LOG8371 : Ingénierie de la qualité en logiciel Assurance Qualité en Logiciel

Hiver 2024

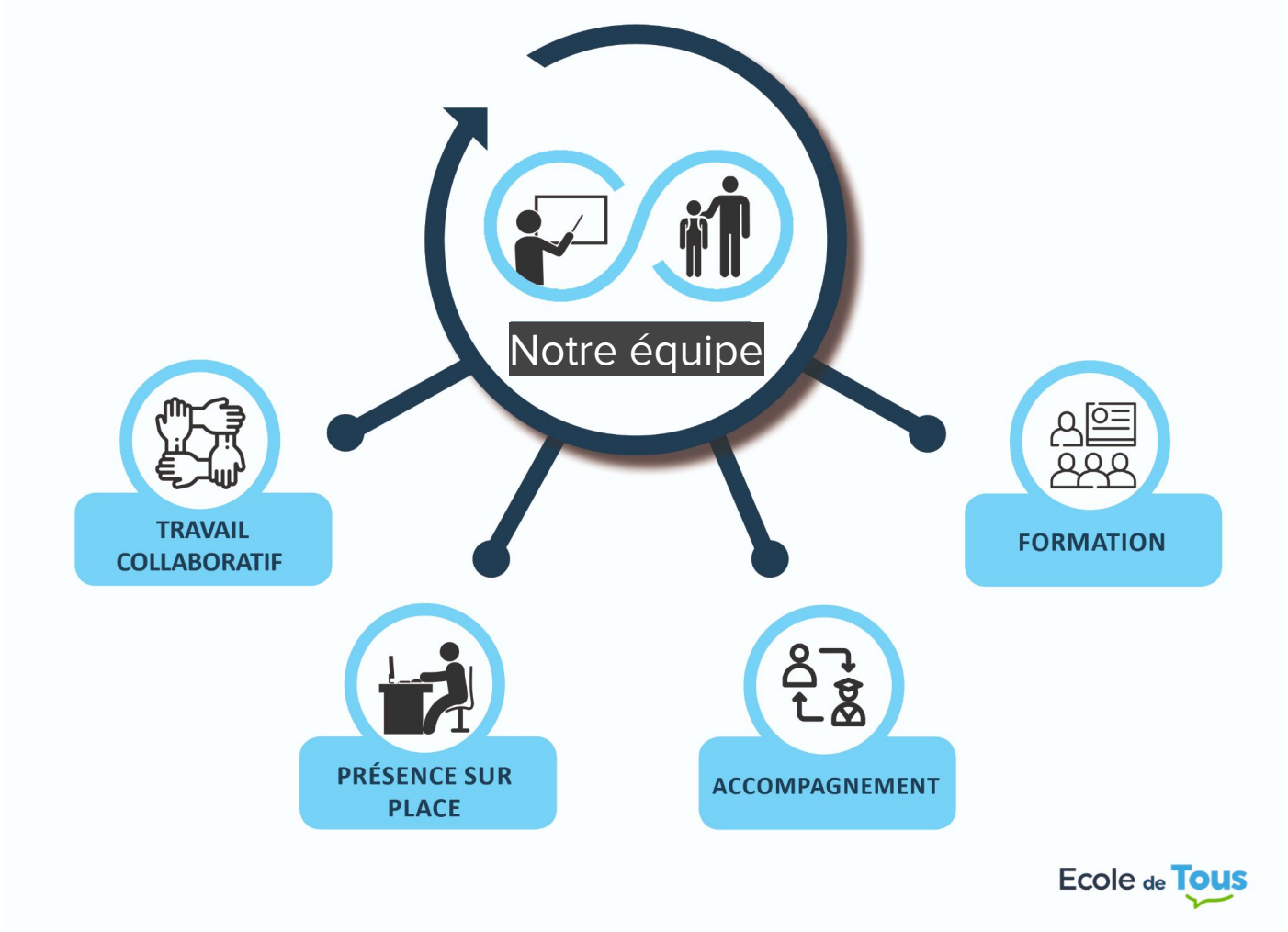
Armstrong Foundjem
Chargé de Cours (groupe 01)

Postdoc fellow DEEL (Since 2022)

Certifiability of safety-critical systems

Chair AI/ML/HPC Track,
Since 2018

Ph.D. Computing
2020-2022



Ecole de Tous

Coordonnées et disponibilités

Nom	Heng Li (professeur responsable)
Bureau	M-4121
Téléphone	
Courriel	heng.li@polymtl.ca
Disponibilité	Sur rendez-vous
Salle	Coordonnées réelles ou virtuelles

Nom	Rhouma Naceur (chargé de cours)
Bureau	
Téléphone	
Courriel	rhouma.naceur@polymtl.ca
Disponibilité	Sur rendez-vous
Salle	Coordonnées réelles ou virtuelles

Nom	Armstrong Foundjem (chargé de cours)
Bureau	M-4014
Téléphone	
Courriel	armstrong-tita-2.foundjem@polymtl.ca foundjem@ieee.org
Disponibilité	Sur rendez-vous
Salle	Coordonnées réelles ou virtuelles

Coordonnateur ou coordonnatrice

Nom	Patrick Loic Foalem (chargé de TP)
Courriel	patrick-loic.foalem@polymtl.ca

Nom	Aurel Lurich Ikama-Honey (chargé de TP)
Courriel	aurel-lucrich.ikama-honey@polymtl.ca

Nom	Altaf Allah Abbassi (chargée de TP)
Courriel	altaf-allah.abbassi@polymtl.ca

Nom	Khouloud Oueslati (chargée de TP)
Courriel	khouloud.oueslati@polymtl.ca

Programme du cours

Module 1 : Qualité fonctionnelle, Fiabilité

- Activités et processus de SQA
- Normes de qualité
- Assurance de qualité fonctionnelle
- Tests unitaires
- Tests automatiques
- Débogage
- Inspections, audits et revues
- Vérification et validation
- Intégration continue
- Livraison continue
- Conflits de fusion

Module 2 : Efficacité

- Performance
- Profiling
- Monitoring
- Allocation des ressources
- Modèle de performance
- Systèmes auto-adaptatifs
- Autoscaling

Module 3 : Sécurité

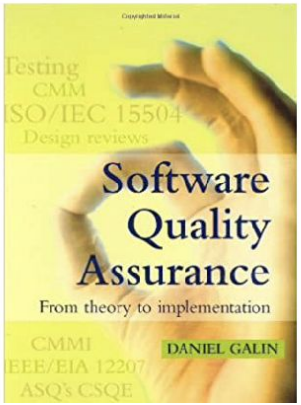
- Tests de pénétration
- Mitigation des attaques
- Chaos Engineering
- Systèmes auto-réparateurs

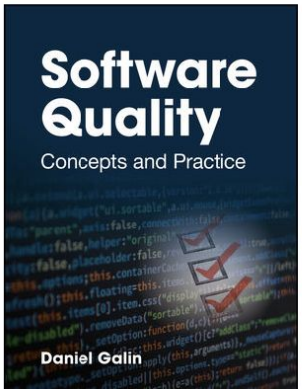
Les livres de ce cours.

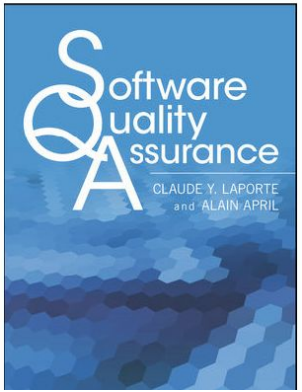
•[Primary textbook] Daniel Galin, Software Quality Assurance: From Theory to Implementation, Pearson, 2004. ISBN-10: 0201709457.

•[Primary textbook] Daniel Galin, Software Quality: Concepts and Practice, Wiley-IEEE Computer Society, 2018. ISBN: 1119134498.

•[Secondary textbook] Claude Y. Laporte and Alain April , Software Quality Assurance, Wiley-IEEE Computer Society, 2017. ISBN-10: 1118501829.
(**All textbooks are not compulsory**)







Outils

- Git
  - GitHub - https://github.com/
  - Bitbucket - https://bitbucket.org/
- Continuous Integration
  - Gitlab - https://about.gitlab.com/
  - Travis CI - https://travis-ci.org/
  - Jenkins - https://jenkins.io/
- Docker
  - Docker - https://www.docker.com/
  - Docker Compose - https://docs.docker.com/compose/
  - Kubernetes - https://kubernetes.io/
  - OpenShift - https://www.openshift.com/
- Load testing
  - Jmeter - https://jmeter.apache.org
- Chaos Engineering
  - ChaosMonkey - https://github.com/netflix/chaosmonkey
- Penetration Testing
  - THC Hydra - https://github.com/vanhauser-thc/thc-hydra

Évaluation

NATURE	NOMBRE	Mode de réalisation (Individuel/équipe)	PONDÉRATION	DATE
Examen final	1	Individuel	30%	?
Quiz	1	Individuel	20%	28 février 2024
TP1	1	Équipe	20%	12 février 2024
TP2	1	Équipe	15%	18 mars 2024
TP3	1	Équipe	15%	15 avril 2024

Les notes des travaux pratiques (devoirs) ne seront pas prises en compte dans la note finale si la note de l'examen final est inférieure à 40%.

Le quiz et l'examen final se dérouleront en présentiel. Le document autorisé pour les examens est une feuille manuscrite recto-verso (format A4/lettre).

Examen final

- L'examen final portera sur tous les cours du semestre.
- Les notes des travaux pratiques ne seront pas prises en compte dans la note finale si la note de l'examen final est inférieure à 40%.
- Individuellement
- Théorie et pratique
- Comme les exercices en classe (mais sur papier/Moodle).
- Exemples de systèmes discutés en classe, en laboratoire ou en TP.

Les quiz sont comme l'examen final mais avec moins de questions.

Note importante

Tout le matériel présenté pendant les cours ou les séances de laboratoire peut être évalué.

Le matériel sera disponible quelques jours avant.

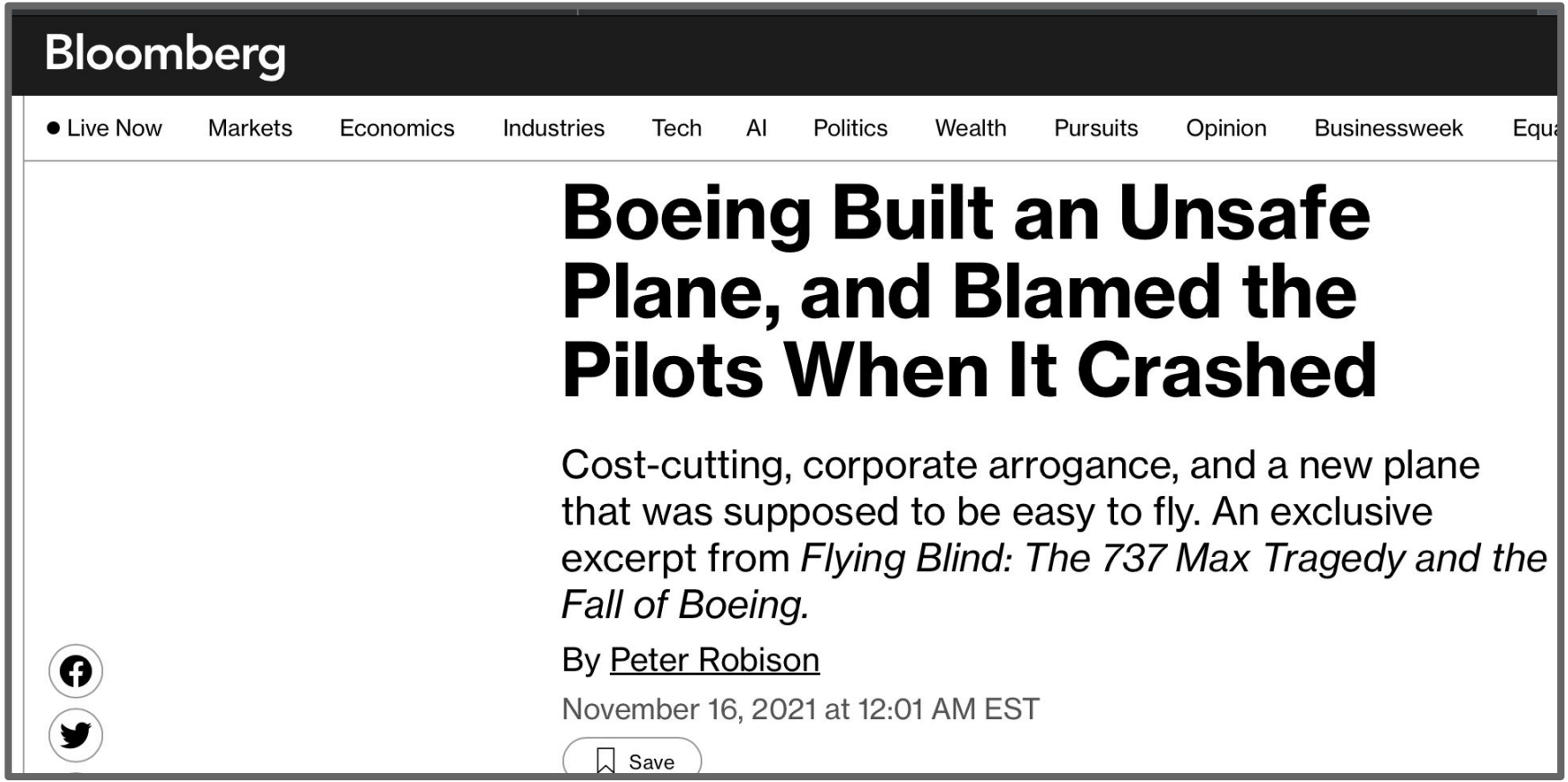
Des annonces précéderont les mises à jour du matériel et du cours.
du cours.

Les travaux remis en retard seront pénalisés de 10% par jour de retard.

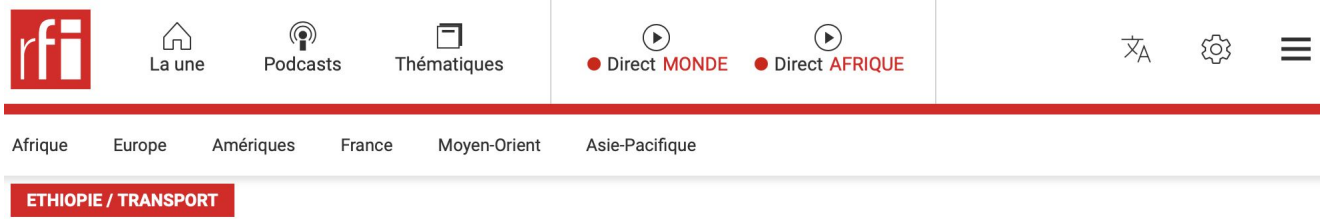
Toute demande de report d'examen doit être faite auprès du bureau des affaires académiques.

La collaboration est autorisée pour les TsP, mais les règles de plagiat s'appliquent en tout temps.

Des questions jusqu'à présent ?



https://www.bloomberg.com/news/features/2021-11-16/are-boeing-planes-unsafe-pilots-blamed-for-corporate-errors-in-max-737-crash



Dans son communiqué, la compagnie affirme que le contact a été perdu avec l'appareil à 8h44, soit six minutes après son décollage de l'aéroport international de Bole. Le Boeing s'est écrasé peu après le décollage alors que le pilote avait évoqué des « *difficultés* » et demandé à faire demi-tour en vue d'un atterrissage d'urgence. Selon la compagnie, le pilote éthiopien était expérimenté et comptait plus de 8 000 heures de vol, et le Boeing n'avait « *aucun problème technique connu* ».

https://www.rfi.fr/fr/afrique/20190310-ethiopie-boeing-ethiopian-airlines-s-ecrase-157-personnes-son-bord

The Ethiopian Airlines 737 Max crashed soon after takeoff from Addis Ababa for Nairobi. It was the second 737 Max disaster in six months after a Lion Air plane in Indonesia crashed in October 2018, killing 189 people.

A Boeing 737 Max on a test flight in Seattle. Photograph: Karen Ducey/Reuters

Investigators identified faults in the sensors and new flight control software that had not been explained to pilots.

https://www.reuters.com/world/us-safety-experts-dispute-aspects-ethiopia-737-max-air-crash-findings-2022-12-28/

Boeing

This article is more than **2 years old**

Boeing admits full responsibility for 737 Max plane crash in Ethiopia

'Significant milestone' paves way for families of 157 victims of 2019 crash to seek compensation, say lawyers

Gwyn Topham *Transport correspondent*

@GwynTopham

Thu 11 Nov 2021 15.49 GMT

Victims' families hold pictures of loved ones outside the transport department in Washington. Photograph: José Luis Magaña/AP

https://www.theguardian.com/business/2021/nov/11/boeing-full-responsibility-737-max-plane-crash-ethiopia-compensation

Final report on Boeing 737 MAX crash sparks dispute over pilot error

Examining all factors to draw safety lessons

Both the NTSB and France's Bureau of Enquiry and Analysis agreed with the Ethiopian agency's conclusion that the design of Boeing's **new flight control software** that repeatedly pushed the jet's nose down — the Maneuvering Characteristics Augmentation System, or MCAS — was a major cause of the accident.

https://www.seattletimes.com/business/boeing-aerospace/final-report-on-boeing-737-max-crash-disputed-agencies-note-pilot-error-as-a-factor/

Après la mort de 157 passagers, dimanche 10 mars 2019, dans le crash d'un Boeing 737 MAX 8 en Ethiopie, les regards se tournent vers l'avionneur américain. Ce crash du vol Addis-Abeba–Nairobi présente en effet beaucoup de similitudes avec celui de la compagnie indonésienne Lion Air, qui avait 189 morts en octobre. En cause dans les deux accidents : le dernier-né de Boeing, son modèle phare. Plusieurs pays, dont la Chine, ont décidé de clouer au sol leurs propres appareils.

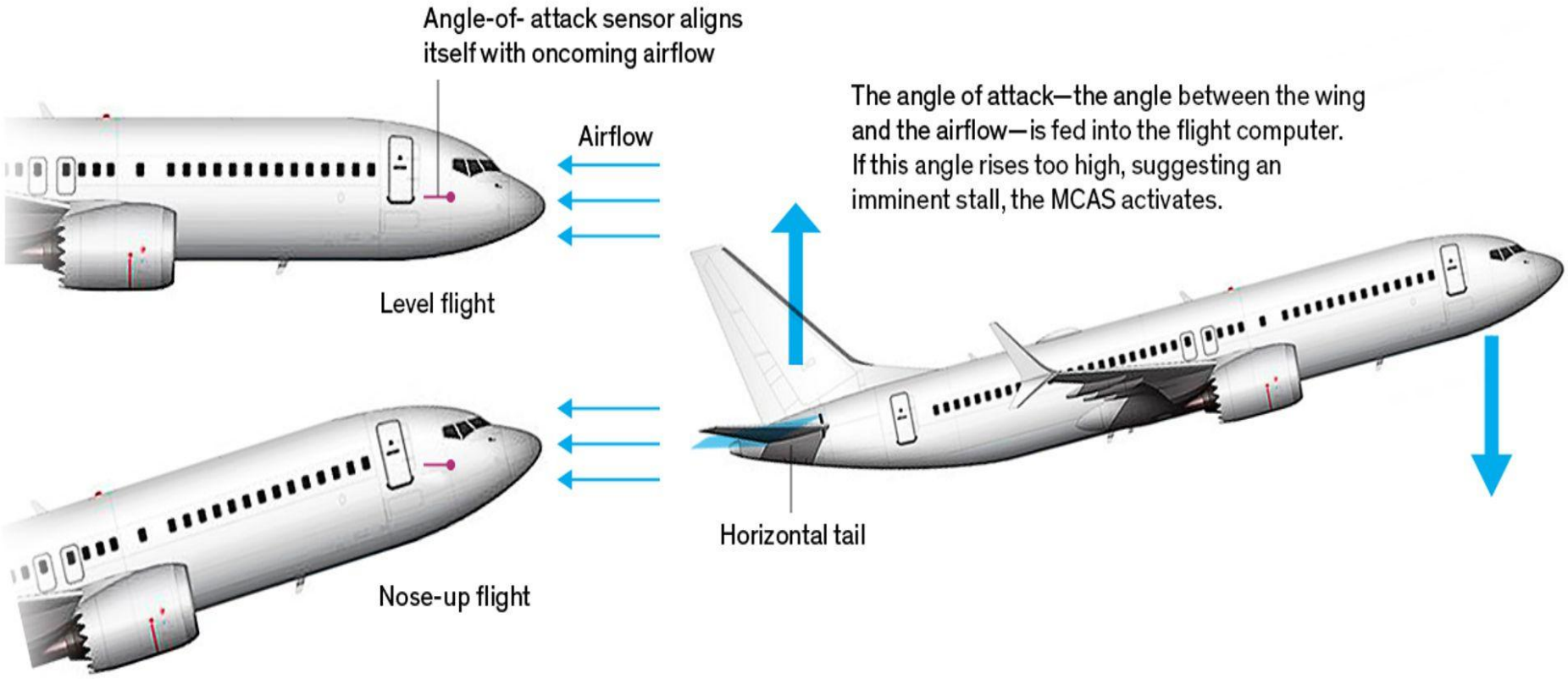
FEATURE | AEROSPACE

HOW THE BOEING 737 MAX DISASTER LOOKS TO A SOFTWARE DEVELOPER

Design shortcuts meant to make a new plane seem like an old, familiar one are to blame

https://spectrum.ieee.org/how-the-boeing-737-max-disaster-looks-to-a-software-developer

How the new Max flight-control system (MCAS) operates to prevent a stall



Error, Fault, Failure

$$(\forall x \in \mathbb{Z}), (\text{square}(x) = x \times x)$$

```
1.  #include <iostream>
2.  int square(int x) {
3.      // Correct usage: Multiply the input by itself to get the square
4.      return x + x;
5.  }
6.  int main() {
7.      // Example usage
8.      int inputValue = 5;
9.      int result = square(inputValue);
10.     // Display the result
11.     std::cout << "The squared value is: " << result << std::endl;
12.     return 0;
13. }
```

Introduction to Software Quality Assurance

Corresponding readings

References:

- Introduction to SQA: Book of Galin (04), Chapters 1-4.
- Quality standards: Book of Laporte/April, Chapter 3.
- Quality plan: Book of Galin (04), Chapter 6.

Supplementary resources:

- ISO 25000 standards (Software quality)

Check materials on Moodle.

Software bug releases thousands of US prisoners early

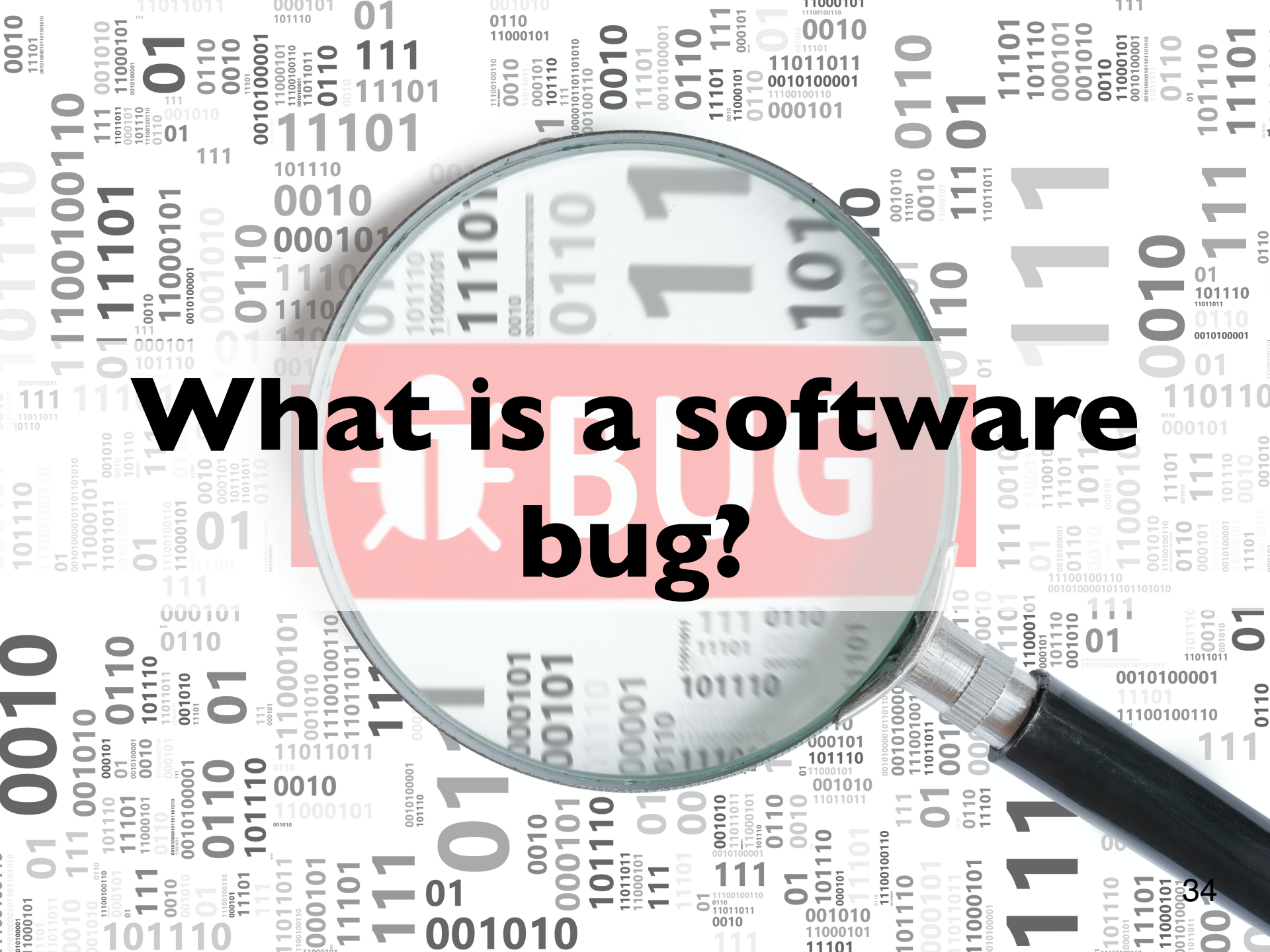
The bug was introduced in 2002 and lasted for 13 years

Caused more than 3,200 prisoners to be released up to two years early

[Source: BBC News 2015]

In 2016, software bugs cost the worldwide economy **$1.1 trillion**.

[Source: tricentis.com 2016]

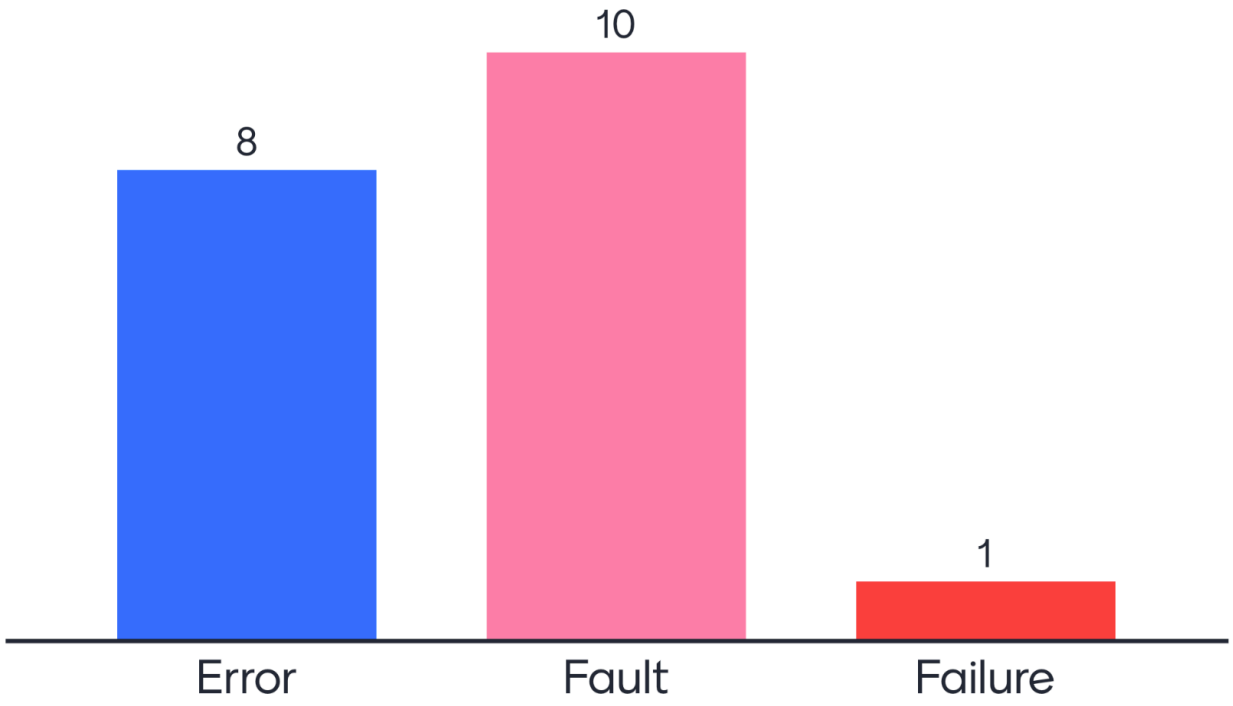

What is a software bug?

Exercise: which concept is closest to "bug"

1. Go to www.menti.com
2. Use the code **9180198**
3. Choose the concept that is closest to **"bug"**?
   A. Error
   B. Fault
   C. Failure

https://www.mentimeter.com/s/341b5110f6cab49c705a76b77209dbdf/8b3c063926f3

Choose the concept that is closest to "bug"



Objectives

- What is software quality?
- Why is software quality important?
- Software quality factors
- What is software quality assurance?
- Elements of software quality assurance

Next week preview:

- Software quality standards
- Quality plans
- Software testing

Software Quality

What is Software and Software Engineering?

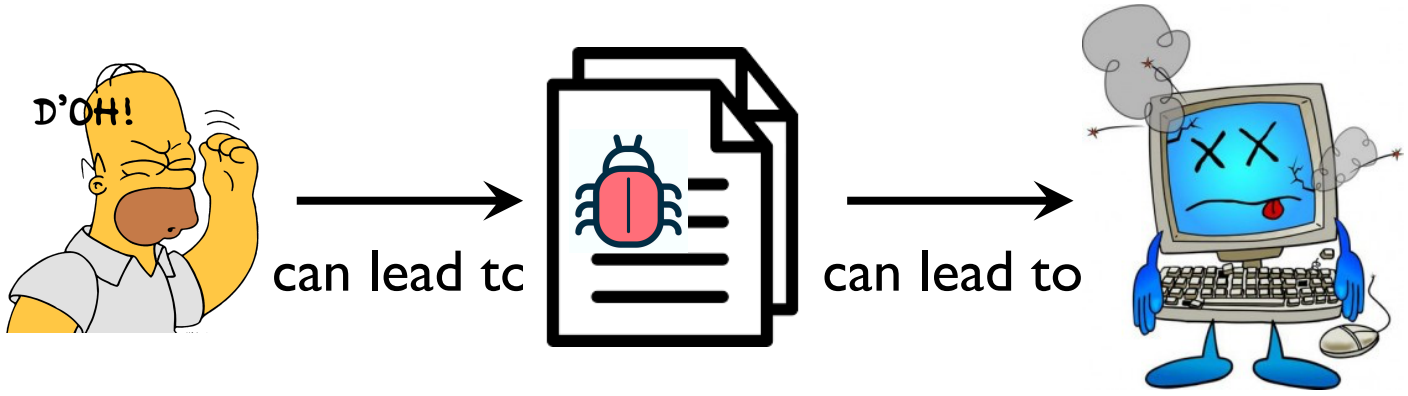
according to the IEEE:

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

software engineering = (1) the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software; (2) the study of approaches as in (1)

Failure, fault, and error

- Failure
  - Observable incorrect **behavior** of a program. Related to the behavior of the program rather than its code.
- Fault (bug)
  - Necessary (not sufficient) condition for the occurrence of a failure. Related to the **code**.
- Error
  - A mistake usually made by **people**. Cause of a fault.



An example of the failure, fault, error

```
1.  int double (int param) {
2.      int result;
3.      result = param * param;
4.      return result;
5.  }
```

A call to double(3) returns 9

- Result 9 represents a **failure**
- Such failure is due to the **fault** at line 3
- The **error** is a typo (hopefully)

Nine Causes of Software Errors

1) Faulty requirements definition (incorrect, missing, incomplete, unnecessary requirements)

2) Client–developer communication failures

3) Deliberate deviations from software requirements (improper reuse, omission due to time pressure)

4) Logical design errors (problems with algorithms, sequencing of actions, boundary conditions, missing states, how to handle "illegal" input)

Nine Causes of Software Errors

5) Coding errors

6) Non-compliance with documentation and coding instructions (more difficult to deal with team attrition and inspections)

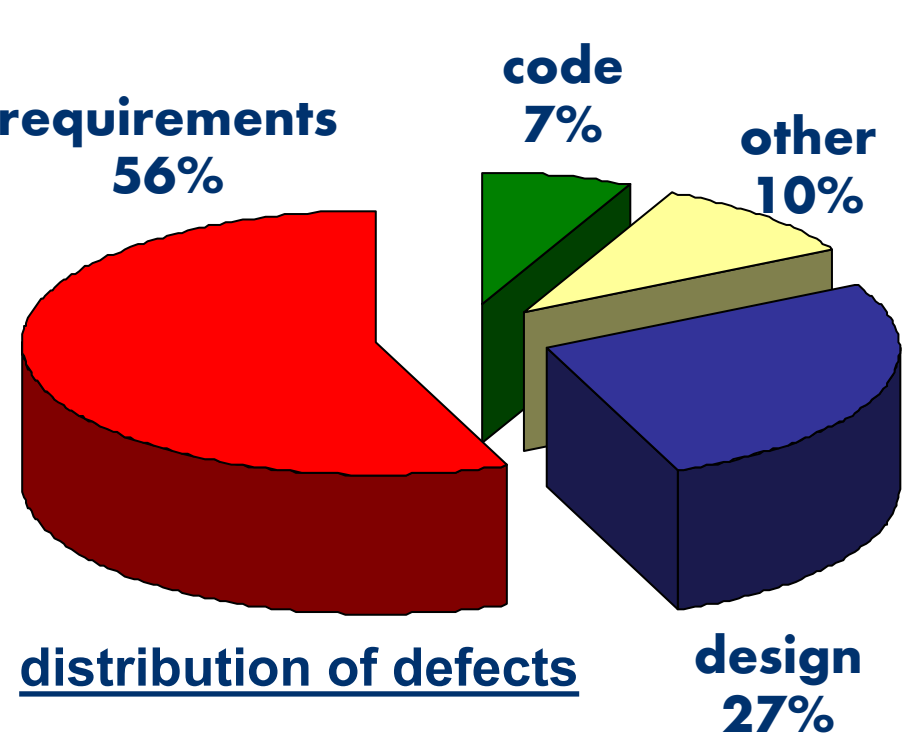
7) Shortcomings of the testing process

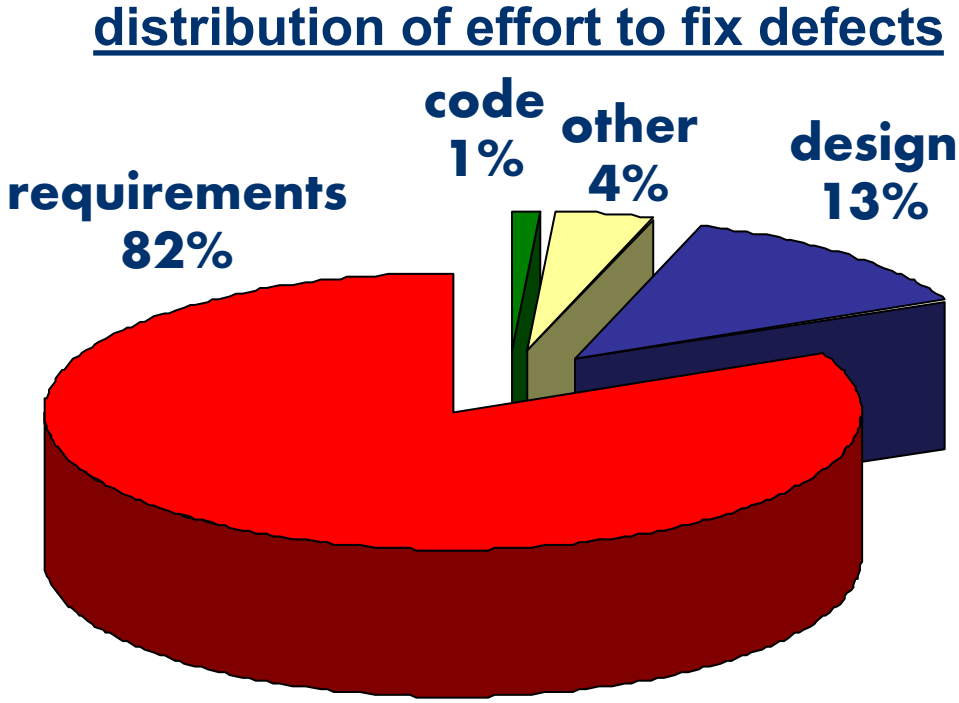
8) Procedure errors (workflow and user interface errors)

9) Documentation errors

Development Phases vs. Defects

- majority of defects are introduced in earlier phases





- requirements are the top reason for project success or failure

Source: Martin & Leffinwell

Development Phases vs. Cost

relative cost of fixing defects

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

What is Software Quality?

according to the IEEE:

software quality = (1) the degree to which a system, component, or process meets specified requirements; (2) the degree to which a system, component, or process meets customer or user needs or expectations

according to Pressman:

software quality = conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software

Importance of Software Quality

- Software is a major component of computer systems (about 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 may be extremely costly in terms of
  - Money
  - Reputation
  - Loss of life

Importance of Software Quality

- Zune 30 leap year freeze:

- On December 31st 2008, players began freezing at about midnight becoming totally unresponsive and practically useless

- Official fix:
  - Wait until January 1st 2009

Importance of Software Quality

- Several historic disasters attributed to software:
  - 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
  - 1992 London Ambulance Service Computer Aided Dispatch System – blamed for 30-45 deaths
  - Therac-25: radiation therapy and X-ray machine killed several patients between 1985 and 1987; cause: unanticipated, non-standard user inputs

Importance of Software Quality

- Ariane 5 crash – June 4, 1996:
  - Maiden flight of the European Ariane 5 launcher crashed about 40 seconds after takeoff
  - Loss 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

Importance of Software Quality

- Ariane 5 crash – June 4, 1996: (cont'd)
  - Runtime error (out of range, overflow) was detected and computer shut itself down
  - Same for the backup computers
  - This resulted in the total loss of attitude control
  - Ariane 5 turned uncontrollably and aerodynamic forces broke the vehicle apart
  - Breakup was detected by an on-board monitor which ignited the explosive charges to destroy the vehicle in the air

  - Ironically, the result of the format conversion was no longer needed after lift off

Importance of Software Quality

- Ariane 5 crash – June 4, 1996: (cont'd)
  - Module was reused without proper testing from Ariane 4
  - Error was not supposed to happen with Ariane 4 (it was shown that such a large input could not occur in the context of Ariane 4, no exception handler)
  - Note this was not a complex, computing problem, but a deficiency of the software engineering practices in place …

Importance of Software Quality

- The Heartbleed Bug:
  - Serious vulnerability in the popular OpenSSL cryptographic software library
  - Potentially affected open source web servers like Apache and nginx with a combined market share of over 66%; plus email servers, chat servers, and VPNs
  - Out in the wild since March 14, 2012 – fixed April 7, 2014
  - August 2014: personal data of 4.5 million patients of U.S. hospital group Community Health Systems Inc. stolen by exploiting the Heartbleed bug

http://heartbleed.com/

Importance of Software Quality

- Pervasive problems:
  - Software is commonly delivered late, way over budget, and of unsatisfactory quality
  - Software validation and verification are rarely systematic and are usually not based on sound, well-defined techniques
  - Software development processes are commonly unstable and uncontrolled
  - Software quality is poorly measured, monitored, and controlled

Importance of Software Quality

- NIST (National Institute of Standards and Technology) study (2002):
  - bugs cost US economy $ 59.5 billion a year—earlier detection could save $22 billion

- Tricentis.com (2016):
  - In 2016, software bugs cost the worldwide economy $1.1 trillion.

The Software Quality Challenge

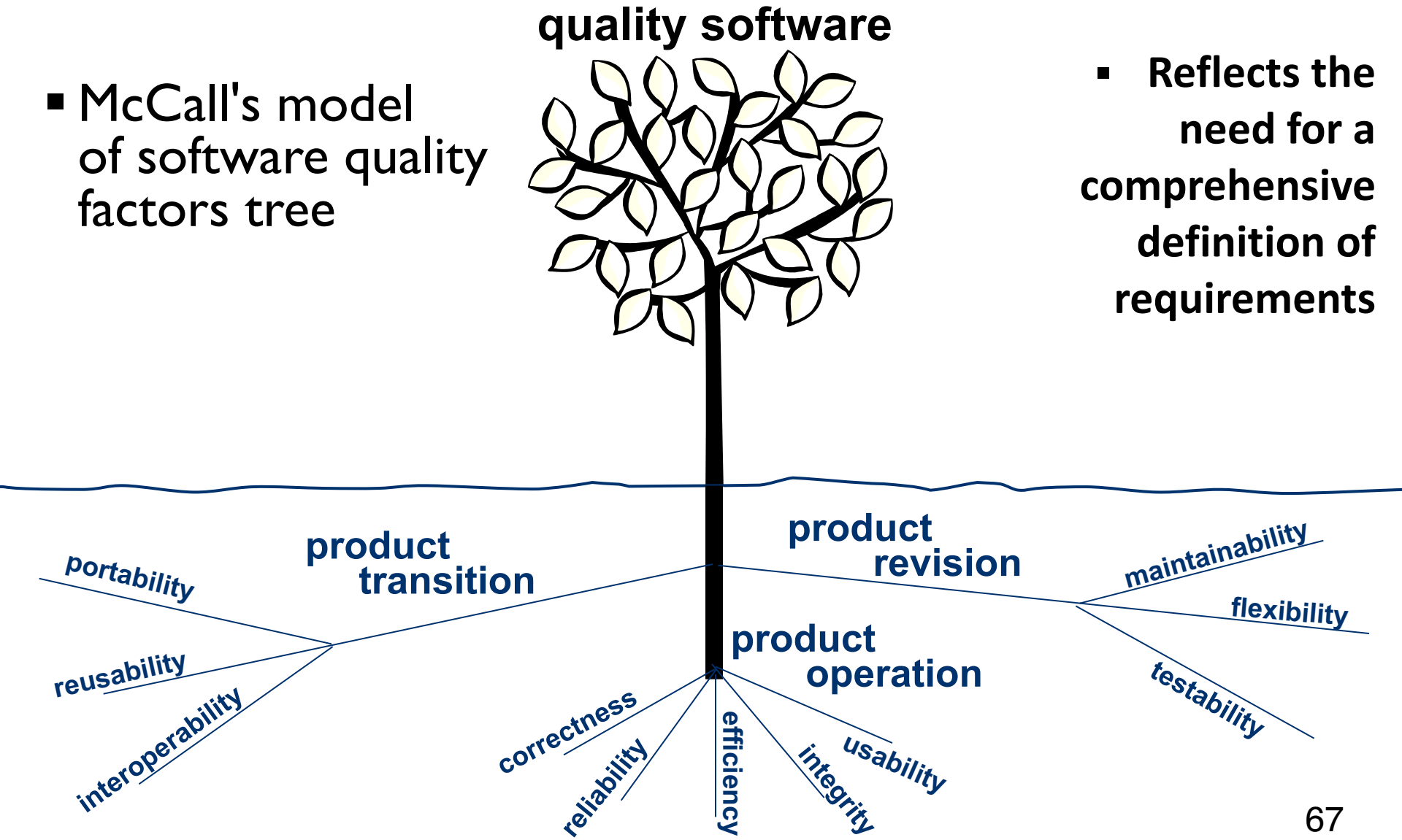
- The **uniqueness** of the software product:
  - High complexity (and increasingly so) – pervasive in an increasing number of industries
  - Invisibility of the product
  - Limited opportunities to detect defects compared to other industries
    - Development, not production (only opportunity to detect defects is product development; product production planning not required; simple manufacturing)

The Software Quality Challenge

- The environments in which software is developed:
  - Contracted (features / budget / timetable)
  - Subjection to customer-supplier relationship (potential miscommunications, change request management)
  - Requirement for teamwork (human-intensive; engineering but also a social process)
  - 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 system for years

Software Quality Factors

- McCall's model of software quality factors tree



- **Reflects the need for a comprehensive definition of requirements**

Quality factors – product operation

- Correctness
  - Accuracy and completeness of required output
  - Up-to-dateness and availability of the information
- Reliability
  - Maximum allowed failure rate
- Efficiency
  - Hardware resources needed to perform software function (processing capabilities, data storage, bandwidth, power usage)

Quality factors – product operation

- Integrity
  - Software system security, access rights
- Usability
  - Training required, ability to learn and perform required task

Quality factors – product revision

- Maintainability
  - Effort to identify and fix software failures (modularity, documentation, etc)
- Flexibility
  - Degree of adaptability (to new customers, tasks, etc)
- Testability
  - Support for testing (e.g., log files, automatic diagnostics, etc), traceability

Quality factors – product transition

- Portability
  - Adaptation to other environments (hardware, software)
- Reusability
  - Use of software components for other projects
- Interoperability
  - Ability to interface with other components/systems

- Other factors: robustness, performance, user friendliness, verifiability, repairability, evolvability, understandability, safety, manageability

Software Quality Assurance

What is Software Quality Assurance?

according to the IEEE:

software quality assurance = (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 – contrast with: quality control

quality control: set of activities designed to evaluate the quality of a developed or manufactured product – after development before shipment

quality assurance aims to minimize the cost of guaranteeing quality

What is Software Quality Assurance?

according to D. Galin:

software quality assurance = a systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system product conforms to established functional technical requirements as well as with the managerial requirements of keeping the schedule and operating within the budgetary confines

Objectives of Software Quality Assurance (SQA)

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

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

3) Initiation and management of activities for the improvement and greater efficiency of software development, software maintenance, and software quality assurance activities

Three General Principles of Software Quality Assurance

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

3 General Principles of SQA

- Know what you are doing:
  - Understand what is being built, how it is being built and what it currently does
  - Implies a software development process with
    - Management structure (milestones, scheduling)
    - Reporting policies
    - Tracking

3 General Principles of SQA

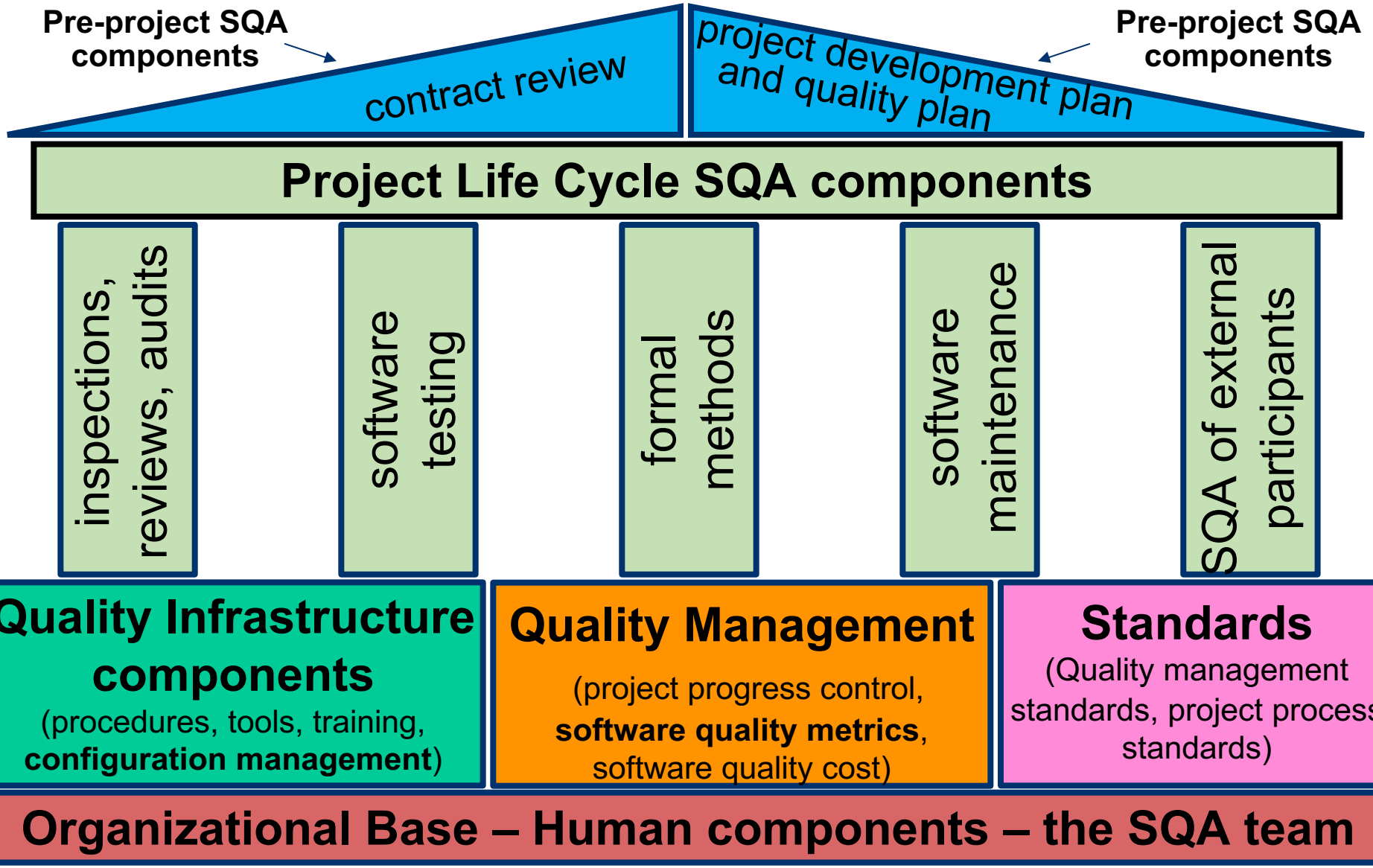
- Know what you should be doing:
  - Having explicit requirements and specifications
  - Implies a software development process with
    - Requirements analysis
    - Acceptance tests
    - Frequent user feedback

3 General Principles of SQA

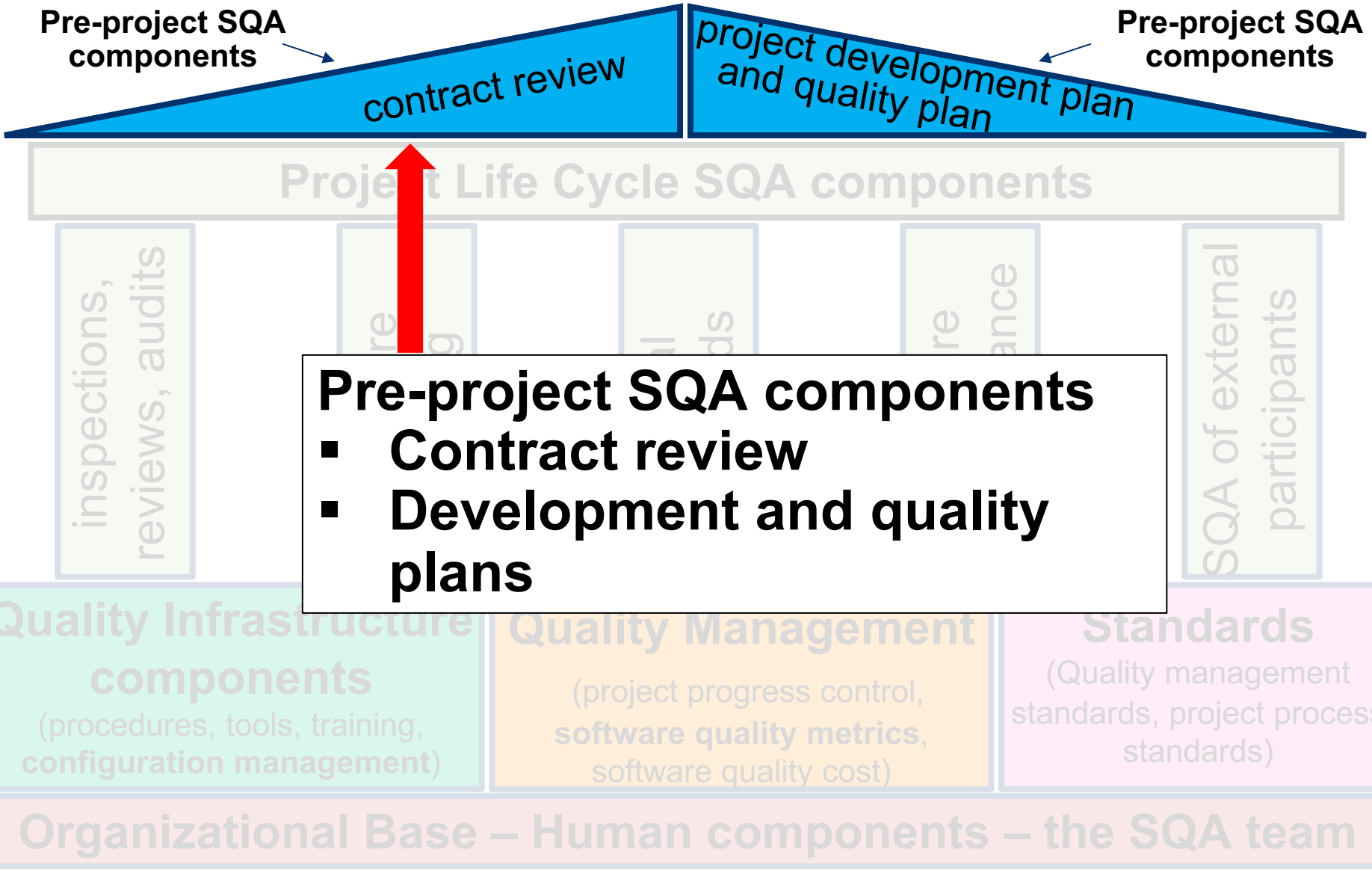
- Know how to measure the difference:
  - Having explicit measures comparing what is being done with what should be done
  - Four complementary methods:

1) Formal methods – verify mathematically specified properties
2) Testing – explicit input to exercise software and check for expected output
3) Inspections – human examination of requirements, design, code, … based on checklists
4) Metrics – measure a known set of properties related to quality

“The Software Quality Shrine”



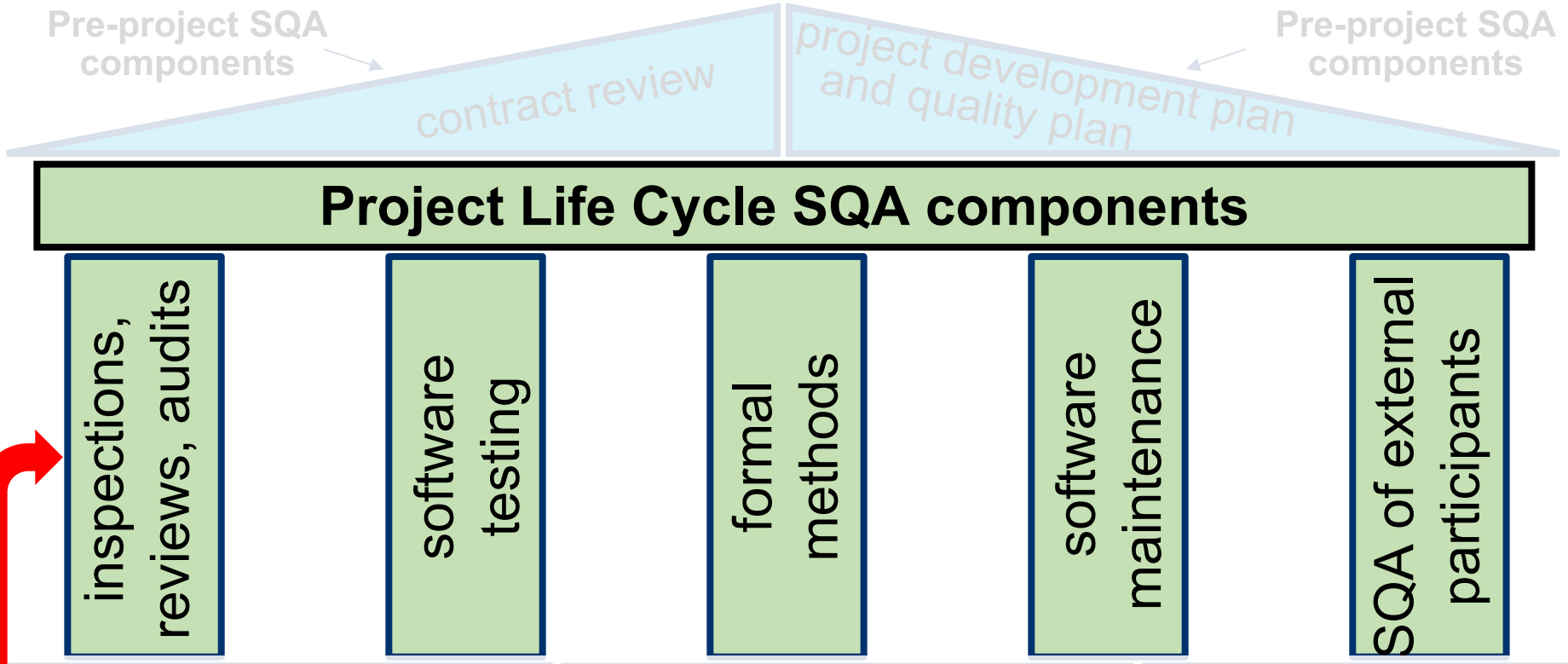
"The Software Quality Shrine"



Pre-project SQA components

- **Contract review**
- **Development and quality plans**

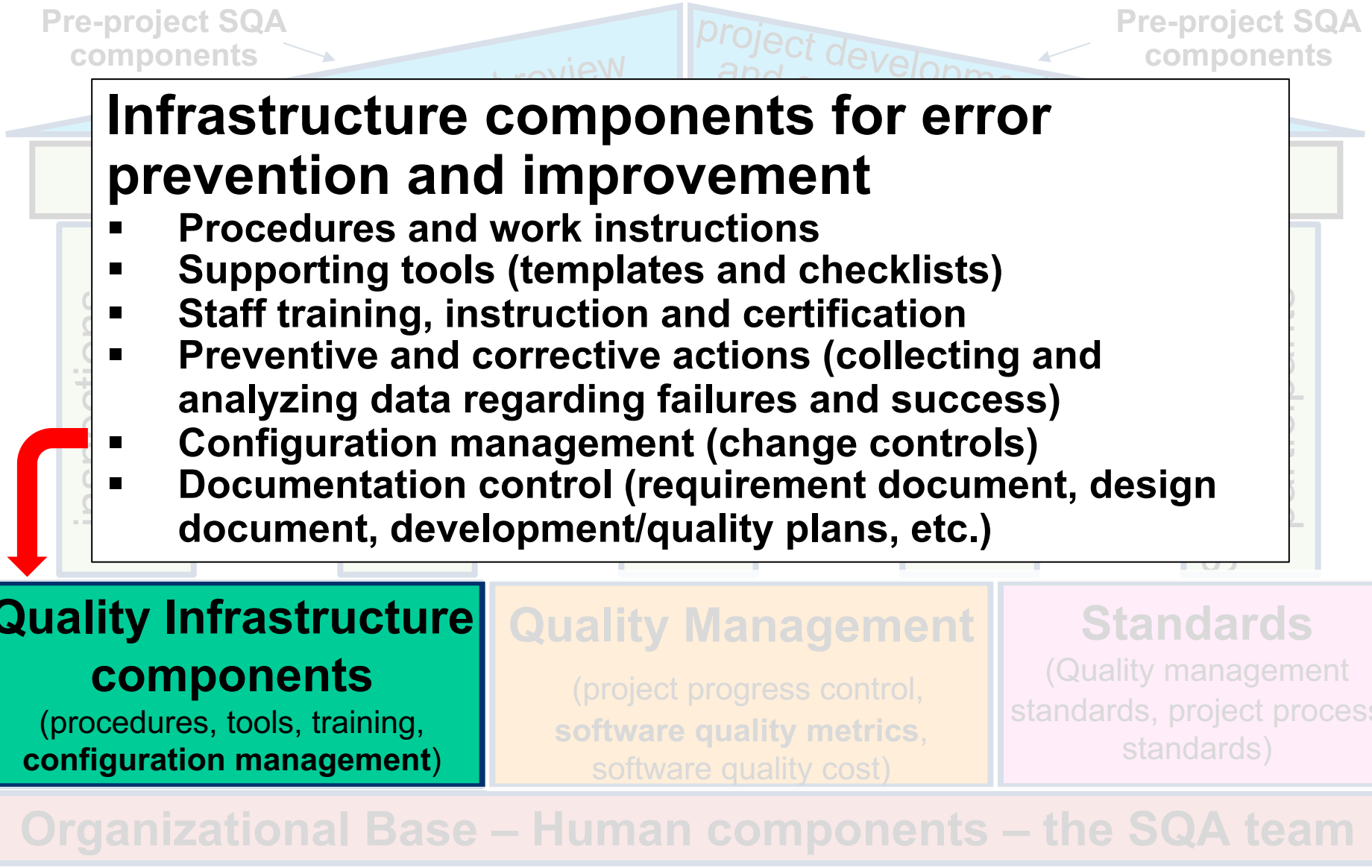
“The Software Quality Shrine”



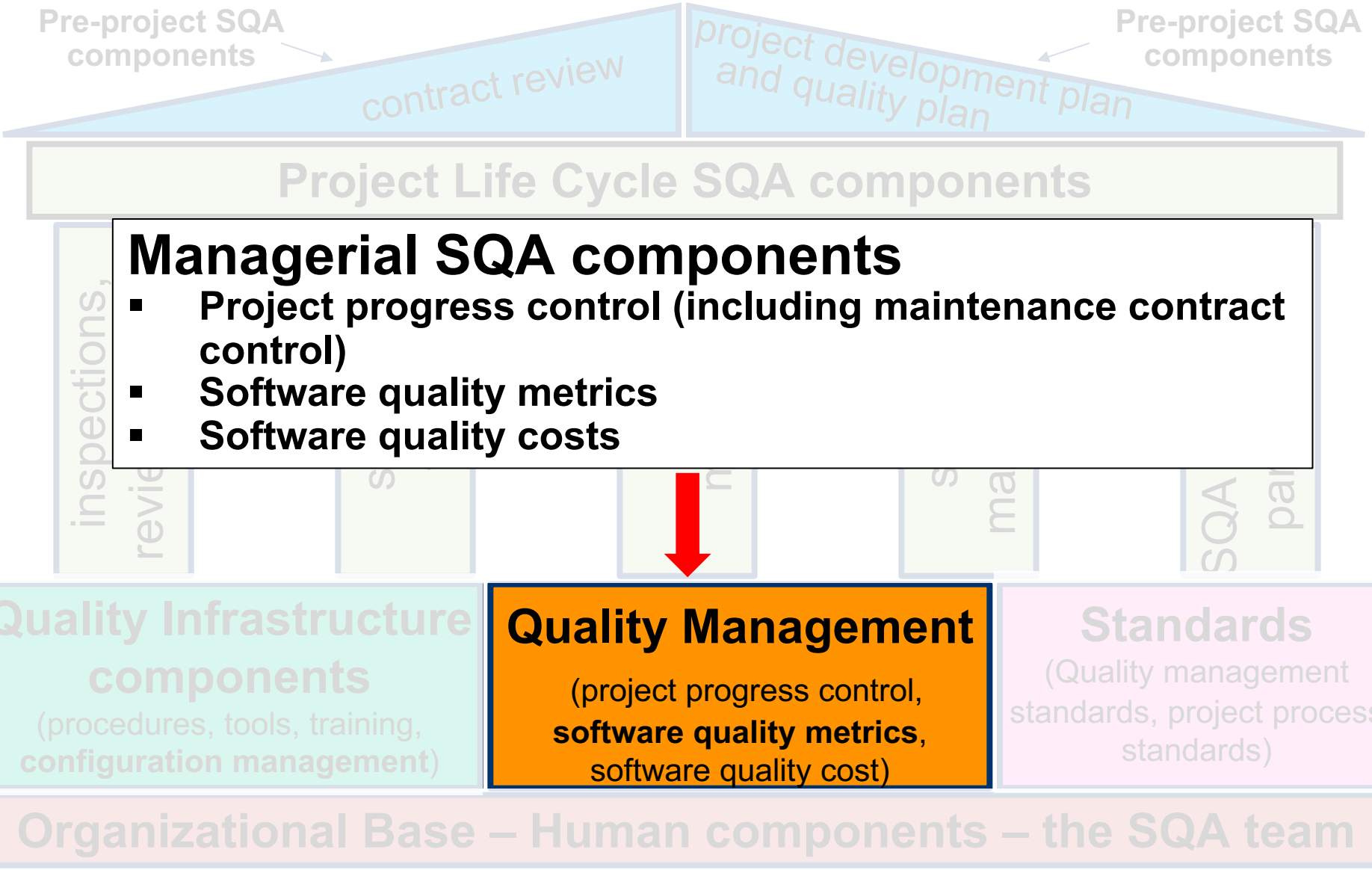
Development life cycle stage and operation-maintenance stage:

- Reviews, inspections, audits
- Software testing
- Formal methods
- Software maintenance
- SQA of external participants

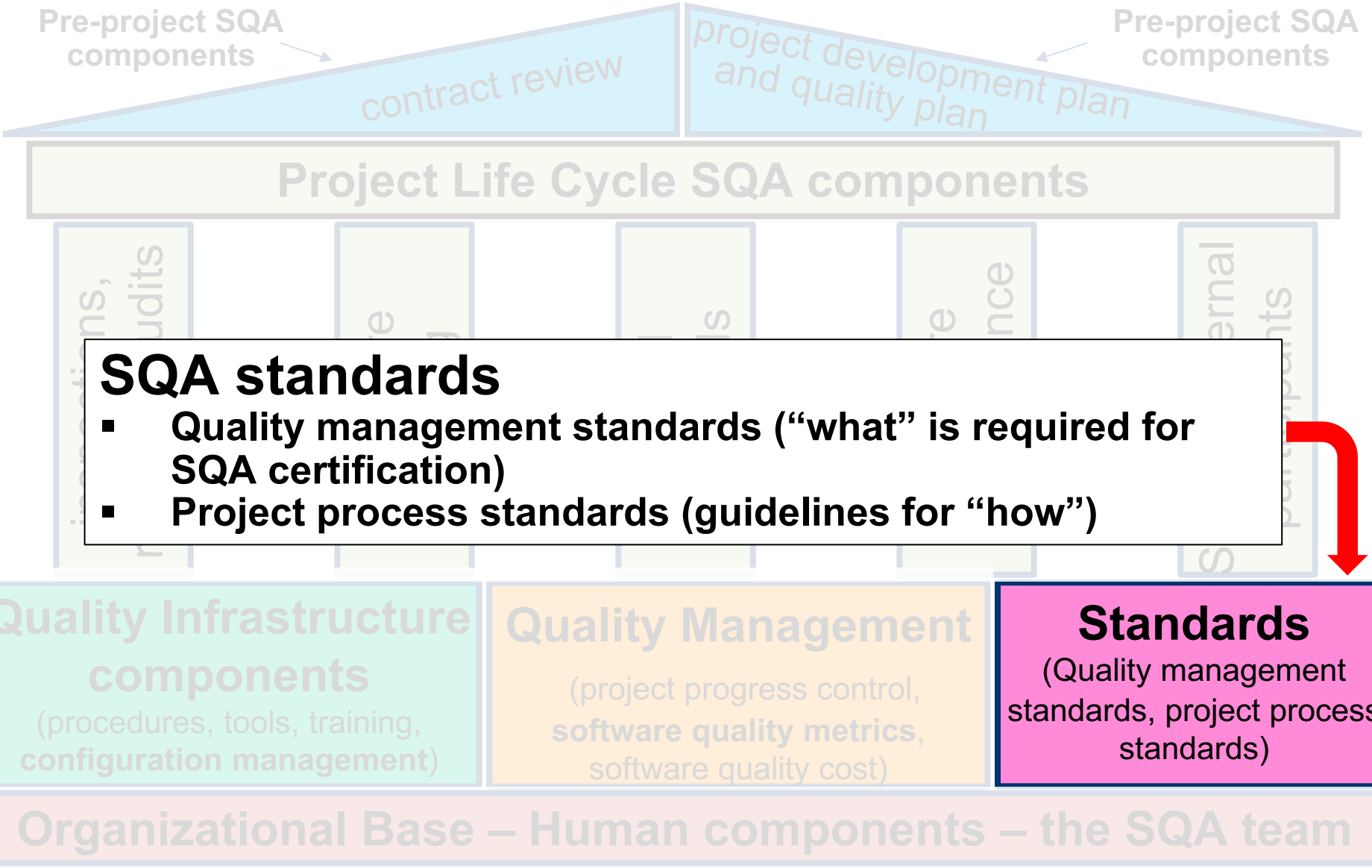
“The Software Quality Shrine”



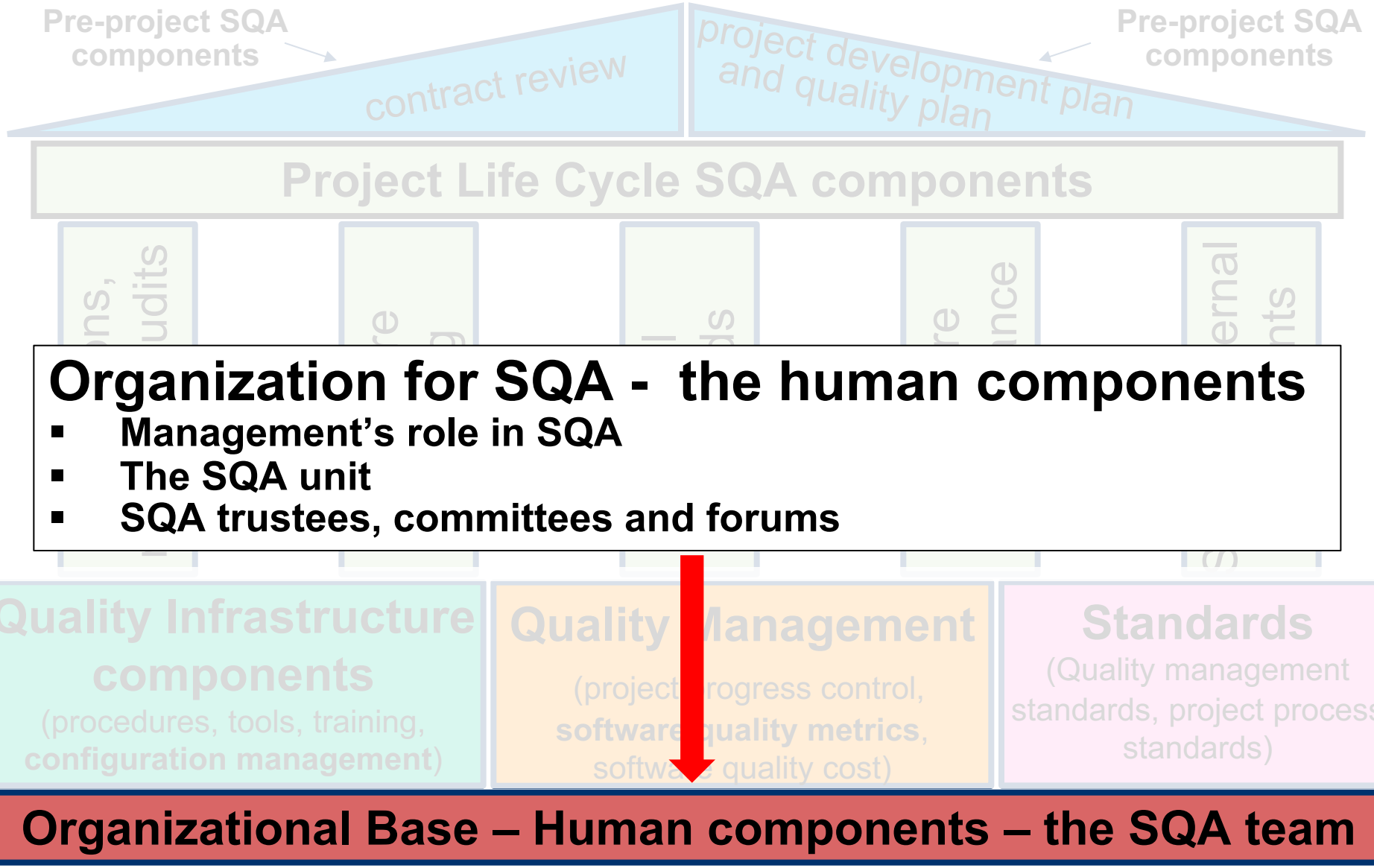
“The Software Quality Shrine”



"The Software Quality Shrine"



“The Software Quality Shrine”



Organization for SQA - the human components

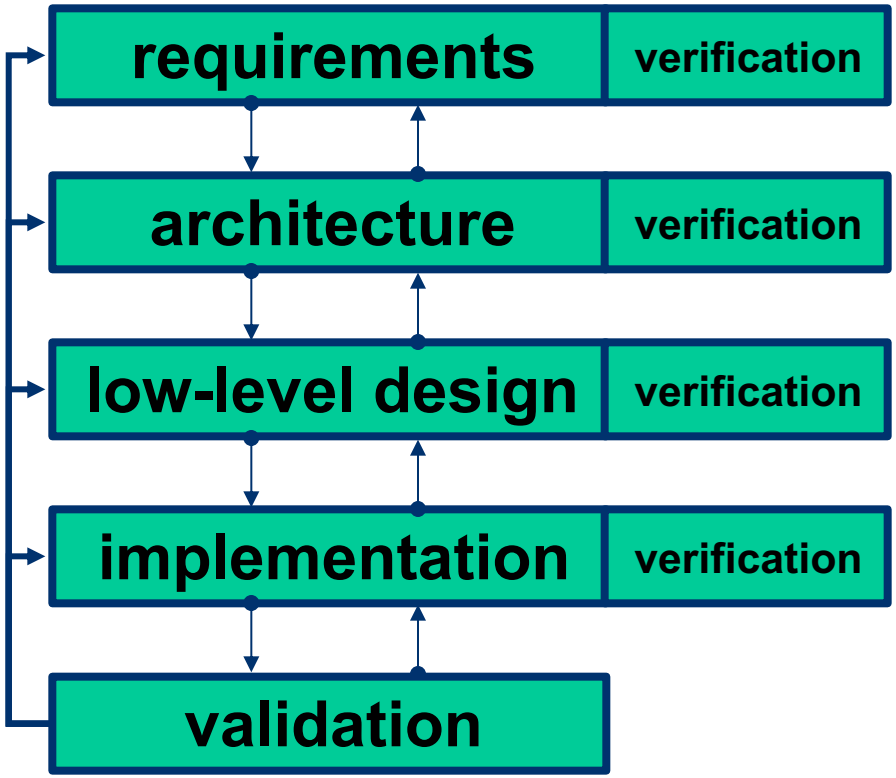
- Management’s role in SQA
- The SQA unit
- SQA trustees, committees and forums

Software Quality Assurance

- Verification
  - Are we building the product right?
- Validation
  - Are we building the right product?

Software Quality Assurance

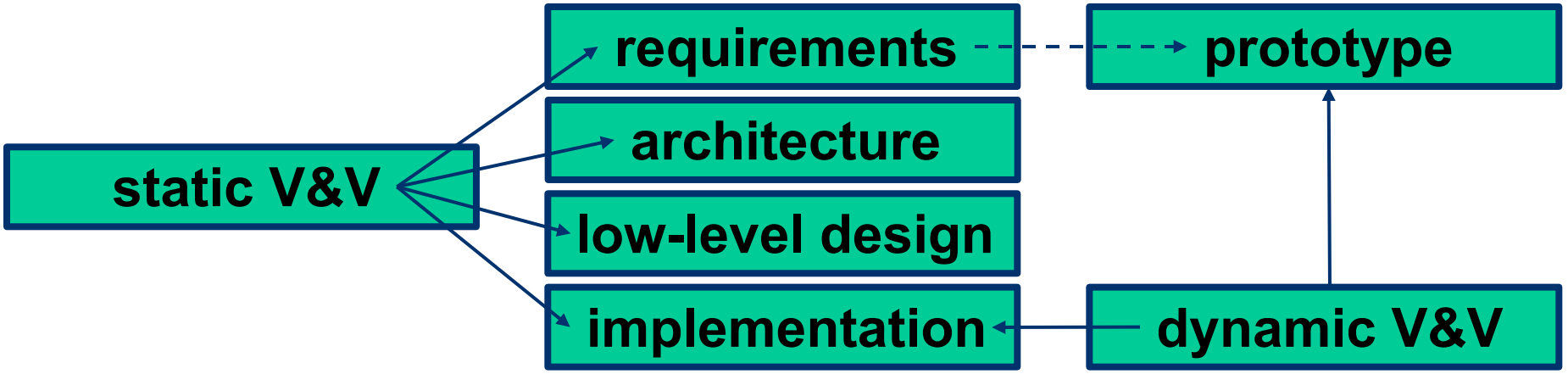
- SQA includes V&V:



- Verification
  - Performed at the end of a phase to ensure that requirements established during the previous phase have been met
- Validation
  - Performed at the end of the development process to ensure compliance with user expectations

Software Quality Assurance

- V&V includes static V&V and dynamic V&V



Software Quality Assurance

SQA includes:

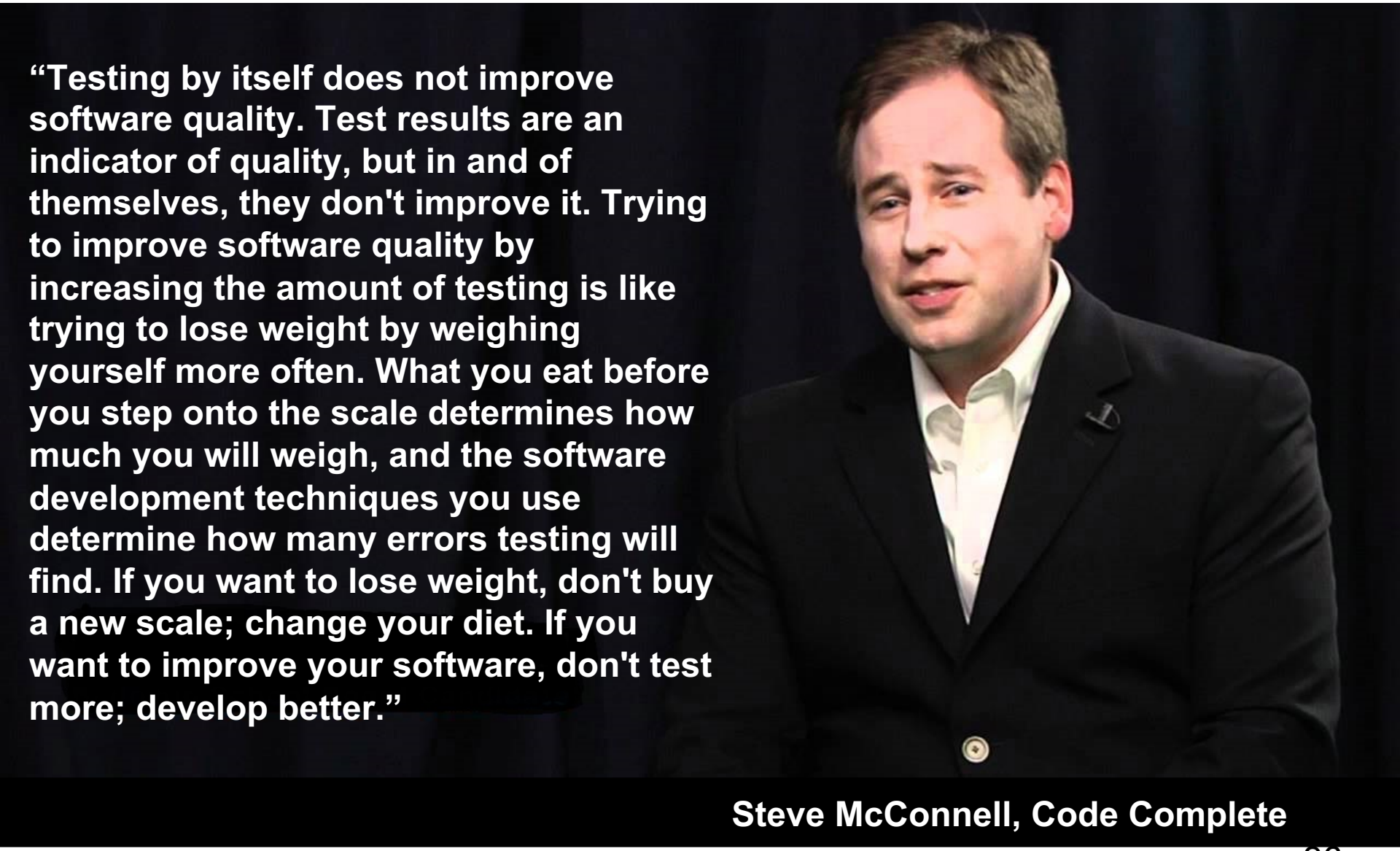
- Defect prevention
  - Prevents defects from occurring in the first place
  - Activities: training, planning, and simulation
- Defects detection
  - Finds defects in a software artifact
  - Activities: inspections, testing, or measuring
- Defects removal
  - Isolation, correction, verification of fixes
  - Activities: fault isolation, fault analysis, regression testing

Things to do

- Find teammates (ideally group of 3, maximum 4).
- Send me the name of your team and the list of your team members (names, student numbers) by Sep. 16.
- Signup on our Slack workspace
- Check assignment TP0 (due Nov. 30th)

Exercises

Closing Thought & Discussion



“Testing by itself does not improve software quality. Test results are an indicator of quality, but in and of themselves, they don't improve it. Trying to improve software quality by increasing the amount of testing is like trying to lose weight by weighing yourself more often. What you eat before you step onto the scale determines how much you will weigh, and the software development techniques you use determine how many errors testing will find. If you want to lose weight, don't buy a new scale; change your diet. If you want to improve your software, don't test more; develop better.”

Steve McConnell, Code Complete

Importance of Software Quality

- Zune 30 leap year freeze:

```
/* days: the number of days since January 1, 1980. */
year = ORIGINYEAR; /* = 1980 */
while (days > 365)
{
        if (IsLeapYear(year))
        {
                if (days > 366)
                {
                        days -= 366;
                        year += 1;
                }
        }
        else
        {
                days -= 365;
                year += 1;
        }
}
```

source of the problem?

Source of Errors?

Eiffel decides to grant a 5% discount at the beginning of the month to those customers with total purchases in excess of $1 million in the last 12 months except for those customers that returned in excess of 10% of their purchases during the last three months.

At the end of each year, Eiffel's central information processing department:

(1) Collects the previous year's sales data for each of the customers from all the chain's stores.

(2) Calculates the cumulative purchases of each customer for the previous year in all the chain's stores.

(3) Prepares a list of all customers whose purchases exceed $1 million and distribute it to all stores.

At the end of each quarter, the individual store's information processing unit:

(1) Calculates the percentage of goods returned during the last quarter for each customer.

(2) Prepares a list of all customers whose returned goods for the last quarter exceed 10% of that quarter's purchase.

At the beginning of the month, the store's information processing unit:

(1) Processes all monthly purchases for each of the customers.

(2) Calculates the discount according to the last year's purchase data in all stores, and according to the store's records of returns in the last quarter.