1. Guidelines for M² Java Code Review

M² home

Goals of the Java code review:

- Perform static code analysis
- Ensure custom Java code meets a minimum level of standards

2. Toolkit

Here is a range of open source tool kits to help measure and achieve acceptable code quality

- sonarQube
- FindBugs
- Checkstyle

2.1. Java code review

There are several static code review tools available to use, it is important that the tool used correctly checks for the 7 axes of code quality, these are:

- duplications
- unit tests
- complexity
- potential bugs
- coding rules
- comments
- design and architecture

2.1.1. Severity of issues

There are five levels of severity of issues found:

- Blocker
  - This issue might make the whole application unstable in production

- Critical
  - This issue might lead to an unexpected behavior in production without impacting the integrity of the whole application
- Major
  - This issue might have a substantial impact on productivity
- Minor
  - This issue might have a potential and minor impact on productivity
- Info
  - Not known or yet well defined security risk or impact on productivity.

2.2. Minimal coding standards

There should be absolutely zero **Blocker** or **Critical** issues open when code analysis is performed. When it comes to managing major issues, if Code analysis has been performed throughout the development process, these should be at an absolute minimum.

2.2.1. Method complexity

Cyclomatic complexity is a way to determine if your code needs to be refactored. The code is analyzed and a complexity number is determined. Complexity is determine by branching (if statements, etc.) Complexity also might take in to account nesting of loops, etc. and other factors depending on the algorithm used.

Cycolomatic complexity ranges below are commonly used to assess code: [1: some methods may by design be complicated, so the values below are to be used not as law but as guidelines]:

- < 10 Easy to maintain
- 11-20 Harder to maintain
- 21+ Candidates for refactoring/redesign