Daimler ICertis - Development Practices

Executive Summary

NEEDS WORK

Recommend incorporating a Shift Left testing culture. The goal is to keep the main branch in an always shippable state. Quality will need to be baked into the pipeline. Investing in unit tests will allow quality to be baked in as part of every build and allow testing at the lowest level possible with the least amount of dependencies. Unit testing is not an exhaustive test for a service, but we should not choose to test something at a functional level if the test can be covered at a unit test level.

Current State

SCM and Branching

Icertis development currently follows a **main-only** branching strategy where developers pull code from the main branch to their local develop environment. Developers make their code modifications, test code, review, and push to the main branch.

Builds on the main branch are not triggered automatically on check-in.

Gates

Testing and Review of code is on the earnest of the development team (no enforced gates). Code does not contain unit tests

Unit Testing

Code Quality Reporting

Currently Icertis does not have SonarQube configured for Daimler builds due to issues when they initially tried to configure.

Target State

SCM and Branching

Gated Check-in to Master branch

Utilize Release Branching

When the main branch is ready for a release, create a release branch.

Gates

Every Check-in to main or release branches should be configured with **Gated Check-in** (requiring a successful build prior to check-in)

Check-In Policies should be configured to require:

- Successful build on check-in
- Code analysis run before check-in with code analysis check-in policies configured

Unit Testing

CI build should be triggered on every check-in to the main and release branches. The build definition should include:

- Execution of unit tests
- Enable code coverage

Unit testing rules:

- Should be fast. Full suite of automated unit tests should not take longer than 7 minutes. Each test should average under 100 milliseconds and no more than 2 seconds.
- Should be 100% reliable. It should have no external dependencies (exception for when SQL or file system may be required).

Recommend incorporating Build Quality Checks and incorporate Code Coverage Policy. This should be configured with **Failed Build on** *Previous Value*. This will fail the build drops below the previous build's coverage percentage.

Note: Do not configure **Code Coverage Policy** if **Test Impact Analysis** is also configured.

Code Quality Reporting

SonarQube is a great static code analysis tool that provides reports on code quality metrics and contains configurable code quality gates.

Currently Icertis does not have SonarQube configured for Daimler builds due to issues when they initially tried to configure.