Introduction to Software Quality Assurance

What is Software ?

According to the IEEE

Software is:

Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system

(IEEE Std 829-2008)

What is Software Quality ?

Software quality is:

> *"degree to which the software product satisfies stated and implied needs when used under specified conditions" [ISO/IEC 25010]*

What is Software Quality ?

- Five major views of software quality
  - Transcendental view
  - User view
  - Manufacturing view
  - Product view
  - Value-based view

What is Software Quality ?

- Transcendental view – Quality
  - is hard to define in abstract terms
  - can be recognized if present
  - some intangible properties that *delight* users
  - ideal that may never be able to be completely attained

What is Software Quality ?

- User view – Quality
  - is fitness for purpose
  - depends on how product meet its users' needs for tasks
  - can be measured based on *reliability*, *usability*

What is Software Quality ?

- Manufacturing view – Quality
  - conformance of product to process standards
  - depends on how well the product was constructed
  - potential to avoid costs associated with rework during development and after delivery
  - measured using *defect counts*, *rework costs*

What is Software Quality ?

- Product view – Quality
  - tied to inherent product characteristics
  - one measures and control internal product properties to improve external product behavior
  - examples of internal product properties measures:
    - size, ratio of code/comments, code complexity, ...

What is Software Quality ?

- Value-based view – Quality
  - depends on how much customers are willing to pay for
  - trades-off cost and quality

What is Software Quality ?

- *Conformance to requirements*
- Lack of defects
  - Low *defect rate* (# of defects/size unit)
- High *reliability* (number of failures per *n* hours of operation)
  - Measured as Mean Time To Failure (MTTF)
    probability of failure-free operation in a specified time
- User/Customer satisfaction

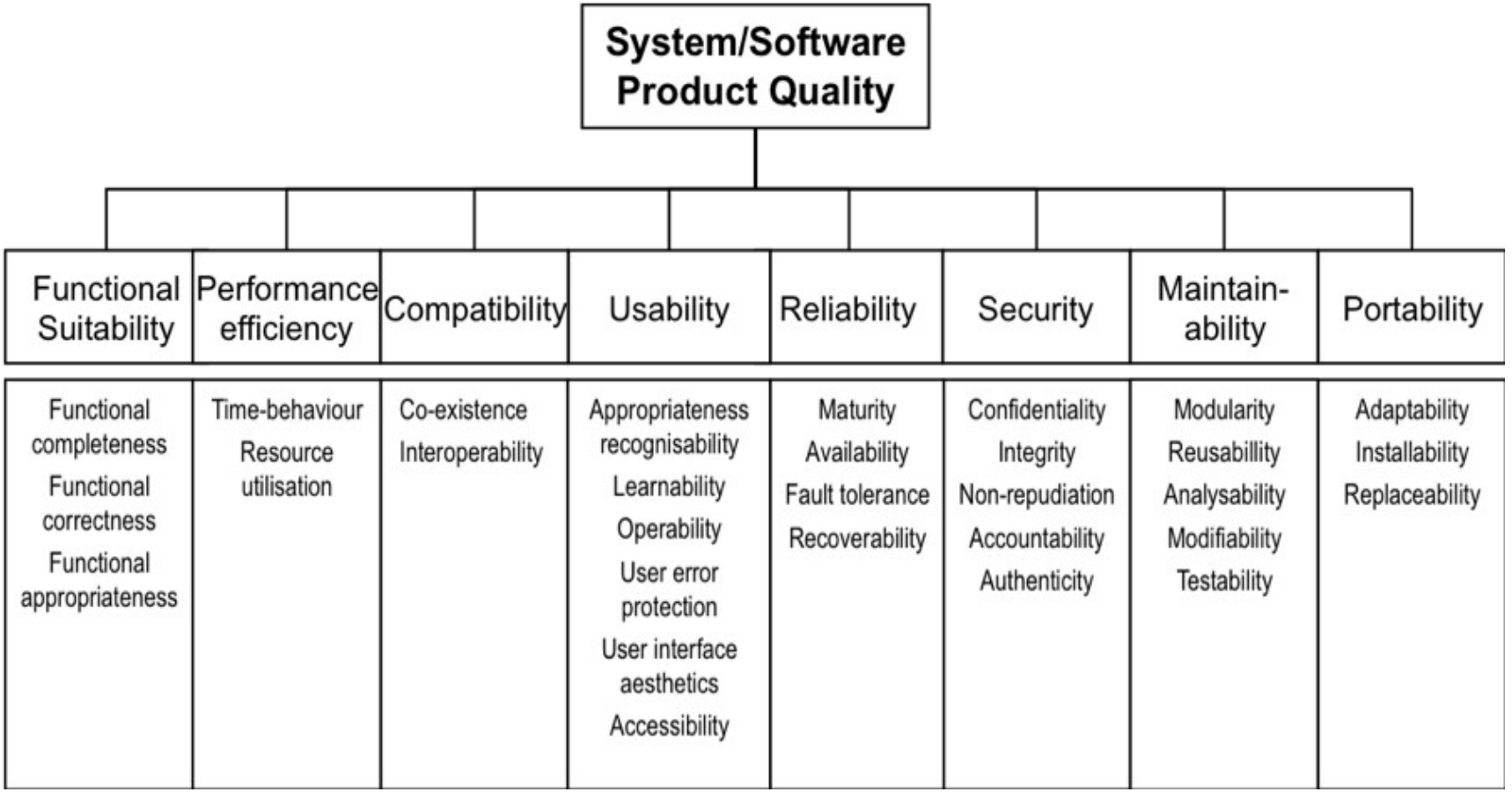
Software Quality Factors

- Characteristics of the overall software quality
  - related to purpose and usage of the product
  - examples: correctness, reliability, efficiency, testability, maintainability, reusability
  - each relevant to specific stakeholders

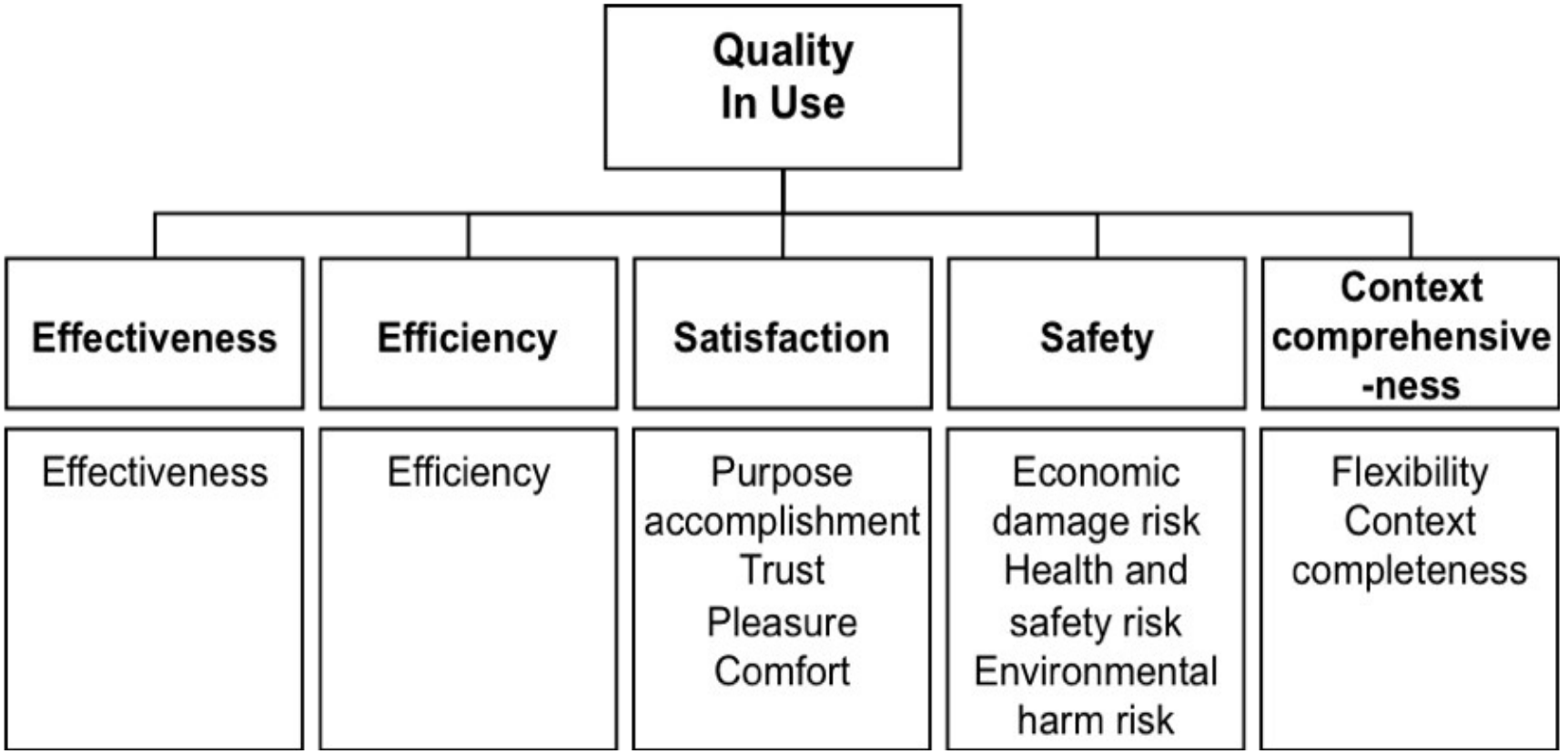
Software Quality Model

- Objectives
  - define quality characteristics,
  - their attributes,
  - features,
  - sub-characteristics, and
  - Measurements
- Example ISO/IEC 25010
  - software product quality model
  - system quality in use model

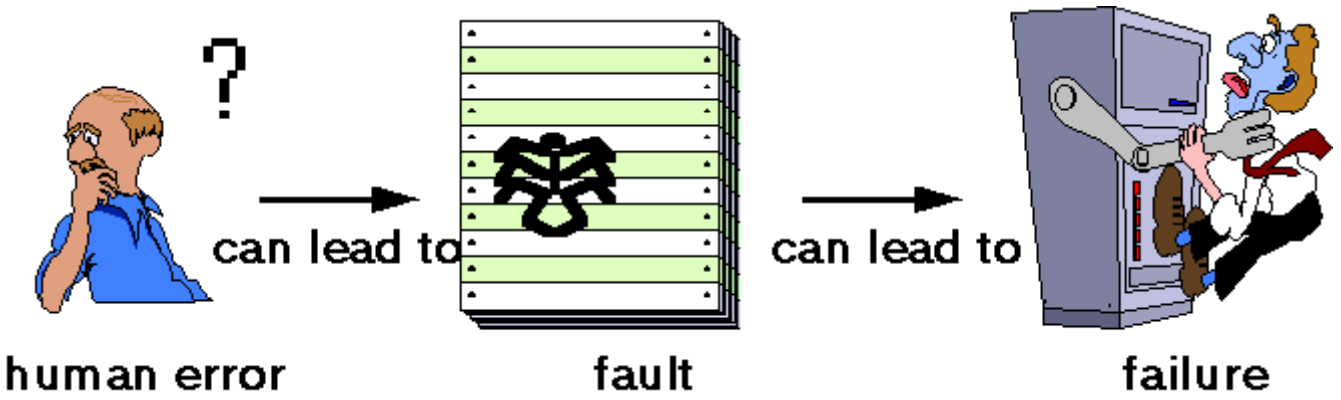
Software Quality Model ISO/IEC 25010



Software Quality Model ISO/IEC 25010



Errors, Faults and Failures

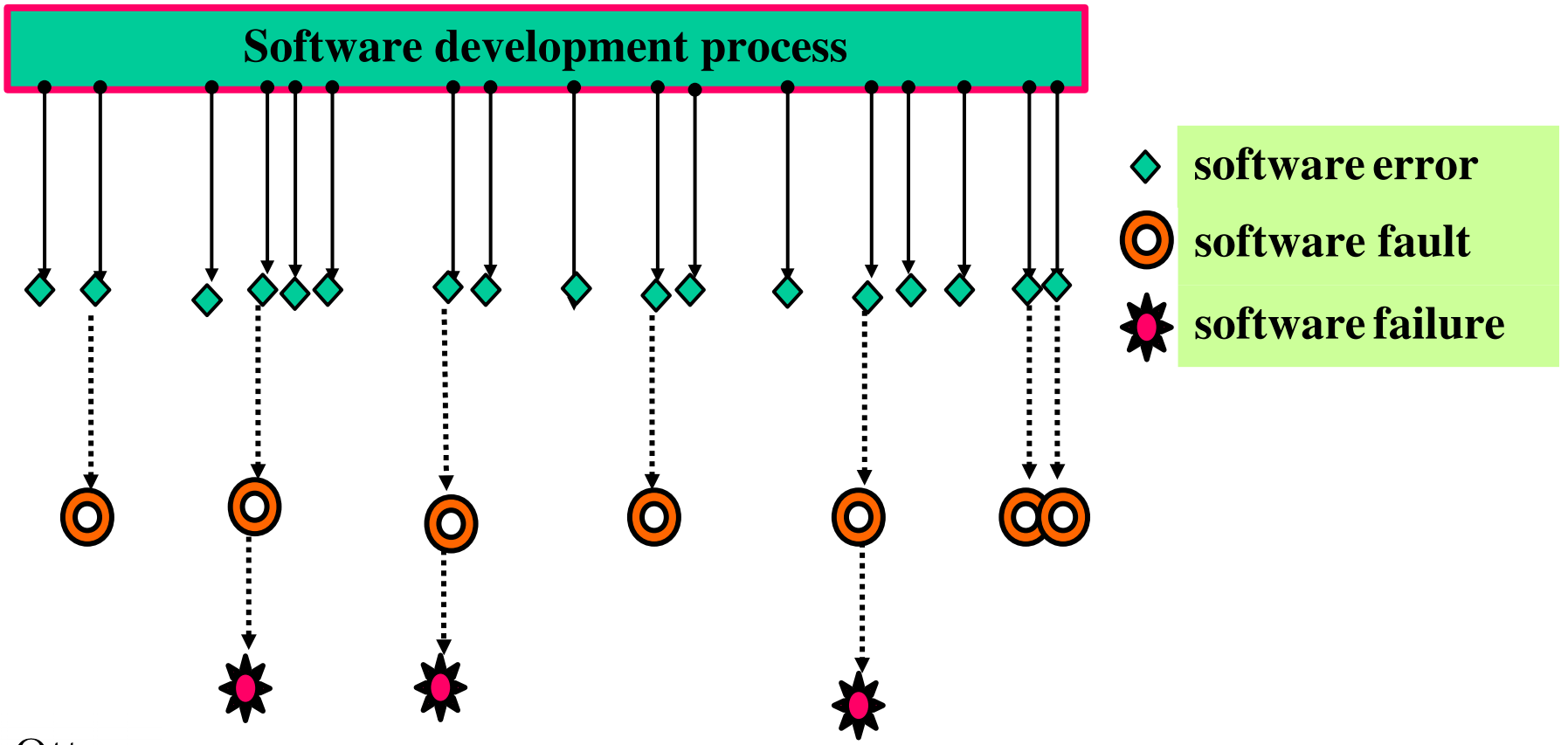


Errors, Faults and Failures

- **Error**, **Fault**, **Failure** – key concepts associated to software correctness
  - Failure: inability of a system or component to perform its required functions within specified performance requirements
  - Fault (Defect/Bug): an incorrect step, process, or data definition in a software artifact
  - Error: a human action that produces an incorrect result

Errors, Faults and Failures

- *Bug/defect/fault* consequence of a *human error/mistake*
  - results in non-conformance to requirements
  - manifests as *failure* in running software



Examples of Errors

1. Faulty requirements definition
2. Client-developer communication failures
3. Deliberate deviations from software requirements
4. Design errors
5. Coding mistakes
6. Non-compliance with documentation and coding instructions
7. Inadequate testing
8. Inadequate documentation

Sample failure situations

1. Software doesn't do something that the specification says it should do
2. Software does something that the specification says it shouldn't do
3. Software does something that the specification doesn't mention
4. Software doesn't do something that the specification doesn't mention but should
5. Software is difficult to understand, hard to use, slow, ...
6. Software is viewed by its stakeholders as just plain not right

Importance of Software Quality

- Software is a major component of computer systems (more than 80% of the cost) – used for
  - communication (e.g. phone system, email system)
  - health monitoring,
  - transportation (e.g. automobile, aeronautics),
  - economic exchanges (e.g. e-commerce),
  - entertainment,
  - etc.
- Software defects are extremely costly in term of
  - money
  - reputation
  - loss of life

Importance of Software Quality

- Several historic disasters attributed to software
  - On June 3, 1980, the North American Aerospace Defense Command (NORAD) reported that the U.S. was under missile attack.
  - Therac-25 radio therapy system defects resulted in several deaths
  - 1988 shooting down of Airbus 320 by the USS Vincennes - cryptic and misleading output displayed by tracking software
  - 1991 patriot missile failure - inaccurate calculation of time due to computer arithmetic errors
  - London Ambulance Service Computer Aided Dispatch System – several deaths
  - 9 hour breakdown of AT&T's long-distance telephone network - caused by an untested code patch
  - ….

Importance of Software Quality

- Ariane 5 crash June 4, 1996
  - maiden flight of the European Ariane 5 launcher crashed about 40 seconds after takeoff
  - lost was about half a billion dollars
  - explosion was the result of a software error
    - Uncaught exception due to floating-point error: conversion from a 64-bit integer to a 16-bit signed integer applied to a larger than expected number
    - Module was re-used without proper testing from Ariane 4
      - Error was not supposed to happen with Ariane 4
      - No exception handler

Importance of Software Quality

- Mars Climate Orbiter - September 23, 1999
  - Mars Climate Orbiter, disappeared as it began to orbit Mars.
  - Cost about $US 125-million
  - Failure due to error in a transfer of information between a team in Colorado and a team in California
    - One team used English units (e.g., inches, feet and pounds) while the other used metric units for a key spacecraft operation.

Importance of Software Quality

- Mars Polar Lander - December, 1999
  - Mars Polar Lander, disappeared during landing on Mars

- Failure more likely due to unexpected setting of a single data bit.
  - defect not caught by testing
  - independent teams tested separate aspects

Importance of Software Quality

- Zune 30 LeapYear freeze

On December 31st 2008

- players began freezing at about midnight becoming totally unresponsive and practically useless

Importance of Software Quality

- Zune 30 LeapYear freeze

Official Fix

- Wait until January 1st 2009

Importance of Software Quality

- Zune 30 LeapYear freeze – source of the problem

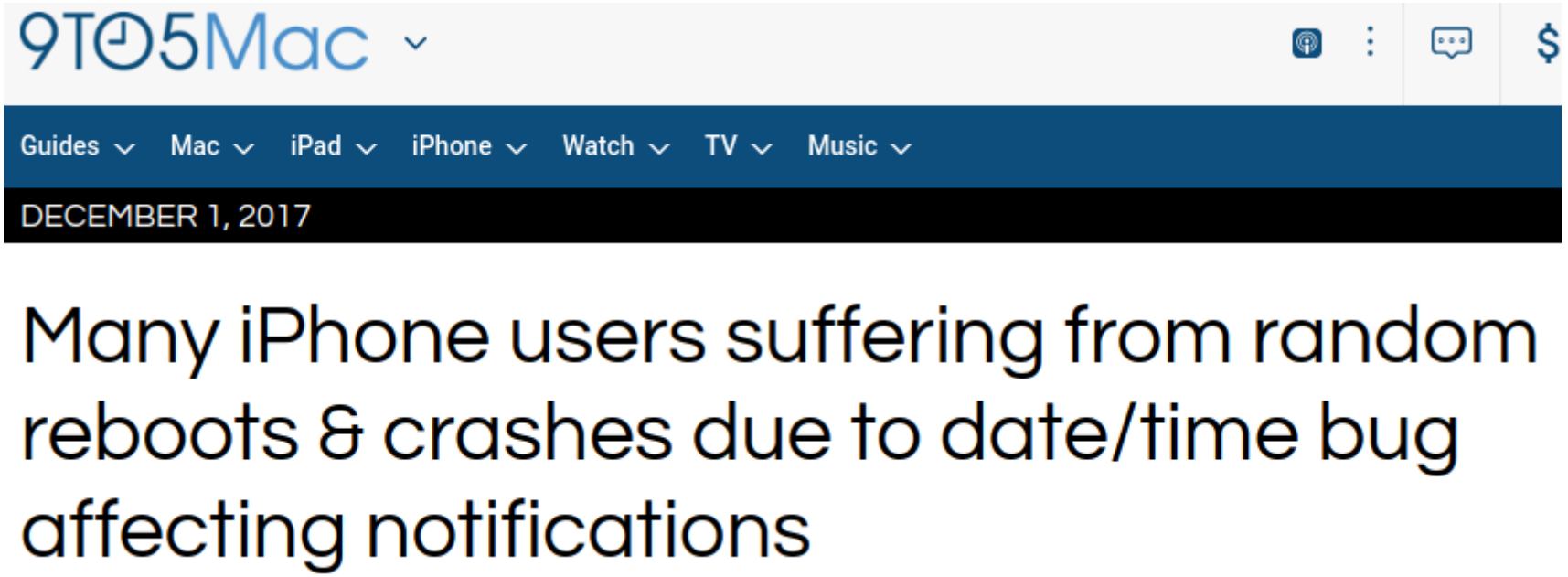
```
year = ORIGINYEAR; /* = 1980 */
while (days > 365)
{
    if (IsLeapYear(year))
    {
        if (days > 366)
        {
            days -= 366;
            year += 1;
        }
    }
    else
    {
        days -= 365;
        year += 1;
    }
}
```

Importance of Software Quality

- IPhone Alarm glitch

- Re-occurring alarm on some devices using the mobile operating system iOS 4.1 failed to properly work after daylight saving time switch
- Problem repeated on January 1st 2011 – this time for non-re-occuring alarms

Importance of Software Quality



Importance of Software Quality

- Malware
  - Viruses, worms, trojans – e.g.
    - Blaster worm ($US 525 millions)
    - Sobig.F ($US 500 millions – 1billions)
    - SMB Worm (used in Sony hack)
  - Bot – connects unsuspecting hosts to botnets
    - for DDOS, Spam, bitcoin mining, …
  - Ransomware – encrypts software resources and ask for ransom (e.g. Petya, WannaCry, ...)

Importance of Software Quality

- Malware - Exploit well known software vulnerabilities
  - *Software developers do not devote enough effort to applying lessons learned about the causes of vulnerabilities.*
  - *Same types of vulnerabilities continue to be seen in newer versions of products that were in earlier versions (CERT).*

Importance of Software Quality

- Apple “goto fail” SSL bug

About the security content of iOS 7.0.6

This document describes the security content of iOS 7.0.6.

iOS 7.0.6

- Data Security

Available for: iPhone 4 and later, iPod touch (5th generation), iPad 2 and later

Impact: An attacker with a privileged network position may capture or modify data in sessions protected by SSL/TLS

Description: Secure Transport failed to validate the authenticity of the connection. This issue was addressed by restoring missing validation steps.

CVE-ID

CVE-2014-1266

Importance of Software Quality

- Apple “goto fail” SSL bug – source of defect

```
. . .
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;  /* MISTAKE! THIS LINE SHOULD NOT BE HERE */
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
err = sslRawVerify(...);
. . .
```

Importance of Software Quality

- Heartbleed bug – another SSL vulnerability

- Found in OpenSSL library
  - allows reading "bleeding" information in memory (e.g. secret keys)
  - Affected 500000 HTTPS websites (17.5%)

Importance of Software Quality

- Heartbleed bug – cause
  - improper bound check before **memcpy**()
  - non-sanitized user supplied input used as length parameter

Heartbeat sent to victim

SSLv3 record:

Length
4 bytes

HeartbeatMessage:

Type	**Length**	**Payload data**	
TLS1_HB_REQUEST	65535 bytes	1 byte	

Victim's response

SSLv3 record:

Length
65538 bytes

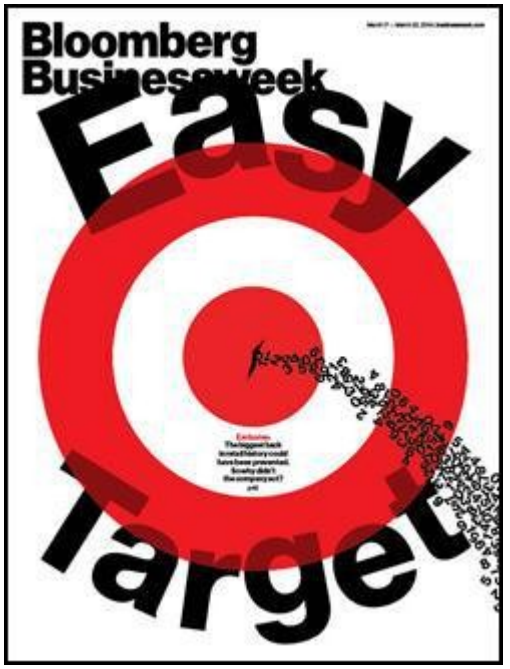
HeartbeatMessage:

Type	**Length**	**Payload data**	
TLS1_HB_RESPONSE	65535 bytes	65535 bytes	

Importance of Software Quality

Importance of Software Quality

- Internet viruses and worms
  - Used for several attacks – e.g. Target

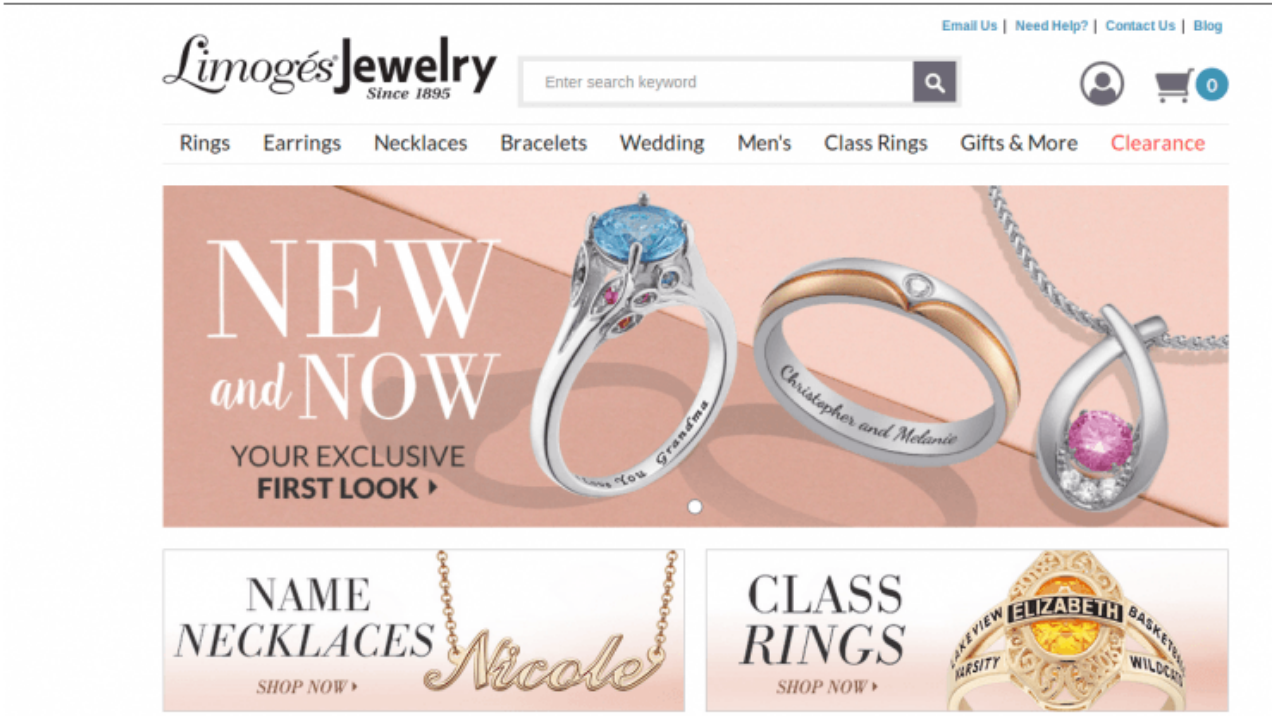


Over 40 million credit cards and 70 million personal customers data stolen

Importance of Software Quality

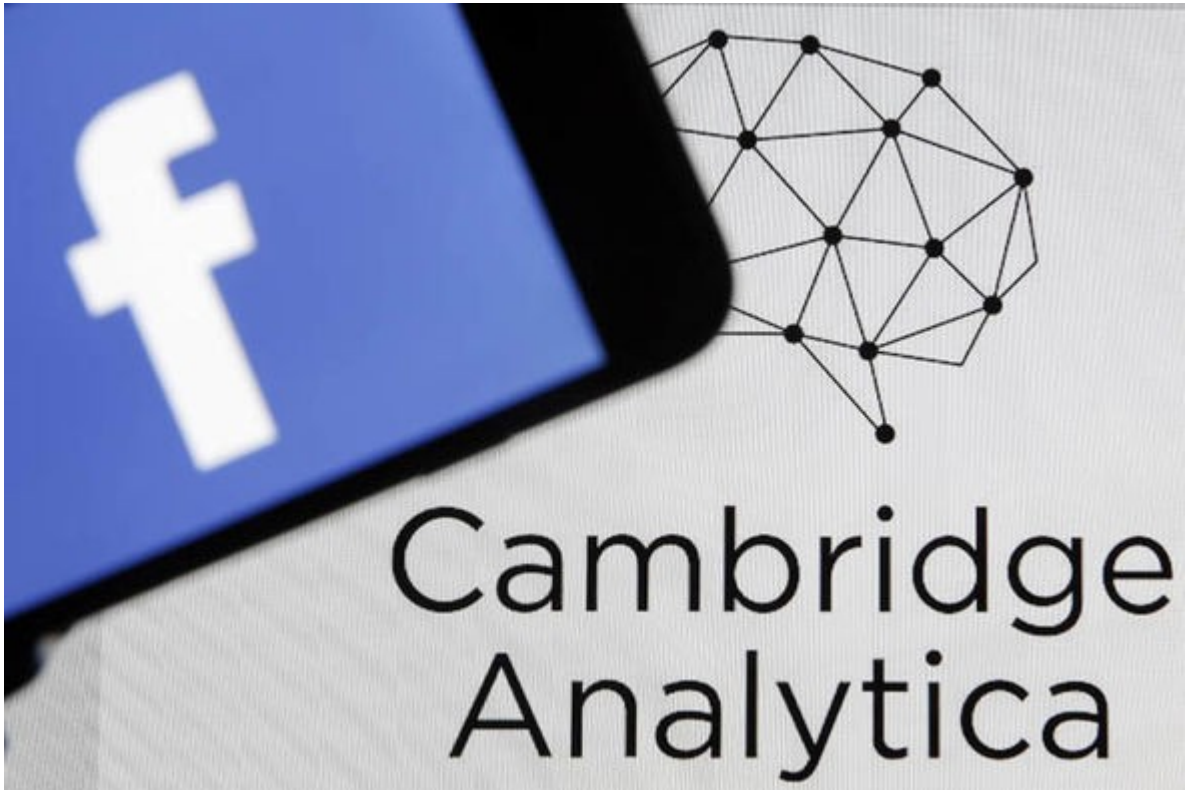
Jewelry site accidentally leaks personal details (and plaintext passwords!) of 1.3M users

by MATTHEW HUGHES — 28 days ago in SECURITY



Due to unsecured Amazon S3 storage bucket, containing a MSSQL database backup file

Importance of Software Quality



- Collection of up to 87 million user data for political influencing
  - data collected without consent from most of users
  - used to build profile for political advertising
- Several lawsuits and fines likely to cost billions to Facebook

Importance of Software Quality

- Ontario Social Assistance Management System (SAMS)
  - more than $250 million project
  - delivered with lots of flaws
    - more complex UI for operators than previous system
    - increased cost of operation
    - improper benefits computation
  - review found inadequate testing before introduction

Importance of Software Quality

- Planned total cost $310 million
  - expected to save $78 million / year in operation
- Launch resulted in multiple problems (not pay, wrong pay, ...)
  - total cost to date ~$1 billion
  - new/replacement system being considered

Economic Importance of Software Quality

- NIST report, "The Economic Impacts of Inadequate Infrastructure for Software Testing" (2002)
  - Inadequate software testing costs the US alone between $22 and $59 billion annually
  - Better approaches could cut this amount in half
- Huge losses due to web application failures
  - Financial services : $6.5 million per hour (just in USA!)
  - Credit card sales applications : $2.4 million per hour (in USA)

The Software Quality Challenge

- The uniqueness of the software product
  - High complexity
  - Invisibility of the product
  - Limited opportunities to detect defects (“bugs”)
    - only opportunity is *Product development*

The Software Quality Challenge

- The environments in which software is developed
  - Contracted
  - Subjection to customer-supplier relationship
  - Requirement for teamwork
  - Need for cooperation and coordination with other development teams
  - Need for interfaces with other software systems
  - Need to continue carrying out a project while the team changes
  - Need to continue maintaining the software system for years

Development process relation to faults

Majority of faults are introduced in earlier phases

Phase	Percentage of faults	Effort to fix faults
Requirements	56	82
Design	27	13
Code	7	1
Others	10	4

Relative cost of fixing faults

Phase in which found	Cost Ratio
Requirements	1
Design	3 – 6
Coding	10
Unit/Integration testing	15 – 40
System/Acceptance testing	30 – 70
Production	40 – 1000

Requirements are the top reason for a project success or failure

What is Software Quality Assurance?

- Set of activities aimed at ensuring that few, if any, faults remain in the software system when it is delivered to its customers or released to the market

- Process that ensures that developed software meets and complies with defined or standardized quality goals

What is Software Quality Assurance?

According to the IEEE

Software quality assurance is:

1. *A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.*
2. *A set of activities designed to evaluate the process by which the products are developed or manufactured.*
3. *the planned and systematic activities implemented within the quality system, and demonstrated as needed, to provide adequate confidence that an entity will fulfill requirements for quality*

Objectives of SQA

(1) Assuring an acceptable level of confidence that the software conform to functional technical requirements.

(2) Assuring an acceptable level of confidence that the software conform to managerial scheduling and budgetary requirements.

(3) Initiation and management of activities for the improvement and greater efficiency of software development, maintenance and SQA activities.

Three General Principles of QA

- Know what you are doing
- Know what you should be doing
- Know how to measure the difference

Three General Principles of QA

- Know what you are doing
  - understand what is being built, how it is being built and what it currently does
  - supposes a software development process with
    - management structure (milestones, scheduling)
    - reporting policies
    - tracking

Three General Principles of QA

- Know what you should be doing
  - having explicit requirements and specifications
  - supposes a software development process with
    - requirements analysis,
    - acceptance tests,
    - frequent user feedback

Three General Principles of QA

- Know how to measure the difference
  - having explicit measures comparing what is being done from what should be done
  - four complementary methods:
    - formal methods – verify mathematically specified properties
    - testing – explicit input to exercise software and check for expected output
    - inspections – human examination of requirements, design, code, ... based on checklists
    - metrics – measures a known set of properties related to quality

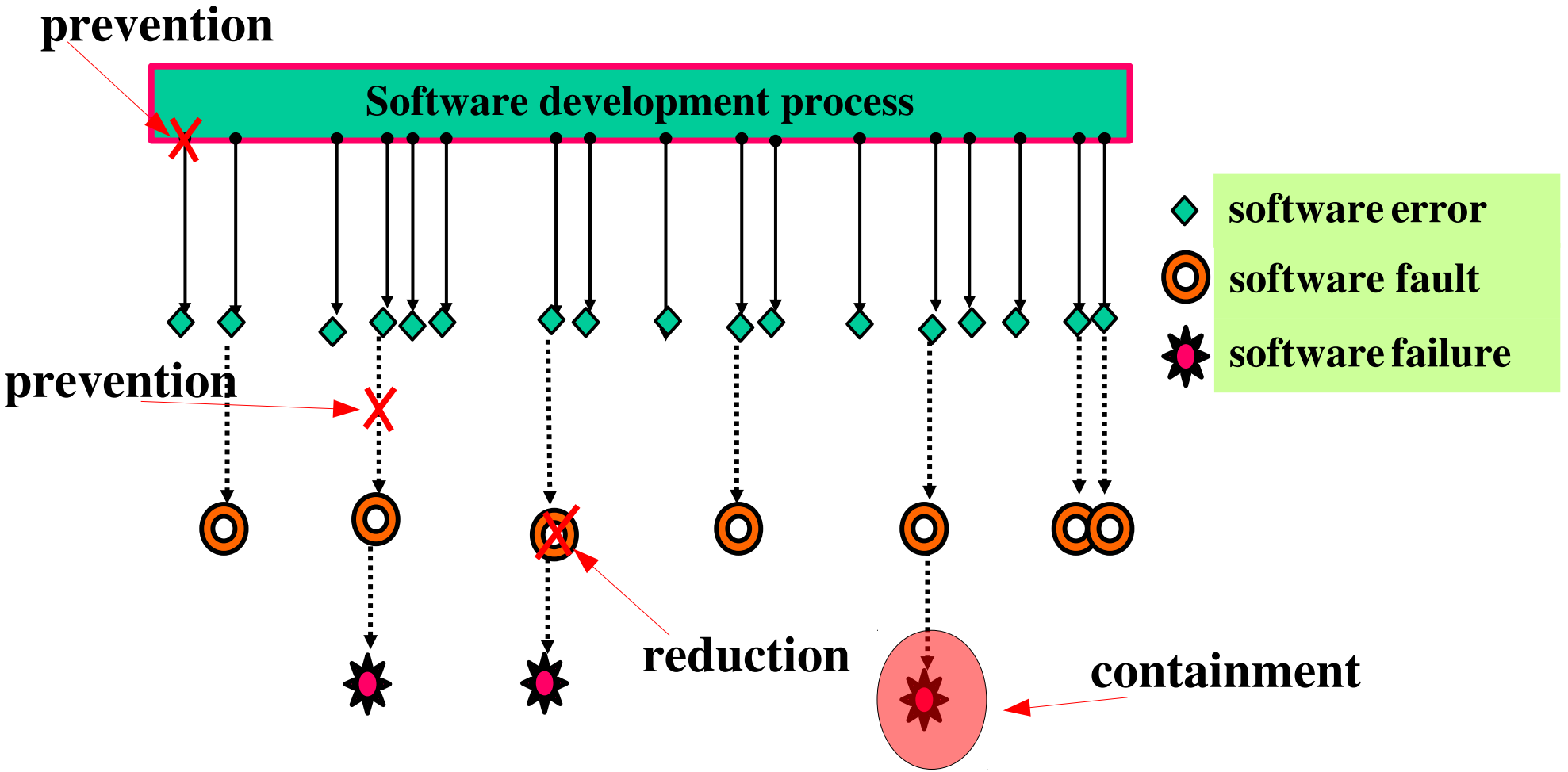
Software Quality Assurance

- SQA: Comprehensive life-cycle approach concerned with every aspect of the software product development process
- Includes
  - comprehensive set of quality objectives
  - measurable quality attributes (quality metrics) to assess progress toward the objectives
  - quantitative certification targets for all component of the software development processes.
- Takes into account:
  - customer product requirements,
  - customer quality requirements, and
  - corporate quality requirements.

Software Quality Assurance Activities

- Objective: ensure few, if any, defects remain in the software system when delivered
- Classes of QA activities
  - defect prevention
    - error blocking, error source removal
  - defects reduction
    - fault detection and removal
  - defects containment
    - failure prevention and containment

Software Quality Assurance Activities



Software Quality Assurance Activities

- Defect prevention - prevent faults from being injected into the software
  - elimination of error sources
    - e.g remove ambiguities, correct misconceptions
  - fault prevention or blocking
    - e.g. tools, technologies, standards to block errors
    - prevents errors from causing defects

Software Quality Assurance Activities

- Sample defect prevention techniques
  - education & training – removes misconceptions
  - formal methods – removes ambiguities
  - process conformance, standard enforcement – prevent non-conformance faults

Software Quality Assurance Activities

- Defect reduction – detects and removes faults before system is delivered
  - examples: inspection, prototype simulation, static analysis, testing

Software Quality Assurance Activities

- Defect containment – minimizes the effects of defects (failures) – e.g.
  - contains local failures so they do not propagate (fault tolerance)
  - limits damage caused by failures (safety assurance)

Software Quality Assurance
Verification & Validation

Verification:

"*Are we building the product right ?*"

The software should conform to its specification.

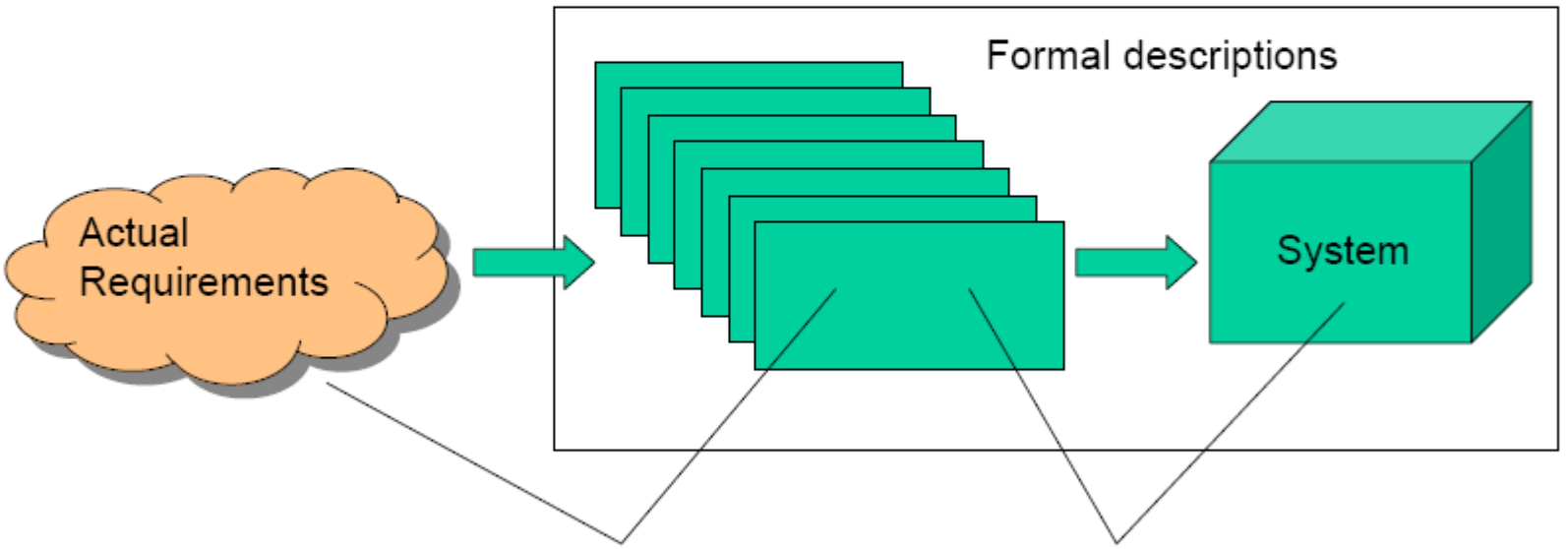
Validation:

"*Are we building the right product ?*"

The software should do what the user really requires.

Validation vs. Verification

ABC, "Validation, Verification and Testing of Computer Software." *ACM Computing Surveys*, June 1982.



Validation
Includes usability testing, user feedback

Verification
Includes testing, inspections, static analysis

Software Quality Assurance Verification & Validation

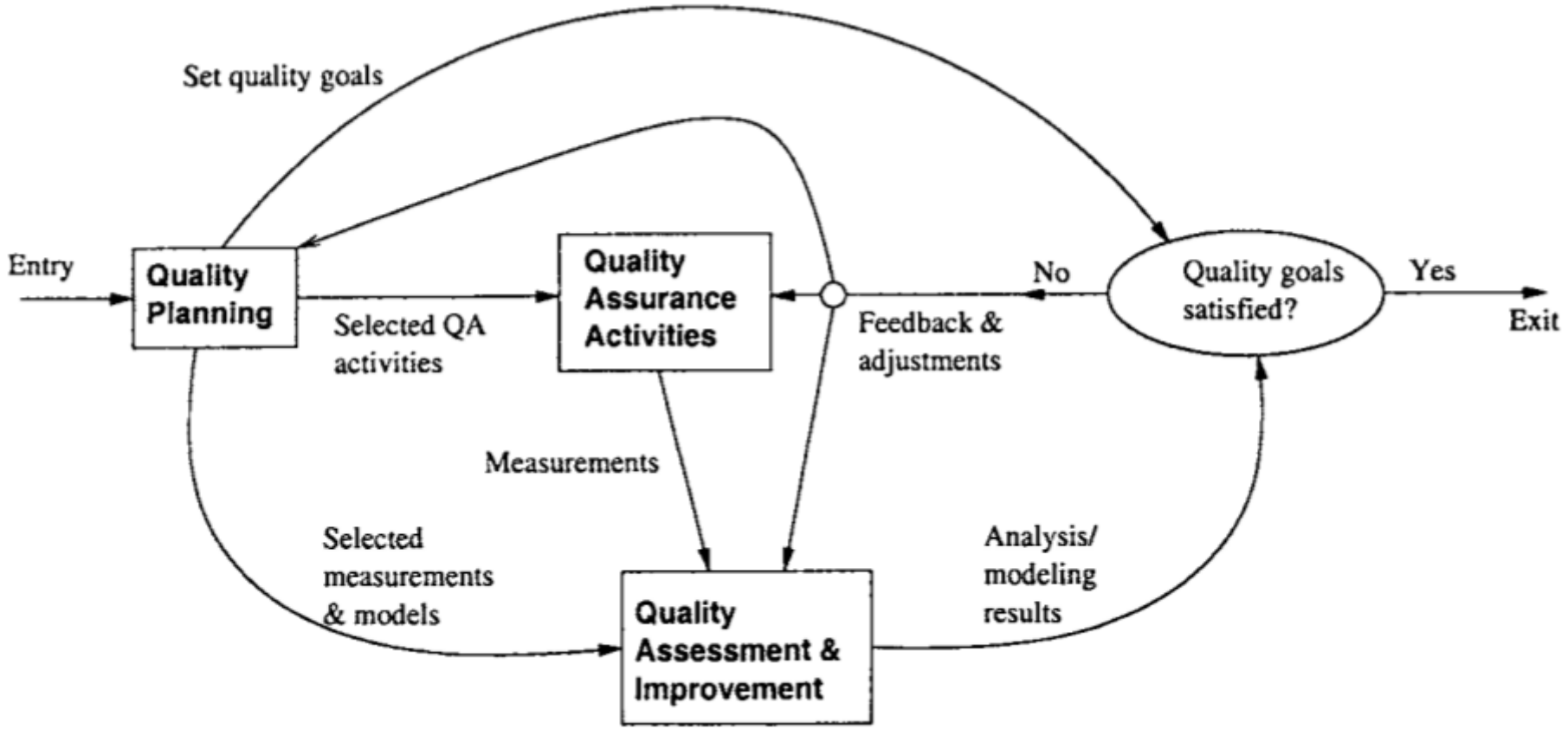
- Verification
  - The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. IEEE Std 1012-2004
- Validation
  - The process of providing evidence that the software and its associated products satisfy system requirements allocated to software at the end of each life cycle activity, solve the right problem, and satisfy intended use and user needs. IEEE Std 1012-2004

Software Quality Engineering

- Software Quality Engineering (SQE)
  - encompasses SQA with related activities (planning, monitoring, improvement)
  - engineering approach to manages quality expectations
    - setting of quality goals
    - measuring of performance toward these goals
    - process analysis and improvement

Software Quality Engineering

- Software Quality Engineering (SQE)



Software Quality Engineering

- Quality Planning
  - sets specific quality goals (based on customer's expectations, economical/time constraints)
  - forms overall quality strategy (selects QA activities, selects quality measurements)

Software Quality Engineering

- Quality Assessment and Improvement
  - feedback from measurements collected during QA activities
  - may prompt changes/adjustment to quality goals, QA strategy, development process, ...