|  |  |  |  |
| --- | --- | --- | --- |
| **NCOI Code Review Guideline** | | | |
| **Client:** | NCOI | **Project:** | NCOI Support & Maintenance |
| **Application:** | Olympus, e-Connect, EVC, Scheduler | **Period:** | 01.01.2012 – 01.02.2013 |
| **Project manager:** | Daniel Rucareanu | **Project leaders:** | Radu Opris, Adrian Pirlea |
| **QA Officer:** | Ionel Mihali | **Version:** | 0.1 |

**Introduction**

This document contains information about the aspects that need to be considered when performing a code review for the NCOI applications. This document is intended to all team members of the NCOI team, for both Olympus and e-Connect/EVC. The scope is to allow the code review to be performed by any team member, in an uniform way.

**Code review guidelines**

The code review guideline contains items that are identified during the review. The items are classified per area and have a recommended criticality level. The items will be categorize in ‘standard and conventions’, ‘maintainability’, ‘design’, ‘performance’. The last area, ‘functional’ is not included here, since the functional defects depend on the context. Any **functional finding** should have the criticality level **high** or **critical.**

The table below presents the items in the guideline.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Item** | **Area** | **Recommended criticality level** |
| 1 | Unused ‘usings’ | Maintainability | Low |
| 2 | Hardcoded values | Maintainability | Medium / High |
| 3 | Improper class or member name | Maintainability | Medium |
| 4 | Duplicated code in the same of different class or method/property | Maintainability | Medium |
| 5 | Redundant, unused or commented out code | Maintainability | Low |
| 6 | Method contains many conditions (IFs) | Maintainability | Medium |
| 7 | Files not placed in the correct project/folder | Maintainability | Medium |
| 8 | Too many database trips | Performance | High |
| 9 | Useless code cyles or recursion | Performance | High |
| 10 | Too much data retrieved from the database | Performance | Medium / High |
| 11 | Useless or avoidable casts | Performance | Medium |
| 12 | Method does more than one thing | Design | Medium |
| 13 | Use of implementation, instead of the interface | Design | Medium / High |
| 14 | Double cross layer dependencies / references | Design | High |
| 15 | Class cohesion (members belonging to the wrong class) | Design | High |
| 16 | Creation and/or usage of dependencies (concrete implementations) in unit-testable code. | Design | High |
| 17 | Methods with too many parameters (more than 4) | Design | Medium |
| 18 | Inappropriate member modifiers (static, private protected, public) | Design | Medium |
| 19 | Multiple code entities (classes, interfaces, structures) placed in a single file. | Standards and Conventions | Medium |
| 20 | Class members put outside or in the wrong regions | Standards and Conventions | Low |
| 21 | Classes, members, files, assemblies, namespaces not named according to conventions (depend on the project, context): prefix, order, casing, typos, etc. | Standards and Conventions | Low / Medium |
| 22 | Possible ‘null reference exception’ encountered | Functional | High / Critical |
| 23 | Missing functionalities or missing unit tests | Functional | High / Critical |