MASTER OF SCIENCE
IN ENGINEERING

# TSM_AdvEmbSof

## How to Improve Software Quality

Serge Ayer | 02.10.2023 | Cours MSE

swissuniversities

Hes·so
Master

# Why do we need to care?

- It pays to make the effort
  - Poor software quality have a very high cost, meaning trillions of US$
  - Poor quality also causes vulnerabilities
- It is sometimes difficult to make the effort
  - Plans that show lower investments and shorter milestones are privileged
  - Deadline pressure
  - Choosing a better solution may take longer in the initial, original implementation
- Technical debt
  - It is the cost of reworking a solution in the future, specifically because of shortcomings chosen to save time at an earlier stage
  - Should be of concern for every software development project

# What does quality mean?

- Code clarity / Improved reliability:
  - Can be easily read and understood by different developers
  - Proper naming and comments (explaining choices and changes for instance)
- Easy maintainability
  - Minor changes to the codebase or the environment should not break or significantly affect the application
  - Prevent changes that have hidden effects outside of their scope
- Improved extensibility
  - Modularity and portability are important
  - Follow the SOLID principle (OO principle)

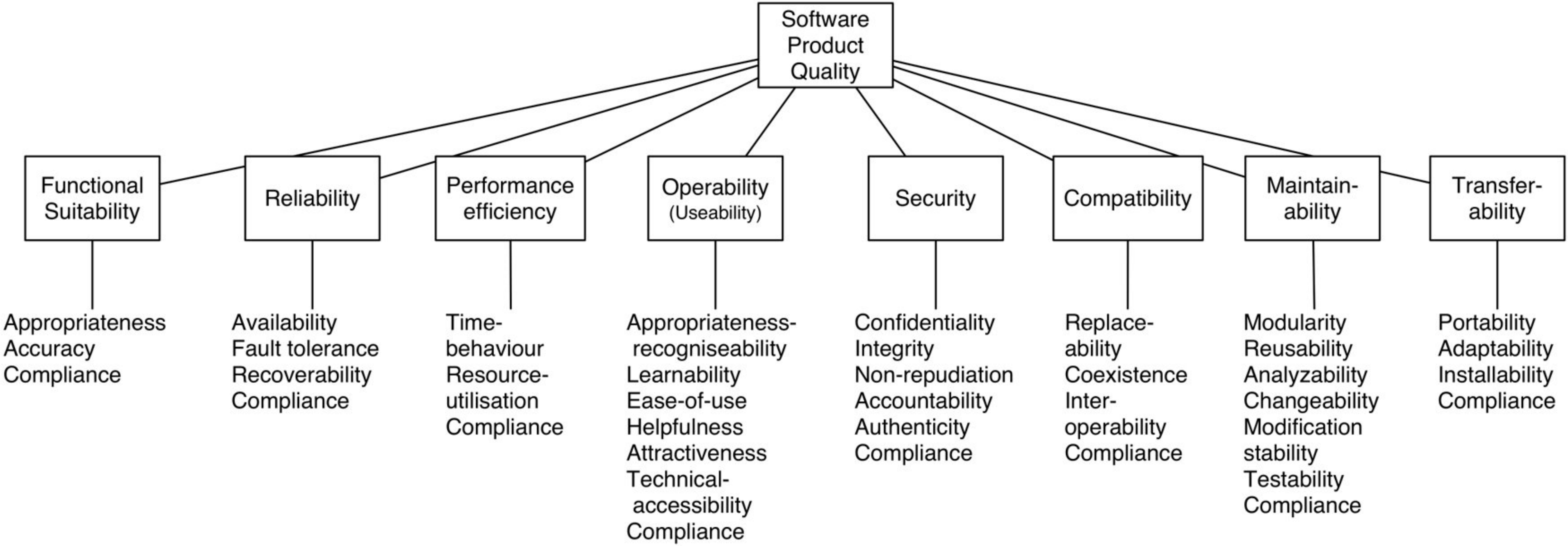
# What does quality mean?

- It applies to any software and programming language
- It starts with correct code
  - Works as specified
  - Passes all functional tests
  - Has no known bug
- And it also assumes that
  - Coding guidelines exist and are enforced
  - Tools are used whenever possible for enforcing guidelines and quality

# Ultimately, what does it bring?

- Reduces technical debt
  - Introducing new features is easier
- Improves stability
  - Easier to test
- Leads to fewer bugs
  - Each bug is a debt
- Eases the detection and resolution of errors
  - Bug corrections cost less (easier to identify and fix)

# Sofware quality: the full ISO-25010 picture



Source: https://nocomplexity.com/wp-content/uploads/2016/08/ISO-25010-QualityTree.png

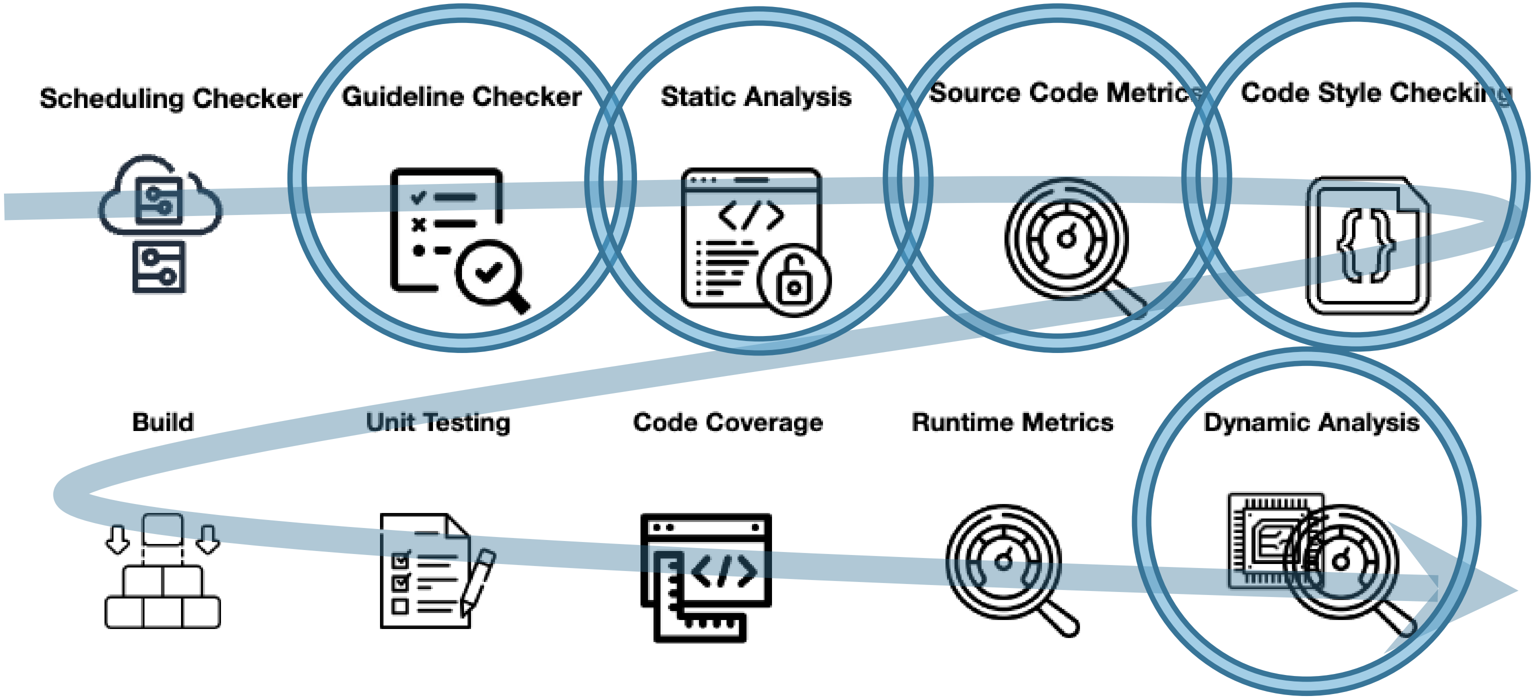
# Code quality and C++

- C++ is complex
- It is easy to write bad quality code with C++
- It is thus even more important to consider code quality during every stage of development
- How?
  - Follow a coding standard/conventions
    - Improves consistency and efficiency
  - Comment code
    - Makes code more readable
    - Document code changes and choices
  - Implement code reviews
  - Use tools whenever it is feasible (e.g. for static code analysis)

# C++ Core Guidelines

- https://isocpp.github.io/CppCoreGuidelines
- https://github.com/isocpp/CppCoreGuidelines
- A Bjarne Stroustrup’s initiative
- Use modern C++ effectively
  - C++ is continuously evolving (C++ 1.0, C++ 2.0, C++98, C++11, C++14, C++17, C++20)
  - Deals essentially with C++11 and beyond
  - High-level issues (interfaces, resource/memory management) for code that is statically type-safe, without resource leaks and programming logic errors
- Organized by chapters (Philosophy, Interface, Functions, …)
- Needs to be supported by tools for enforcement

# Ensure Continuous Checking



# Ensure continuous checking

- Different tools for different jobs
- Guideline checker
  - Enforce some dos and dont’s for writing C++ code: cpplint
- Static C++ code analysis:
  - Detect bugs with cppcheck, understand or clang-tidy
- Source code metrics
  - Use understand metrics for reducing complexity
- Code style checking:
  - Enforce common coding style with clang-format

# C++ Core Guidelines: an example

- https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Ri-global
- I.2: Avoid non-const global variables
  - Reason: Non-const global variables hide dependencies and make the dependencies subject to unpredictable changes.
  - Example: …
  - Alternative: …
  - Exception: …
  - Enforcement: …