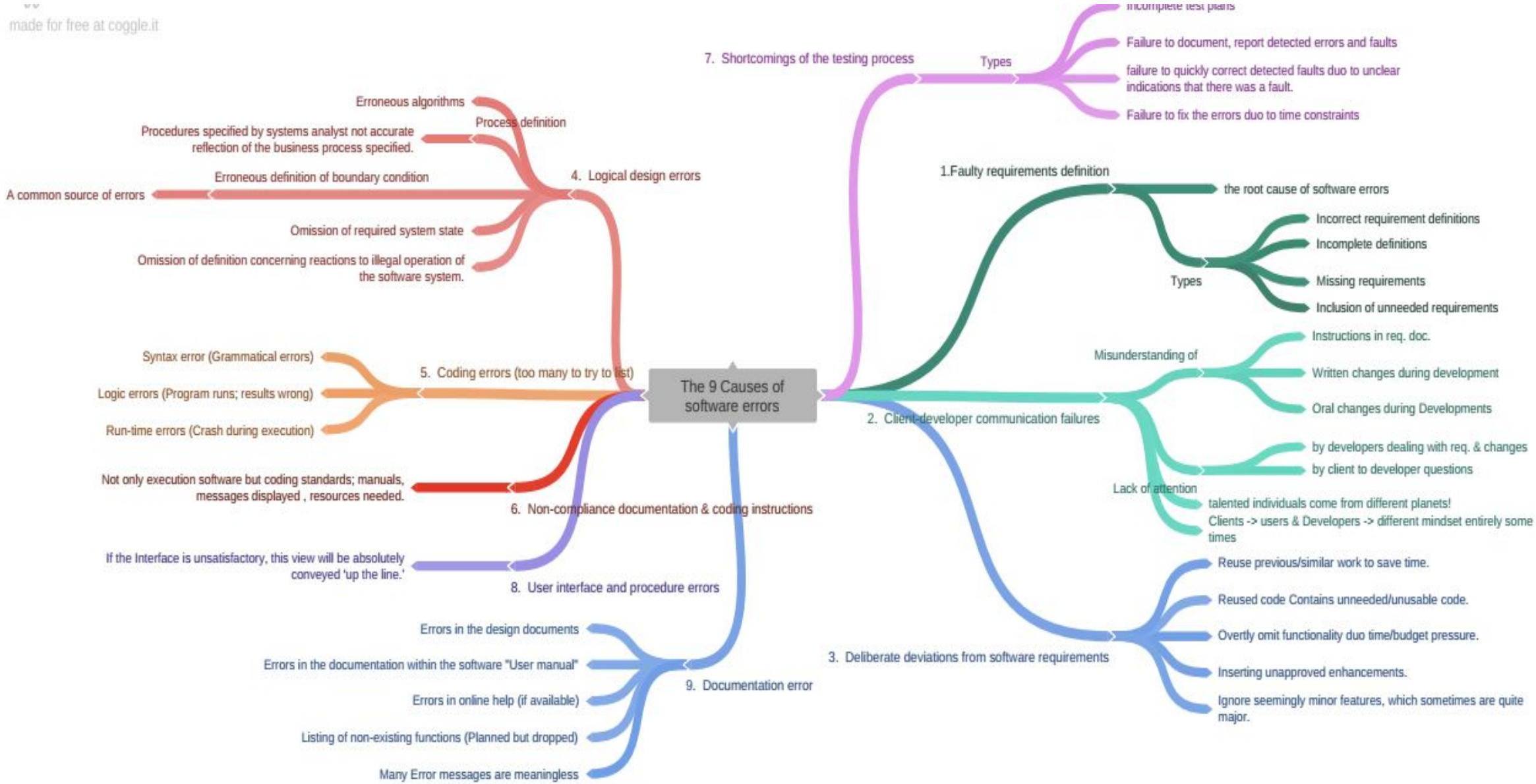
(а) небрежностью Пользователя,

(б) любой несанкционированной модификацией или изменением, внесенным Пользователем в Программное обеспечение,

(в) данными, которые не соответствуют формату данных, указанному Пользователем,

(г) ошибкой оператора или

(д) использование не соответствующие поддерживаемой Пользователем технической среде, указанной в Документации.



- Причины ошибок ПО целесообразно увязывать с процессом создания ПО ИС.
- Можно рассматривать создание ПО как ряд процессов перевода (трансляции), начинающихся с постановки задачи и заканчивающихся текстом программы и документацией на нее.
- Отсюда следует, что программирование — это процесс решения задачи, а ПО — некоторый специальный набор информационных элементов, описывающих реализацию решения этой задачи в соответствии с установленным алгоритмом.
- Тогда процесс производства ПО можно представить как набор процессов трансляции.
- Трансляция начинается с момента преобразования постановки задачи до получения сначала промежуточных результатов и заканчивается детальным набором машинных

***1. Неверное определение требований.* - Обычно считается *основной причиной ошибок в программном обеспечении*.**

- ***Неверные определения требований-*** Упрощенные, неправильные определения (формулы и т.д.)
- ***Неполные определения-*** Неясные или неявные требования отсутствующие требования- Просто "отсутствует" (например, код элемента программы)
- ***Включение ненужных требований-*** многие проекты провалились, потому что в них было слишком много требований, которые никогда не будут использованы.
- ***Влияние бюджета***, сложности, времени разработки, ...

2. Сбои связи между клиентом и разработчиком

- Непонимание инструкций в документации требований (письменные/графические инструкции)
- Непонимание написанных изменений во время разработки.
- Непонимание устных изменений во время развития.
- Недостаток внимания:- **Клиента** к сообщениям **Аналитика** по изменение требований и
- на реакцию **Клиента** на сообщения **Аналитика**

Очень часто кажется, что эти талантливые личности родом с разных планет. Клиент представляет пользователей;
Аналитик представляет другое мышление!

3. Преднамеренные отклонения от письменных требований к программному обеспечению

- Разработчик повторно использует предыдущую/подобную работу для экономии времени.
- Часто повторно используемый код требует модификаций, которые могут содержать ненужный/непригодный для использования посторонний код.
- Письменые требоваия предполагает, что разработчики, очевидно, могут пропустить некоторые функции из-за нехватки времени/бюджета.

Или

- Другой плохой выбор; который будет обнаружен Системным тестирование ко всеобщему ужасу!

Или

- Разработчик применяет некоторые неутвержденные "улучшения" (идеальное кодирование; новая сортировка/поиск...); он также может игнорировать некоторые, казалось бы, незначительные функции, которые иногда очень важны.

- ***4. Ошибки логического проектирования***
- Ошибочные алгоритмы в Определения, которые представляют требования к программному.

 - Неверные формулы;

 - Неверная логика в таблицах;

 - Неверные описания в тексте

  Ошибочные Определения в процессах: процедуры, указанные системным аналитиком, которые не точно отражают фактический бизнес-процесс.

- Примечание.
- Не все ошибки обязательно являются ошибками программного обеспечения.
- - Процедурная ошибка не является частью программной системы... Но это все равно ошибка!
- Неправильное определение граничного условия, таких как: «Абсолюты» как «не более», «менее», «в **n** раз и более»; «В первый раз» и др.

5. Ошибки кодирования

- Их Слишком много, чтобы перечислить всех.

Некоторые из них:

- Синтаксические ошибки (грамматические ошибки)
- Логические ошибки (программа запущена, результаты неверны)
- Ошибки выполнения (сбой выполнения)
-
-

- Если Вы знаете и другие дополните это список.
-

6. Несоблюдение документации и инструкций по кодированию

- - Несоблюдение опубликованных шаблонов (форматов).
- - Несоблюдение стандартов кодирования
- - Размер программы;
- - Другие программы, которые должны работать в данной среде!
- Кодирование- Элементы данных и коды:
- Процедуры разработки: необходимая документация, руководства и инструкции по эксплуатации;
- ***Группа*** **Software Quality Assurance** ***(SQA):*** тестирует не только исполнительное программное обеспечение, но и ***стандарты кодирования; руководства, отображать сообщения; необходимые ресурсы; имя ресурса (имена файлов, имена программ,*** ...)

7. Недостатки процесса тестирования

- - ***часто случается что это, та часть процесса разработки, которая сокращается больше всего!***

 - Неполные планы испытаний

 - Части приложения не тестировались или тестировались поверхностно!

 - Поверхностное тестирование- Лимипроф условия…б Тестирование маршрута, тестирование ветвей...

 Неспособность документировать и сообщать об обнаруженных ошибках и дефектах

 - Много уровней тестирования .....

 Неспособность быстро реагировать на обнаруженные сбои из-за нечетких указаний на то, что произошла «ошибка».

 Ошибка в исправления ошибки из-за нехватки времени

  Много философий относительно серьезности ошибки.

8. Ошибки Пользовательского интерфейса и процедуры

- **Помните: для пользователя интерфейс — это вся система.**
- Если интерфейс не устраивает, то эта слава будет распространятся очень быстро

9. Ошибки при документировании

 Ошибки в проектной документации.

- Если реализованный дизайн отличается от представленого в документе, ***то это проблема для редизайна и повторного использования***.

 Ошибки в документации в руководствах пользователя, руководстве оператора, других руководствах (установка ...)

- Ошибки при доступе в режиме онлаин, если таковой имеется.
- Наличие несуществующих функций программного обеспечения,

ранее Запланировано, но заброшены и остались в документации!

- Некоторые сообщения об ошибках совершенно

The Nine Causes of Software Errors

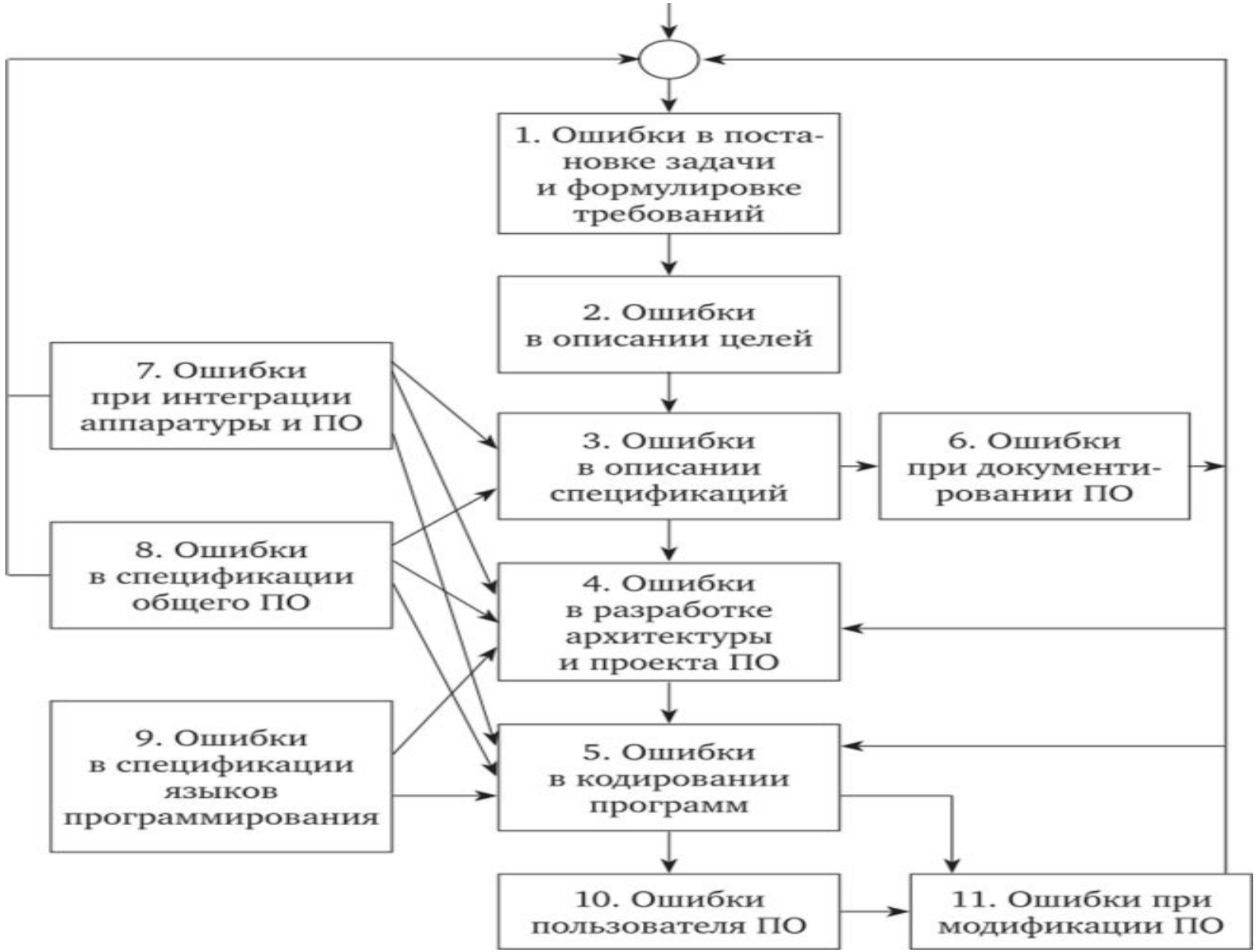
The nine causes of software errors are:

1. **Faulty requirements definition**
2. **Client-developer communication failures**
3. **Deliberate deviations from software requirements**
4. **Logical design errors**
5. **Coding errors**
6. **Non-compliance with documentation and coding instructions**
7. **Shortcomings of the testing process**
8. **User interface and procedure errors**
9. **Documentation errors**

E binesă fii familiarizat cu acestea

Mai mult, unele din ele le puteți defini în propriile cuvinte și puteți oferi un exemplu!?

- Другой подход


1. Ошибки в постановке задачи и формулировке требований
2. Ошибки в описании целей
3. Ошибки в описании спецификаций
4. Ошибки в разработке архитектуры и проекта ПО
5. Ошибки в кодировании программ
6. Ошибки при документировании ПО
7. Ошибки при интеграции аппаратуры и ПО
8. Ошибки в спецификации общего ПО
9. Ошибки в спецификации языков программирования
10. Ошибки пользователя ПО
11. Ошибки при модификации ПО

- ***1. Ошибки в постановке задачи и формулировке требований.***
- Процесс создания ПО начинается с описания решаемой задачи, которое представляет собой перечень требований пользователя.
- Этот перечень разумнее всего составлять Аналитикок и пользователем совместно.
- При составлении данного перечня всегда открываются «большие возможности» для ошибок: например, потребности пользователя неверно поняты и учтены разработчиком, либо не учтены в полном объеме, либо пользователь не сумел адекватно выразить свои потребности.
- Ошибки этого уровня обходятся чрезвычайно дорого.

2. Ошибки в описании целей.

- Второй процесс состоит в переводе требований пользователя в цели программы.
- Ошибки на этом этапе возникают, когда неверно интерпретируются требования, не удается выявить все требующие компромиссного решения проблемы или приняты неправильные решения.
- Они могут возникнуть также в случае, когда не сформулированы цели необходимые, не поставленные явно в требованиях пользователя.

3. Ошибки в описании спецификаций.

- Третий процесс предназначен для точного описания поведения ИС с точки зрения пользователя.
- Иначе говоря, должна решаться задача преобразования целей программы в ее внешние спецификации.
- Это самый объемный и трудоемкий этап разработки ПО.
- Он наиболее подвержен ошибкам, которые здесь, как правило, наиболее многочисленные и наиболее серьезные.

4. Ошибки в разработке архитектуры и проекта И.С.

- Четвертый этап представляет собой несколько процессов трансляции: от внешнего описания готового продукта до получения детального проекта.
- На этом этапе внешнее описание переводится в структуру компонент *И.С.* и создаются алгоритмы программных модулей и составных программ.
- Поскольку приходится иметь дело с большими объемами информации, то вероятность внесения ошибок становится чрезвычайно высокой.

5. *Ошибки в кодировании программ.*

Пятый процесс проектирования — перевод алгоритма в предложения языка программирования, а также компиляция, т.е. перевод представления программы на языке программирования в объектный код.

Ошибки на этом этапе трансляции имеют место, однако их вероятность возникновения в связи с автоматизацией программирования много меньше, чем на предыдущих этапах проектирования И.С.

- **6. *Ошибки при документировании ПО.***
- Шестой этап завершает разработку И.С. как законченного продукта, который может быть передан пользователю.
- На этом этапе создается документация на И.С. , в том числе руководства по инсталляции программ и их эксплуатации.
- Качество документации оказывает существенное влияние на возможность возникновения программных ошибок. В работе [5] подчеркнуто, что «прочитав руководство, пользователь начнет работать с программой и обнаружит, что она ведет себя не так, как он ожидал, — это и является, по определению, ошибкой в программе». К сожалению, часто качество документации не соответствует предъявляемым к нему требованиям. Это обусловлено следующим: разработчики программ редко обладают качествами технических писателей, не любят и не всегда умеют кратко, четко и понятно описать программу, правила и порядок работы с ней; технические же писатели недостаточно знают и понимают данную программу.

- 7. *Ошибки при интеграции аппаратуры и И.С.*
- В процессе разработки программы системные спецификации оборудования используются в качестве входных данных.
- Решения, принимаемые разработчиками программ, во многом зависят от характеристик компьютерных и периферийных средств информационной техники, от характеристик телекоммуникационных средств.
- Незнание или неправильное толкование этих данных может привести к ошибкам в проектируемых *И.С.*

8. Ошибки в спецификации общего ПО.

- Решения разработчиков прикладных программ ограничены возможностями общего ПО, зависят от их средств динамического распределения ресурсов, диспетчеризации вычислений, ввода-вывода информации.
- Незнание или неправильное толкование возможностей общего ПО может привести к ошибкам в проектируемую И.С.

9. Ошибки в спецификации языков программирования.

Написание программы обработки информации может быть выполнено на одном или нескольких языках программирования.

Конечный продукт формируется с помощью одного языка программирования.

Неправильное использование языковых конструкций, синтаксиса и семантики языка (языков) программирования может послужить серьезной причиной возникновения программных ошибок.

•

10. Ошибки пользователя ПО.

- Если работает невнимательный пользователь, то вероятность того, что он сделает ошибку, велика.
- Ошибки пользователя часто приводят к возникновению новых, непредвиденных состояний системы, что, в свою очередь, может повлечь за собой повторение в полном или частичном объеме всех работ, связанных с созданием программ.
- Следовательно, повторяются ситуации, которые приводят к корректировке целей, спецификаций, алгоритмов и, как следствие, к возникновению новых ошибок.

11. *Ошибки при модификации ПО.*

Эксплуатация И.С. является продолжением этапов разработки.

На этом этапе жизненного цикла И.С. решаются задачи устранения обнаруженных ошибок, которые не направлены на совершенствование программы, но по существу приводят к созданию новой ее версии при каждом случае восстановления работоспособности программы.

Процедуру устранения обнаруженных ошибок принято квалифицировать как *модификацию* данной программы.

В процессе эксплуатации программы возможно изменение некоторых целевых установок, приводящих к необходимости ее доработки. Кроме того, возникают обстоятельства, требующие совершенствования программы. В обоих указанных случаях изменяются некоторые функции программ. Этот процесс изменения программы называется *модернизацией.* Часто указанные понятия объединяются одним термином «модификация».

Характеристики качества требований

- В отличие от производственной деятельности, результаты которой характеризуются материальными продуктами, сфера информационных технологий и особенно производство программного обеспечения это нематериальные продукты или содержит такие продукты.
- В этой области в основном задействованы Интеллектуальные ресурсы.
- С диверсификацией ИТ-продуктов, с увеличением количества поставщиков возникла необходимость сравнения производительности соответствующих продуктов.
- Кроме того, элементы сравнения появились в отношении установления цен на программные продукты и определения результатов работы программистов.

- Требование определяется ***как представление потребности или ожидания, которое заявлено клиентом***, как правило, неявно или обязательно.
- ***Заявленное требование*** – это требование, изложенное в документе.
- Под несоответствием понимается невыполнение требования, при этом
- Дефект – это невыполнение требования, связанного с предполагаемым или указанным использованием.

Характеристики Качества требований к программному обеспечению

După - Karl Wiegers and Joy Beatty

- **Правильная**
- **Выполнимая**
- **Необходимое**
- **Полное**
- **Последовательное**
- **Приоритизиронность -**
- **Недвусмысленность -(однозначна)**
- **проверяемое - (поддающийся проверке)**
- **Модифицируемое**
- **прослеживаемое**

- ***Правильное требование***

- ***Точно описывает функциональность***, которая должна быть реализована

- источники "Правильного" - требованияа):

а)Текущий ***(реальный) клиент,***

b) ***Спецификация (документированные) требований*** к системе высокого уровня.

Для определения правильности требований пользователя

- Необходимо участие их представителей, которые :
- ***а) будут проверять написанные требования***,
- ***б) Избавят разработчиков от «угадывания» требований***

- ***Выполнимое требование*** - что можно сделать; осуществимо, возможна.
- ***Реализуйте требование в пределах известных ограничений и возможностей, определяемых системным контекстом.***
- ***Аналитик должен сотрудничать с разработчиками требований или персоналом отдела маркетинга на протяжении всего процесса сбора информации.***
- ***Обеспечить фактическую проверку требований***

- ***Что можно и что нельзя делать-***
- ***Что можно сделать только с чрезмерными затратами***
- ***Что можно сделать только с другими***

- ***Необходимое требование***
- ***То, что нужно клиенту***
- ***Что-то, необходимое для соответствия:*** внешнему требованию, внешнему интерфейсу или стандарту.
- ***Необходимое требование должно быть прослеживаемым до его источника (прослеживаемость)*** — источник, санкционировавший спецификацию этого требования.
- ***Неотслеживаемые требования*** на самом деле ***могут не быть необходимыми***

- ***Приоритетное требование***
- ***Определите приоритет развертывания выпуска определенных модулей И.Т. продукта исходя из:***
  - Значимости требование, предлагаемая клиенту
  - относительной стоимости реализации
  - Относительный технический риск, связанный с внедрением
- ***Приоритеты Назначаются клиентом*** (заинтересованной стороной)
  - Приоритет, который дает больше контроля руководителю проекта
  - Новые требования, добавленные во время разработки,
  - Урезание (уменьшение) бюджета,
  - Выходит за рамки расписания,
  - уход некоторых членов команды

- ***Пример уровней приоритета требований***
- ***Высокий приоритет*** - должен появиться в следующей версии продукта
- ***Средний приоритет*** — требуется, но его можно отложить для более позднего выпуска.
- ***Низкий приоритет*** — хорошо иметь это требование, но от него можно отказаться, если у вас недостаточно времени или ресурсов.

- ***Недвусмысленное требование*** – (неточное, двусмысленное выражение или утверждение)

Недвусмысленность это когда:

- ***Читатель требования иметь возможность определить его только в одной интерпретации.***
- ***Несколько читателей пришли к одной и той же интерпретации требования***
- ***В Требование не должны быть субъективных слов*** (легко, просто, быстро, эффективно и т.
- ***Требование Написано простым и понятным языком*** пользовательского домена.

- ***Ambiguitatea poate fi Dezvăluită prin:***

- Inspecții formale ale specificațiilor cerințelor

- Scrierea cazuri de testare conform cerinței

- Creare a scenariilor de utilizare care arată comportamentul așteptat

- ***Неоднозначность может быть выявлена***:
- при:

- Проведения Формальных проверок спецификаций требований
- Написание тест-кейсов согласно требованию
- Создание сценариев использования, которые показывают ожидаемое поведение

- ***Требование Поддающееся проверке***
- ***Определите, правильно ли требование реализовано в продукте***:

- Разработайте тест
- Проведите осмотр или демонстрацию

- ***Требование является верифицируемым, если оно***: ***последовательное, выполнимое и однозначное***

- ***Полное требование***

- Никакие требования или необходимая информация не должны отсутствовать

- Трудно ввести недостающие требования по ходу разработки.

- ***Пути достижения цельности***

- ***Примените метод вариантов использования*** - сосредоточьтесь на пользовательских задачах, а не на системных функциях во время сбора информации.

- ***Модели графического анализа*** – представляют различные взгляды на требования

- ***Сообщите об отсутствующей информации о требованиях***

- ***TBD («подлежит определению»)*** — решить до разработки продукта.

- Пример: «Продукт должен выдавать сообщения о состоянии через регулярные промежутки времени не реже, чем каждые 60 секунд».

- ***Неполный - отсутствует информация о - "сообщениях о состоянии"? как они должны отображаться для пользователя?***

- ***Непротиворечивость требований (SRS)*** (качество аксиоматической системы, заключающееся в том, что она не содержит какой-либо формулы одновременно с ее отрицанием).

- ***Различные требования не должны конфликтовать***
- ***Несоответствия между требованиями должны быть устранены до начала разработки***
- ***Убедитесь, что изменение требований не приводит к несоответствиям***

- ***Изменяемость требование***

- Требование должно легко пересматриваться (изменяться)

- При этом:

- ***Храните историю*** изменений по каждому требованию

- ***Дайте Уникальную метку*** для каждого требования (журнал изменений)

- ***Сгруппируйте связанные требования*** вместе

- ***Создайте оглавление***, указатель и список перекрестных ссылок

- ***Прослеживаемостиь Требования***

Свяжите каждое требование с источником требования

- Системные требования более высокого уровня
- Вариант использования
- Запрос, озвученный клиентом (из документации встречи)

Свяжите каждое требование с его разработчиком

- - Кто его разработал
- - Кто написал исходный код
- - Кто сделал тестовые случаи

Сделайте Уникальную маркировку каждого требования

Выразите каждое требование в структурированной и уточненной форме

- ***Инструкции по разработке требований***
- - Делайте предложения и абзацы короткими,
- - Используйте голосовые записи,
- - Соблюдайте правильную грамматику, орфографию и пунктуацию,
- - Используйте термины последовательно и определяйте их в глоссарии или словаре данных,
- - Прочитайте требования с точки зрения разработчика (чтобы проверить, правильно ли они определены),
- Используйте правильный уровень детализации,
- - ***Избегайте длинных описательных абзацев, содержащих несколько требований,***
- Пишите индивидуально проверяемые требования,
- Избегайте избыточного формулирования требований
- - Облегчает ведение документации

- Bertrand Meyer предложил Другой подход

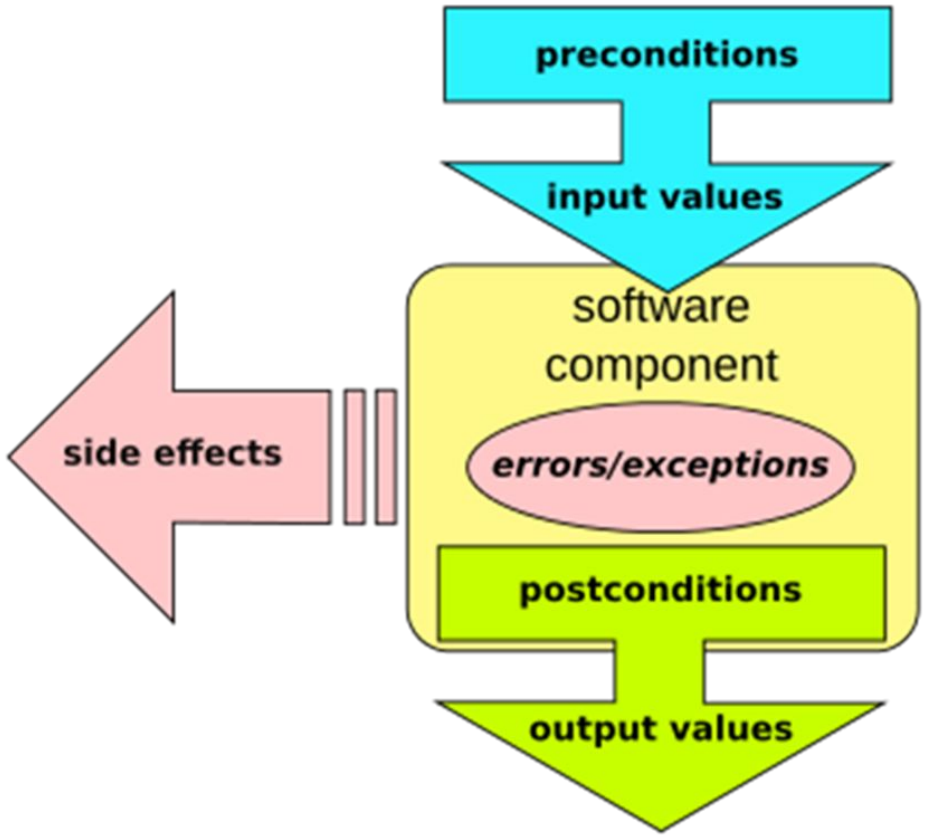
is a French academic, author, and consultant in the field of computer languages

- **Бертран Мейер — Раделяет факторы качества программного обеспечения на: внутренние и внешние**
- ***Внутренние факторы качества могут быть восприняты только ИТ - специалистами***

Внешние факторы качества являются релевантными, поскольку они воспринимаются пользователем.

Однако ***внешние факторы качества зависят*** от ***внутренних факторов***

- Бертран Мейер — автор подхода Проектирование по контракту (DbC), также известное как программирование по контракту, подход к проектированию программного обеспечения.
- Это требует от разработчиков программного обеспечения определения формальных, точных и проверяемых спецификаций интерфейса для программных компонентов, которые расширяют обычное определение абстрактных типов данных с предусловиями, постусловиями и инвариантами.
- Эти спецификации называются «контрактами» в соответствии с концептуальной метафорой условий и обязательств деловых контрактов.



- **Дизайн по контракту** (**DBC**), также известный как **контракт программирование** , **программирование по контракту** и **дизайн по-контракту программирования** , является подход к разработке программного обеспечения .
- Он предусматривает , что разработчики программного обеспечение должны определять формальные , точные и проверяемые спецификации интерфейса для программных компонентов , которые расширяют обычное определение абстрактных типов данных с предпосылками , постусловиями и инвариантами . Эти спецификации называются «контрактами» в соответствии с концептуальной метафорой условий и обязательств деловых контрактов.
- Подход DbC предполагает, что все *клиентские компоненты,* которые вызывают операцию на *серверном компоненте,* будут соответствовать предварительным условиям, указанным как требуемые для этой операции.
- Если это предположение считается слишком рискованным (например, в многоканальных или распределенных вычислениях), используется обратный подход , означающий, что *серверный компонент* проверяет выполнение всех соответствующих предварительных условий (до или во время обработки запроса *клиентского компонента*). и, если нет, отвечает подходящим сообщением об ошибке.

- Центральная идея DbC - это метафора того, как элементы программной системы взаимодействуют друг с другом на основе взаимных *обязательств* и *выгод* . Метафора пришла из деловой жизни, где «клиент» и «поставщик» договариваются о «контракте», который определяет, например, следующее:
- Поставщик должен предоставить определенный товар (обязательство) и вправе ожидать, что клиент уплатил его вознаграждение (выгоду).
- Клиент должен оплатить комиссию (обязательство) и имеет право получить товар (выгоду).
- Обе стороны должны соблюдать определенные обязательства, такие как законы и постановления, применимые ко всем контрактам.
- Аналогичным образом , если метод из класса в объектно-ориентированном программировании обеспечивает определенную функциональность, она может:
- Ожидайте, что определенное условие будет гарантировано при входе любым клиентским модулем, который его вызывает: предварительное условие метода - обязательство для клиента и выгода для поставщика (сам метод), поскольку он освобождает его от необходимости обрабатывать случаи вне рамок предварительное условие.
- Гарантировать определенное свойство при выходе: постусловие метода - обязательство для поставщика и, очевидно, выгода (основное преимущество вызова метода) для клиента.
- Поддерживать определенное свойство, предполагаемое при входе и гарантированное при выходе: инвариант класса .
-

- Контрактное программирование (программирование по контракту, Design by Contracts, DbC) – подход к созданию программ высокого качества
  - Автор подхода – проф. Бертран Мейер
  - Впервые введено в языке программирования Eiffel 1985 год
- Основная идея – объединить программный код и спецификации
- Спецификации (контракты) встраиваются в программу
  В основе лежит логика Хоара
- Тройка Хоара: ***{P}C{Q}***
  - P и Q – утверждения
  - C – часть программы

- Входное условие называется предусловием ***(precondition).***
- Выходное условие – постусловием ***(postcondition)***
- Дополнительно обеспечивается поддержка инвариантов.
  Инвариант ***(invariant)*** – условие, которое не должно нарушаться из-за выполнения метода,
- т.е. должно гарантироваться выполнение инварианта до и после выполнения метода
- Во время выполнения инвариант может быть временно нарушен!
-

- В терминах контрактного программирования метод (или функция) обязуется выполнить контракт:
- Если на вход поступили данные удовлетворяющие входному условию контракта, то метод гарантирует соблюдение выходного условия
- Если входные данные НЕ удовлетворяют первому условию, то ничего не гарантируется
- При этом соблюдаются некоторые обобщенные условия

- Пример. Снятие денег в банкомате
  - **Balance** – сумма на счете клиента
  - **Cash** – сумма, которую клиент хочет снять
  - **Amount** – денежный запас банкомата
  - **Предусловия:**
  - **Cash** > 0;
  - **Cash** ≤ 10000;
  - **Постусловия:**
  - **Balance** = old(Balance) – Cash;
  - **Amount** = old(Amount) – Cash;
  - **Инвариант:**
  - **Amount** ≥ 0;
  - **Balance** ≥ - 1000;
-

Языковая поддержка

- Ада 2012
- Чао
- Clojure
- Кобра
- D [10]
- Дафни
- Эйфелева
- Крепость
- Котлин
- Меркурий
- Oxygene (ранее Chrome и Delphi Prism [11])
- Ракетка (включая контракты более высокого порядка и подчеркивание того, что в нарушениях контрактов должна быть виновная сторона и должно быть сделано это с точным объяснением)
- Sather
- Скала
- СПАРК (через статический анализ программ на Аде)
- Спецификация #
- Вала
- VDM
-

	Фактор	Определения
1.	Обоснованность	Способность программного продукта выполнять функции, определенные Техническим заданием и спецификацией
2.	Надежность	Способность программного обеспечения функционировать в нештатных условиях
3.	Расширяемость	Легкость, с которой программный продукт поддается модификации или расширению дополнительных функций
4.	Возможность повторного использования	Возможность повторного использования программного продукта полностью или частично в новом приложении
5.	Совместимость	легкость, с которой одна система может быть объединена с другой
6.	Эффективность	Возможность использование альтернативных ресурсов (процессоры, внутренняя и внешняя память, протоколы связи...)
7.	Портативность	Легкость , с которой программный продукт может быть перенесен в различные программно-аппаратные среды
8.	Проверяемость	Простота подготовки процедур тестированя и валидации испытаний, процедуры обнаружения ошибок
9.	Итегральность	Способность программного обеспечения защищать код и данные от несанкционированного доступа
10.	Простота использования	Простота понимания, использования, подготовки данных, возврата к нормальной ситуации

- ***Валидность*** – способность программного продукта точно выполнять свои функции, определенные Технисеским Заданием в спецификациях.

- ***Расширяемость*** — это возможность легко модифицировать программное обеспечение, добавляя в систему новые функции.
- Этот фактор качества трудно измерить, поскольку он часто связан с уровнем языка программирования (например, легче модифицировать на языке XXXX, чем на Ассемблере, легче модифицировать на языке YYYYY, чем на языке XXXX).

- ***Возможность повторного использования*** — возможность использования программного обеспечения в новых приложениях.
- Концепция не нова, уже давно делают общие подпрограммы для обработки данных, или программные скелеты для автоматизации различных функций.
- Свойство наследования, введенное в объектно-ориентированном подходе, позволяло методам, написанным для одного объекта, наследоваться другим.

- ***Совместимость***, которую не следует путать с переносимостью,
- — это простота, с которой один программный продукт можно комбинировать с другими.
- В некоторых ситуациях необходимо приобрести приложения со стороны (которые станут подсистемами в создаваемом приложении) и которые должны работать вместе с уже существующими приложениями

- ***Эффективность*** - это оптимальное использование ресурсов вычислительного комплекса.
- С момента развития Информатики мощность машин удваивается каждые 18 месяцев, но этого все еще недостаточно, чтобы можно было игнорировать проблемы ***Эффективности***.
- ***С одной стороны***, создаются все более сложные приложения, требующие высокопроизводительных компьютеров,
- ***с другой стороны***, мощности машины расходуется на комфорт пользователя: ***на Macintosh или ПК с Windows более 80% мощность используется для обработки графического интерфейса***, и только 20% остается для использования приложением

- ***Портативность*** — это легкость, с которой программный продукт может быть перенесен на другое оборудование или операционную систему.
- Это критерий, который выдвигают поставщики инструментов, поскольку он позволяет уменьшить зависимость пользователя от оборудования.

- ***Верифицируемость*** - это средство, с помощью которого подготавливаются процедуры ***Верификации*** и ***валидации*** системы.
- ***Это важный критерий, который следует немного отразить: можно ли легко проверить, работает система или нет, можно ли построить тесты, гарантирующие, что часть программного обеспечения или вся система работают?***
- Когда мы сталкиваемся с проблемой тестирования программного обеспечения, которое неизвестно и не проводилось ранее, непросто определить, является ли результат, выдаваемый системой, правильным или нет.

- ***Целостность*** — *это способность системы защитить свой код и данные от несанкционированного использования;*
- это скорее общая характеристика среды, в которой должна работать система, чем непосредственная функция, обеспечиваемая программным инструментом.
- Но инструмент должен со временем компенсировать недостатки среды разработки.

- ***Простота использования*** - представляет собой все, что связано с эргономикой приложения, простотой его использования.
- Для пользователя неодинаково, если он интуитивно и быстро понимает, как работает система, или ему нужно больше времени, чтобы привести ее в действие.
- Это также не то же самое, если он получает четкое сообщение об ошибке или ему приходится обращаться к руководству, чтобы понять его значение.
- Эргономика станет очень важным критерием функциональности самой системы, потому что, без сомнения, считается «компетенцией» системы, если она правильно выполняет свою задачу (критерий валидности).

După - Karl Wiegers and Joy Beatty

- **Correct - (corectă)**
- **Feasible - (fezabilă)**
- **Necessary – (necesară)**
- ***Prioritized - (proritizată)***
- ***Unambiguous–(neambiguă)***
- ***Verifiable -(verificabilă)***
- ***Completă***
- ***Consistentă***
- ***Modificabilă***
- ***Trasabilă***
-
-

După Bertrand Meyer.

- ***Validitate***
- ***Fiabilitate***
- ***Extensibilitate***
- ***Reutilizare***
- ***Compatibilitate,***
- ***Eficacitatea***
- ***Portabilitatea***
- ***Verificabilitate***
- ***Integritate***
- ***Facilitate de utilizare***
-

- ***Выводы*** - ***На этапе анализа находим ответ на вопрос «ЧТО»*** мы хотим построить - Он относится к предметной области

На этом этапе выполняются три основных действия:

- ***Анализ требований***
- ***Спецификация требований***
- ***Проверка требований***
- ***Документ «Спецификация требований к системе и программному обеспечению» (SRS) представляет собой набор требований, согласованных между заказчиком и командой разработчиков.***

- **Стоимость подхода к качеству ИТ-продукта**

- В целом ***понятие «качество»*** (***происходит от латинского «qualitas», «qualis», что означает «способ бытия»)*** - можно присвоить несколько значений в зависимости от контекста, в котором он употребляется.
- При этом оно может означать:

- ***способ бытия вещи или индивидуума, его признаки и недостатки;***
- ***набор условий жизни в данный момент - "качество жизни"***
- ***коммерческое качество продукта;***
- ***уровень «превосходства» услуги.***

Наряду с ростом и диверсификацией спроса на продукцию и развитием промышленного производства понятие «качество» продукции постоянно развивалось.

приходящие сегодня, в условиях использования информационных технологий для контроля качества, к новым значениям через понятия ***«направленное качество, гарантированное качество, общее качество и управление качеством*** и т. д.».

- ***Основными факторами, способствовавшими повышению значимости качества продукции и услуг в современной экономике,*** являются:

-Обострение конкуренции,

-повышение требований клиентов и компаний, а также

-увеличение сложности изделий и процессов их изготовления.

- ***Определение***
- Качество представляет собой выражение степени общественной полезности продукта, степени, в которой по всем его характеристикам:
  - технико-функциональная,психосенсорный и др.
  - экономические параметры, удовлетворяет потребность, для которой продукт был создан,
  - соблюдая ограничения, налагаемые общими интересами общества, в отношении социально-экономической эффективности, а также защиты природной и социальной среды.

- ***Согласно стандарту ISO 8402,***
- качество представляет собой совокупность характеристик сущности, продукта, деятельности, процесса, организации, человека -который дает возможность удовлетворять выраженные или подразумеваемые требования.
- ***Продуктом, рассматриваемым в соответствии со стандартом ISO 8402, может быть***: ***любое материальное благо (результат производства), либо нематериальное (ПО - программы, информация, данные) и/или услуга (банковское дело, транспорт и т.п.)***, возникающее в результате некоторых действий и/или процессов.

- ***Качество означает:***

- Возможность применения или использования;

- удовлетворение требований клиента;

- соответствие документации или требованиям бенефициара.

Качество выражается не одной характеристикой, а через совокупность характеристик, не являющихся самостоятельными, существующими только по отношению к потребностям клиентов.

- Согласно ISO 9000:2001, качество – это степень, в которой набор внутренних характеристик соответствует требованиям.
- ***С точки зрения ISO 9000 продукты делятся на 4 категории***, а именно:

- ***услуги*** (например: транспорт),
- ***программное обеспечение*** (компьютерная программа, словарь),
- ***оборудование*** (например, механическая часть двигателя, компонент компьютера) и
- ***обработанные материалы*** (например, смазка).

- В соответствии с определенными принципами ***качество*** может быть рассмотрена из некоторые ***особые аспекты***:

- ***качество в продуктовой ориентации*** – качество считается измеримой характеристикой;
- ***качество в технологической ориентации*** – учитывается точка зрения производителя,
- согласно которой качественная продукция – это та, которая удовлетворяет ***заданным требованиям***; в то времея у пользователя может быть своя точка зрения;

- ***качество в стоимостной ориентации*** (имплицитно продажная цена) – продукт считается качественным, если он предлагает определенные характеристики при приемлемом уровне цены. Такую ориентацию принимает значительный сегмент потребителей (в Германии 17%);

- **качество в ориентации на пользователя** – качество определяется как способность продукта быть пригодным для использования (пригодность для использования). Он представляет собой соответствие требованиям бенефициара в отношении функциональности, безопасности эксплуатации, цены, срока поставки, стоимости использования, совместимости с окружающей средой и т. д.

- ***Требование определяется как представление потребности или ожидания, которое заявлено, как правило, неявно или обязательно.***
- ***Заявленное требование*** – это требование, изложенное в документе.
- ***Под несоответствием понимается невыполнение требования***, при этом
- ***Дефект*** – ***это невыполнение требования***, связанного с предполагаемым или указанным использованием.

- *Как продукт - ИТ-продукт отличается от других промышленных объектов.*
- *Некоторые очевидные различияю:*

- *Продукт имеет высокие постоянные затраты и несколько более низкие переменные затраты.*
- *продукт Не изнашивается, но требует ухода.*
- *Легче добавить дополнительную ценность будет В будущем (например, по сравнению с аппаратным обеспечением).*
- *ИТ-продукт более сложен*
- *Он более неосязаем и менее заметен, потому что он нефизичен.*

- ***Определение качества программного продукта***
- ***Что измеряется?!,***
- ***Что совершенствуется?!***
- ***«Качество» может означать разные вещи для разных людей.***
- ***Понятие и словарный запас понятия качества неуловимы.***
- ***Значения различаются в зависимости от обстоятельств и восприятия.***

Стоимость продукта ИТ — Модель айсберга

- Многие издержки низкого качества ИТ-продуктов скрыты, и их трудно определить с помощью формальных систем измерения.
- Модель айсберга (изображение ниже) очень часто используется для иллюстрации этой концепции:
- Очевидна лишь малая часть издержек низкого качества программного обеспечения — те, которые происходят выше ватерлинии.
- Но существует огромный потенциал для снижения затрат ниже ватерлинии.
- Выявление и улучшение этих затрат значительно снизит эксплуатационные расходы на ИТ-продукт для компании/организации.


Costs Usually Visible
Customer problem reports
Customer service calls
Lawsuits/warrantee claims
QA & test department costs
Service outages
Costs Usually Not Visible
Finding & fixing internal problems/defects
Cancelled and troubled projects
Unaccounted overtime (crisis mode)
Waste and rework
Successful cyber attacks
Staffing problems (e.g.turnover)
Poor teamwork
Lack of good planning
Dubious project value/ROI
Excessive systems costs
Lost market opportunities
Lack of good practices & quality standards
Understanding complex code
Technical debt
Poor quality data

- ***Видимые затраты*** (обычно):
- ***отчеты о проблемах клиентов***
- ***звонки в службу поддержки клиентов***
- ***судебные иски / гарантийные претензии***
- ***Стоимость отдела поддержки и тестирования***
- ***перебои в работе службы***

Costuri invizibile (de obicei)

- ***găsirea și remedierea problemelor și/sau defectelor interne***
- ***proiecte cu probleme și anulate***
- timp nedeclarat (modul de criză)
- deșeuri și refaceri
- atacuri cibernetice de success
- probleme de personal (de exemplu, fluiditae specialiști)
- activitate slabă în echipă
- lipsa unei bune planificări
- valoare / rentabilitate dubioasă a proiectului
- Costuri excesive a sistemului
- oportunități de piață pierdute
- Lipsa unei practice bună
- intelegerea codului complex
- lipsuri tehnice
- date de calitate slabă

Невидимые затраты (обычно)

- ***поиск и устранение внутренних проблем и/или дефектов***
- ***проблемные и отмененные проектыне***
- недекларированное время (кризисный режим)
- ***брак и переделка***
- ***успешные кибератаки***
- ***кадровые вопросы*** (текучесть специалистов)
- ***плохая командная работа***
- ***отсутствие хорошего планирования***
- Сомнительное отношение ***ценность /прибыльность*** проекта
- Чрезмерные системные затраты
- Упущенные рыночные возможности
- ***Отсутствие хорошей практики***
- Понимание сложного кода
- Технические недостатки
- ***Данные низкого качества***

- Качество программного обеспечения более точно определяется как сочетание следующих 4 аспектов:

1. Соответствие требованиям

Требования четко сформулированы и продукт должен им соответствовать.

Любое отклонение от требований считается дефектом

Товар хорошего качества содержит меньше дефектов

2. Пригодность к использованию/назначению

В соответствии с ожиданиями пользователя: отвечает потребностям пользователя

Продукт хорошего качества обеспечивает большее удовлетворение пользователей

- ***3. Соблюдение стандартов***

Во многих отраслях и организациях должны соблюдаться определенные внешние и внутренние стандарты.

Продукт хорошего качества соответствует требуемым стандартам качества/процесса (например, ISO 25000, уровень CMMI).

- ***4. Основные аспекты,*** которые включают

Структурное качество (например, сложность)

Эстетическое качество (например, внешний вид)

•

- Если бы существовала простая мера «хорошего» программного продукта, мы бы все использовали его и все просили бы его.
- На практике в качестве индикаторов используются несколько метрик, обычно в комбинации.
- Цикломатическая сложность, глубина наследования, степень связанности классов и некоторые другие структурные показатели являются показателями кода.
- Количество усилий, необходимых для понимания того, что выполняет часть кода, является еще одним хорошим показателем.

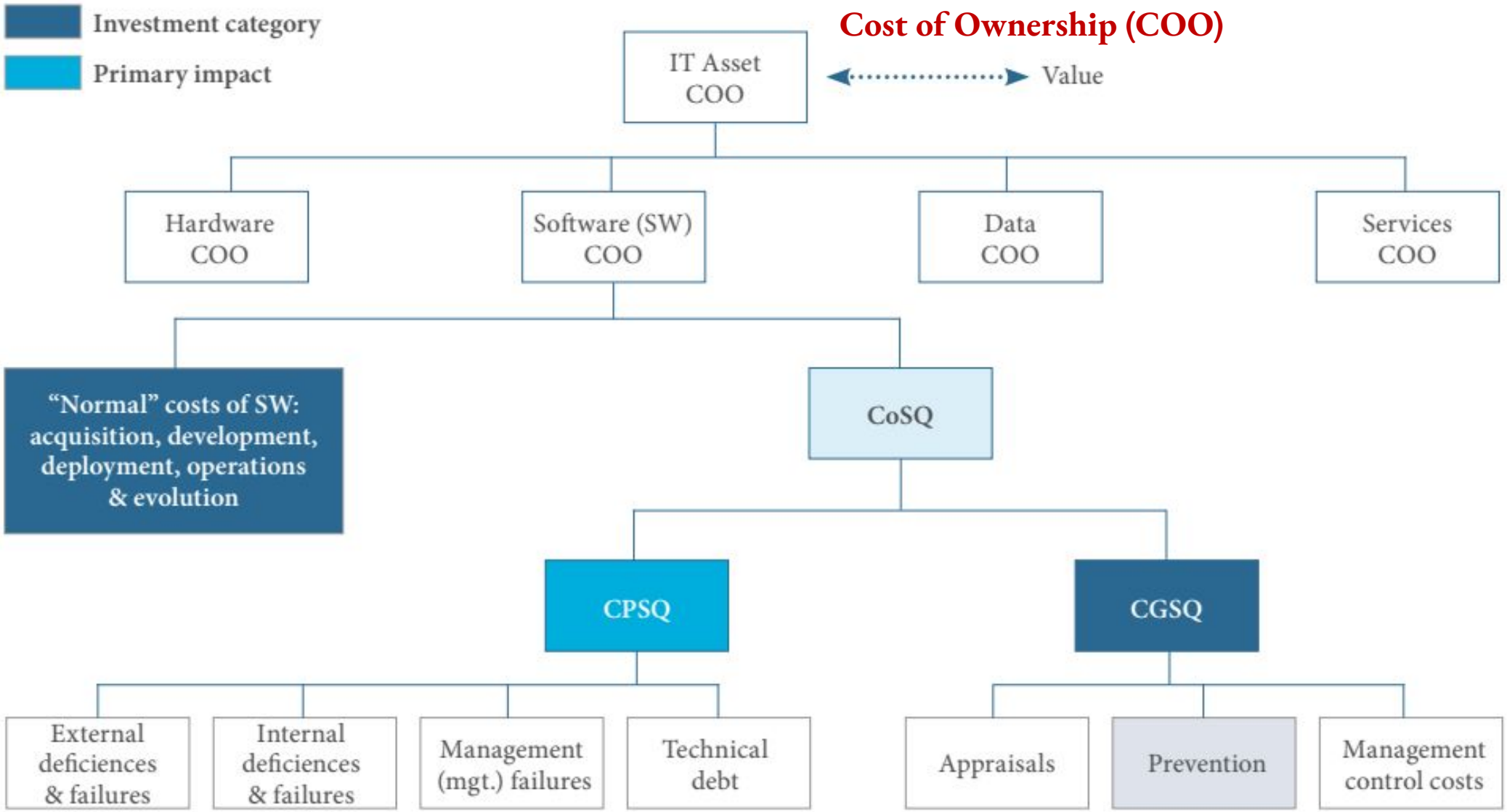
- Низкое качество не является неизбежным атрибутом Программного Продукта.
- Возникает по известным причинам.
- Качество можно предсказать и контролировать, но только в том случае, если её причины поняты и устранены.
- ***Основными причинами низкого качества Программного Продукта являются:***

- ***Недостаток знаний предметной области*** (что приводит к слабым требованиям)
- ***Отсутствие технологических знаний*** (что приводит к неуверенности в качестве)-
- ***Нереалистичные графики*** (из-за плохой практической оценки)-
- ***Плохо разработанное программное обеспечение*** (в результате незрелых, недисциплинированных и использования менее квалифицированных программистов)
- ***Неправильная практика закупок***
- ***Нарушения связи и координации в командах***
- **Отсутствие данных о состоянии качества программного обеспечения**

- ***Модель стоимости качества.***
- ***Модель стоимости качества — это методология, которая позволяет организации определить, в какой степени ее ресурсы используются для действий, которые непосредственно влияют на качество продуктов или услуг организации и являются результатом сбоев и недостатков.***
- ***Наличие такой информации позволяет организации определить потенциальную экономию, которая может быть достигнута за счет внедрения улучшений процессов и продуктов.***

- *Первоначальная модель затрат на качество была разделена на четыре категории:*

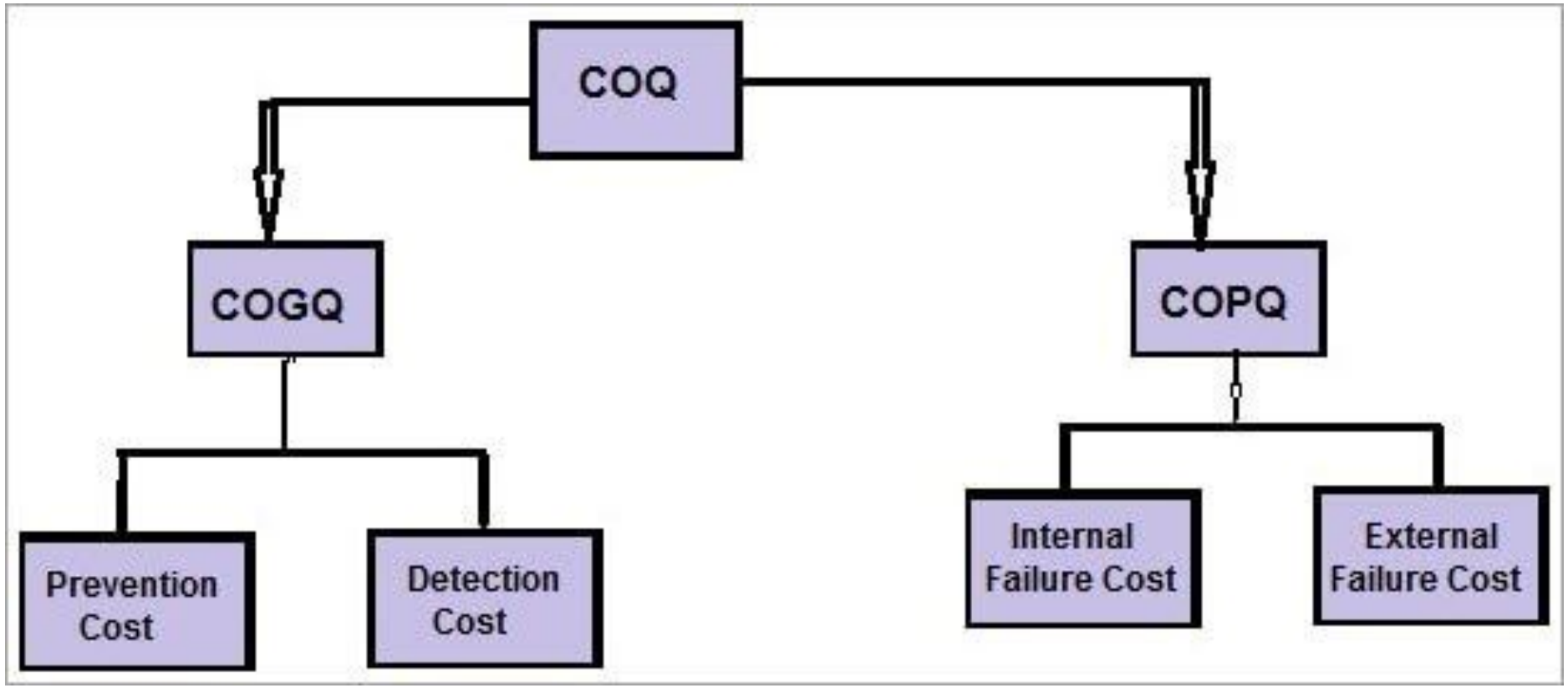
1. *Цена внешнего отказа*
2. *Цена внутреннего отказа*
3. *Стоимость обследования (оценки)*
4. *Стоимость профилактики*


Investment category
Primary impact
Cost of Ownership (COO)
IT Asset COO
Value
Hardware COO
Software (SW) COO
Data COO
Services COO
"Normal" costs of SW: acquisition, development, deployment, operations & evolution
CoSQ
CPSQ
CGSQ
External deficiences & failures
Internal deficiences & failures
Management (mgt.) failures
Technical debt
Appraisals
Prevention
Management control costs

Cost Of Quality Formula

COQ = Cost of control + Cost of failure of control or

COQ = Cost of Good Software Quality + Cost of Poor Software Quality



- Cost of Good Quality (COGQ) = Prevention costs + Detection costs
- Cost of Poor Quality (COPQ) = Internal failure costs + External failure costs
- While it is most desirable to check for all quality points, every check is a cost overhead, so a balance needs to strike, and clarity of the quality coverage expected should be very clear from the beginning. This coverage should be shared with all stakeholders.
- Quality management is about striking a good balance and prioritizing the high-risk quality dimensions achievable within a desirable/optimal budget.
- Quality is really crucial, but it need not be attained at the micro-level of all features. The trick is to achieve the **right balance between quality and cost** and achieve the best that is required by the client at an optimal cost.
-

Quality Is Multi-Dimensional

- **Functional and Non-Functional Quality**

1. **Fitness for Purpose:** Software performs all tasks as specified in the SRS document.
2. **Infrastructure Support:** Software supports the environment specified and is scalable **E.g.,** the software works in all browsers.
3. **Cost:** The cost heads associated with the development and delivery of software is within the budget.
4. **Process:** The development and review process is well established and as per standards.
5. **Management:** The review and monitoring system has all the checkpoints to assess and ensure Product quality.
6. **Reporting:** The reporting and documents are timely, informative, and actionable.
7. **Functional Suitability / Appropriateness:** Appropriateness, Accuracy, and Compliance **E.g.** If the user wants confirmation of the ticket booked to be mailed instead of printing. Is this option available?

-

1. **Performance Efficiency (Time Behavior):** Response Time, Resource Utilization, and Compliance **E.g.** What is the response time for the searches available? Is it quick enough?
2. **Compatibility (Inter-Operability):** Co-existence, replaceability, interoperability, compliance **E.g.,** Is the Website usable from a mobile/iPad?
3. **Usability:** Ease of use, help available, learning ease, compliance **E.g.** Is the site easy to navigate and use?
4. **Reliability:** Fault tolerance, availability, recoverability, compliance **E.g.,** Will the application display the bookings when there is a power shutdown while printing the ticket?
5. **Security:** Confidentiality, integrity, accountability, authenticity, compliance **E.g.** Are personal data shared on the site secure?
6. **Maintainability:** Component reusability, ease of change, ease of replication**,** testability**,** stability, compliance, **E.g.** How easy is it to enhance the site with new features?
7. **Portability:** Ease of installation, portable, adaptable, compliance **E.g.** Will the site behave the same way if the operating system is changed/updated?

•

- După cum putem vedea în imaginea de mai sus, Costul calității software-ului (CoSQ) este o parte a costului de proprietate COO (Cost of Ownership (COO) pentru componenta software a unui sistem, care este o parte a COO-ului total pentru un activ IT care conține software.
- ***Pe măsură ce analizăm îmbunătățirile proceselor și produselor, cuantificarea costurilor de „calitate" pentru organizație este definită drept Costul calității (COQ).***
- De ce să cuantificăm datele de calitate? COQ clasifică aceste costuri, astfel încât organizația să poată vedea cum trecerea de la o asigurare a calității (control și corectare) la un accent pe prevenire ajută la reducerea costurilor eșecurilor și deficiențelor.

- ***Categorii de CPSQ***
- Costul de calitate neproductivă (COPQ):
- Acestea sunt costurile asociate cu furnizarea de produse, sisteme sau servicii de calitate slabă.
- Există patru categorii de costuri :

- costuri de eșec interne (de ex. Costuri asociate cu defecte constatate înainte ca clientul să primească produsul sau serviciul),
- costuri de eșec externe (de ex. Costuri asociate cu defecte constatate după ce clientul primește produsul sau serviciul),
- costuri de evaluare (costuri suportate) pentru a determina gradul de conformitate cu cerințele și standardele de calitate) și
- Cost eșecu de gestionare (costuri suportate de directori și care se ocupă mai jos de ramificațiile software-urilor de calitate slabă).

- *Internal Failure and Defciency Costs*

  Tese costs are associated with system failures and defciencies discovered before the system leaves the development organization and is deployed into the operational environment. Tese defciencies occur when a system fails to meet a certain requirement, resulting in waste or rework. Te defciencies could be in the work
  products of development, the development process, and/or components if they fail to meet quality standards and requirements. Te largest category of costs here are the professional effort to fnd and fx all of the defects. Te impact of cancelled and delayed projects are also included here. Unfortunately, very few organizations track this category prior to the commencement of testing. Included are:
  - **Waste**—performance of unnecessary work or holding of work products as a result of errors, poor organization, or communication, cancelled and challenged projects
  - **Scrap**—defective product or material that cannot be repaired, used, or sold
  - **Rework or rectifcation**—correction of defective material or errors, agile refactoring
  - **Failure analysis**—activity required to establish the causes of internal failure

- External Failure and Defciency Costs
- Tese costs occur when products or services that fail to reach quality standards are not detected until afer transfer into operation or to the customer. External failure/defciency costs are incurred during customer use and can include defective products, warranty charges, customer complaints, rejections, recalls, returns, patches and repairs. While external costs are the most apparent, these costs sometimes can be difcult to quantify. Terefore, businesses fail to include them in the overall quality costs because failures such as poor installation and usage problems are not always reported by the customer. A large category of costs here are massive failures, and latent defects in the sofware when delivered. Te largest category of costs here are professional effort to replicate, fnd and fx all of the felded defects and re-appraisals to verify fxes. Loss of sales, tarnished reputation, legal/litigation and excessive customer complaint handling, are large costs in this category. Included are: wasted marketing costs, brand damage, and technical support team effort.

- ***Technical Debt***
- Technical debt in sofware is a relatively new concept. Te term was coined by Ward Cunningham to describe the obligation that a sofware organization incurs when it chooses a design or construction approach that's expedient in the short term but that increases complexity and is costlier in the long term.
- Tere are two basic types of technical debt: intentional and ***unintentional.*** One of the important implications of technical debt is that it must be serviced, i.e., once you incur a debt there will be interest charges. A good example of this is future refactoring that needs to be done.

- Technical debt is measurable.
- For example, one organization we've heard about maintains a debt list within its defect tracking system. Each time a debt is incurred, the tasks needed to pay off that debt are entered into the tracking system along with an estimated effort and schedule. Te debt backlog is then tracked, and any unresolved debt more than 90 days old is treated as critical. Since this is a fairly new concept, there are still questions raised about what kinds of flaws should or shouldn't be classifed as Technical Debt. Tere are others who defne it more precisely in order to quantify the level of structural quality problems in the operational system. Structural quality metrics measures how well a system is designed and constructed with respect to best practices.

- *Management Failures*

  Tese are the non-technical costs incurred by an organization who suffers from poorquality sofware management practices at the executive level and below. Tis includes:

  • Unplanned costs for professional and other resources, resulting from underestimation of the resources in the planning stage.

  • Damages paid to customers as compensation for late project completion, a result of the unrealistic schedule in a project's proposal/plan.

  • Damages paid to customers as compensation for late completion of the project, a result of management's failure to recruit qualifed team members.

  • Damages to other projects planned to be performed by the same teams involved in the delayed projects. Te domino effect may induce considerable hidden failure costs.

  • Excessive management crisis mode behaviors, like lots of meetings to solve urgent problems.

  • Hidden external failure costs, that is, reduction of sales as a result of damaged reputation, increased investments in sales promotion underpricing of tender bidding to counter the effects of signifcant past delayed completion of projects due to managerial failures in appraisal and/or progress control tasks

- Categorii de CGSQ (Costul unei bune calități)
- Costul unei bune calități a software-ului este la fel de variabil ca organizațiile reprezentate.
- Unele grupuri investesc mult în managementul și planificarea proactivă a calității, în timp ce altele se ocupă de sisteme de patch-uri și programe reactive care vizează rezolvarea problemelor după apariția acestora.
- Costurile de bună calitate sunt în general împărțite în:

- ***costuri de control al managementului,***
- ***costuri de prevenire și***
- ***costuri de evaluare.***

•*Costuri de evaluare*

- Costurile de evaluare sunt cele asociate acțiunilor menite să găsească probleme de calitate cu: măsurarea, evaluarea, inspecția, testarea și auditarea sistemelor și a produselor de lucru pentru a se asigura că respectă standardele de calitate și cerințele de performanță.
- Investiția în resurse pentru a identifica și pentru a diagnostica în cele din urmă calitatea slabă ajută o organizație să își atingă obiectivele strategice și să crească valoarea sistemului și satisfacția generală a clienților.
- Cele mai mari părți de cost aici sunt de obicei teste și QA.
- Sunt incluse: V&V, audituri de calitate, inspecții, evaluări inter-pares, evaluări ale furnizorilor etc.

•*Costuri de prevenire*

- Costurile de prevenire sunt suportate pentru a preveni sau evita problemele de calitate.
- Acestecheltuieli mențin costurile eșecului / deficienței produsului la un nivel minim și pot contribui la reducerea costurilor de evaluare.
- Eliminarea defectelor înainte de începerea implementării reduce costurile de calitate și poate ajuta companiile să crească profiturile.
- Costurile de prevenire includ planificarea proceselor, revizuirea și analiza auditurilor de calitate și instruirea angajaților pentru a preveni eșecurile viitoare. Părțile majore sunt ***managementul proactiv al calității***, ***planificarea calității***, ***instruire și programe de îmbunătățire***.

- *Management Control Costs*
  Management can perform several activities to prevent or reduce the costs that result from the types of failure particular to its functions: contract reviews, planning, goal establishment, and progress review and control of the sofware project. Tis includes:
  - Costs of carrying out contract reviews
  - Establishing quality goals, objectives, gating/release criteria and quality standards
  - Costs of preparing project plans, including quality management plans
  - Costs of periodic updating of project and quality plans
  - Costs of performing regular progress review and control
  - Costs of performing regular progress control of external participants' contributions to projects

  A detailed chart of accounts for our CoSQ model is beyond the scope of this report.

  Te author can be contacted for examples of such.

- Understanding Cost of Poor Sofware Quality in your organization is the frst step toward gaining executive buy into quality-led operations. Tis is fundamental to agile, DevOps as well as Proactive and Predictive Quality Management. With a CPSQ number in hand, you have the basis for a business case to invest smartly in quality. Determining CPSQ may sound daunting, but in fact, it's very achievable and
simply requires some tried-and-true methods along with a cross-functional team to get the brainstorming on paper. Tis author has developed a survey instrument to help organizations get started. You can gain an understanding of the true impact of problems, mistakes, bugs, defects, security gaps, and general sloppiness.