Code Quality Health Check

Pritesh Shetty[1], Swapnil Mirjolkar[2], Indira Bhattachariya[3]

V.E.S Institute of Technology
Approved by AICTE & Affiliated to Mumbai University
Hashu Advani Memorial Complex, Collector colony, Chembur-400074

Abstract— **This paper briefs about "QHC Report".**
QHC report refers to technique that measure and analyzes correctness, performance and maintainability that are involved in creating great code. These diagnostic tools can help you and your team to develop and sustain high standards of code excellence.

***Keywords* – QHC (Quality Health Check), Code Metrics, Code analyzer, JS Lint, W3C standard.**

1. INTRODUCTION

Code quality is not something that can be easily added later. Problems that are complex obscure and discovered late in the product cycle remains unfixed usually. The paper describes about principles and procedures for improving code quality

The definition of code quality is **LTFCE**, based on the notion that code is literature. Over its lifetime (maintenance, reuse, etc.) the code will be read many several times. So, good code is (*in order*)...

1. **L**egible - The code (the *code itself*, not comments) should clearly state the intent. In turn, reader can understand meaning and requirement for every statement of the code.
2. **T**estable - The code should be organized in a way that facilitates unit testing. That supports all subsequent efforts (refactoring for modification, correction of defects, revision due to changed specs, etc.)
3. **F**lexible - Dependencies, both on other code in the code base and arbitrary implementation choices, should be minimized. Hard-coded assumptions about data size, concrete classes or data structures, etc. make the code more brittle, and therefore harder to reuse or adapt.
4. **C**ompliant - The code should comply with its requirements, functional and otherwise.
5. **E**conomical - The code should make reasonable use of system resources: memory, CPU etc.

The point of insisting that these be considered in order is that each property supports the ones that follow. For example, defects in code which is legible, testable, and flexible can probably be corrected with reasonable effort. On the other hand, I suspect that every programmer has had experience with code which was micro-optimized for performance to the point that it was too brittle for reasonable maintenance.

Having established that, we then can see that the skill of the coder is really what we are looking for in this metric.

2. CODE QUALITY CHECKLIST

The General Code Review checklist and guidelines for Developers, which will be served as a reference point during development. This is to ensure that most of the General coding guidelines have been taken care of, while coding. Especially, it will be very helpful for entry-level and less experienced developers (0 to 3 years exp.) to refer this checklist until it becomes a habitual practice for them

Unit Testing:

A developer must have unit tests for all code changes. It's a peer code review best practice to also review unit tests and ensure all scenarios are covered.

- Must have pre-requisite for starting code review.

- Use code coverage tools to ensure completeness of tests.

Peer Code Review Best Practices

ISSN: 2278 – 7798

Some of these peer code review best practices are already achieved during agile code review and pair programming. Peer reviews in software help reduce late identification and resolution of bugs. Below mentioned practices will help you drive effectiveness in code review.

Code Documentation

Writing code comments is helpful however not always necessary. Sometimes the code is not self explanatory, or does something un-usual. It needs to be well documented.

Sticking To Coding Standards

Software needs to follow consistent coding practices. There are coding standards available in almost any programming language. Stick to the standard coding practices to ensure everyone speaks the same language. It makes it easier for a new developer to start understanding the code faster.

Avoid Hard Coding - Prefer Configuration

Many configurations are often hard coded by developers inside the compiled code and make it difficult to change the behavior of software. For example an optional feature should have a configuration that can enable or disable a feature. I code reviewer my look for such opportunities to ensure configuration is preferred over hard coding.

Code Performance

Performance of the code is important, however difficult to measure depending on type of software. It's important that a code reviewer looks at the performance of a code in various aspects.

- Time to do the task - How much time it takes to finish the task. Is it acceptable?
- Memory or Number of objects used to do the task. Is it acceptable?
- Number of threads or processes to do the task. Is it acceptable?
- Concurrency limits of the task. How many tasks can be performed by the program concurrently

Encourage Code Reusability

Copy/Paste is developer's best friend. Many lines of code are duplicated by simply doing a copy paste. This seems to solve problem in short term however creates maintenance issues in long run. Any copy/paste candidate in code must be refactored to be re-usable. It needs to be done carefully since many other code blocks may be calling it already.

Code Security Review

This is most difficult step in code review since not many people know how to write secure code. Some common guidelines are listed below for a secure code

- Encrypt sensitive user information (e.g. SSN, passwords, credit card numbers etc)
- Never log sensitive information on log files, it can be dangerous and make hacking easy.
- Disable Weak Ciphers on your app/web servers.
- Use trusted libraries for cryptography and encryption instead of inventing your own.
- Top 10 issues are addressed in a web facing application.

Eliminate Unwanted Code and Libraries If Not Used

Its common habit of developers to not remove unused code from a file. This is mainly since they think it may be required in future. Some developers tend to comment it so it can be used later. This makes the code unreadable. The code reviewer must make sure that all such code is removed. Developers can always go back to a previous version of code in version control. However, in my observation, a commented code is almost always not reused.

Logging Key Attributes For Debugging

Debugging information is important to support and maintain applications. Logging required key attributes makes the job of dev ops and support team easy. Developer and Code reviewer needs to ensure that all key attributes are being logged.

ISSN: 2278 – 7798

Code Indentation and Readability

All developers in a team needs to use common formatting tool with same formatting styles to avoid version control conflicts and difficulty in reading file changes. There is no right or wrong way to format code, however make sure the team uses the same formatting.

3. LITERATURE STUDY

Code quality is a loose approximation of how long-term useful and long-term maintainable the code is. Code that is not used tomorrow: **Low quality**.

Code that is being carried over from product to product, developed further, maybe even opens sourced after establishing its value: **High quality**.

Since looking into the future can be somewhat tricky, we look at the present signs that may help predicting it.

So at different levels code quality has to be filtered by using following quality tools:

1) Code Analyzer
2) Code Metrics
3) JS Lint
4) HTML validator
5) CSS validator

1) **Code Analyzer:**

Code analysis for managed code analyzes managed assemblies and reports information about the assemblies, such as violations of the programming and design rules set forth in the Microsoft .NET Framework Design Guidelines

The analysis tool represents the checks it performs during an analysis as warning messages. Warning messages identify any relevant programming and design issues and, when it is possible, supply information about how to fix the problem.

Team members run code analysis on their development computers. In Visual Studio, developers configure and run code analysis runs for individual code projects, view and analyze issues found by the runs, and create work items for warnings.

2) **Code Metrics**:

There are various metrics that can be calculated for source code. These can help to determine its complexity, readability or maintainability. When you are involved in code reviews, it can be useful to determine such metrics for the code being examined. This can help you to identify problem areas and rectify issues before they become too embedded in the software

The following list shows the code metrics results that Visual Studio calculates:

- **Cyclomatic Complexity** – It is a measure of *structural* complexity. It is related to the number of possible routes through a unit of code. A simple method with no branching cause by if or switch statements will have a very low cyclomatic complexity. Each additional conditional processing command increases the complexity and decreases the code's readability.

- **Lines of Code –** Indicates the approximate number of lines in the code. The count is based on the IL code and is therefore not the exact number of lines in the source code file. A very high count might indicate that a type or method is trying to do too much work and should be split up. It might also indicate that the type or method might be hard to maintain

- **Depth of Inheritance –** It describes the number of classes from which a specific type inherits functionality. The idea is that if more types exist in an inheritance hierarchy, the code will likely be more difficult to maintain as a result. However, a high depth of inheritance can also indicate a greater level of code reuse. This means that it is difficult to say what a good depth is.

- **Class Coupling** – Measures the coupling to unique classes through parameters, local variables, return types, method calls, generic or template instantiations, base classes, interface implementations, fields defined on external types, and attribute decoration. Good software design dictates that types and methods should have high

ISSN: 2278 – 7798

cohesion and low coupling. High coupling indicates a design that is difficult to reuse and maintain because of its many interdependencies on other types.

- **Maintainability Index** – Calculates an index value between 0 and 100 that represents the relative ease of maintaining the code. A high value means better maintainability. Color coded ratings can be used to quickly identify trouble spots in your code. A green rating is between 20 and 100 and indicates that the code has good maintainability. A yellow rating is between 10 and 19 and indicates that the code is moderately maintainable. A red rating is a rating between 0 and 9 and indicates low maintainability.

3) **JS Lint:**

JSLint is a static code analysis tool used in software development for checking if a JavaScript source code complies with coding rules. It is provided primarily as an online tool, but there are also command-line adaptations.

4) **HTML Validator:**

An HTML validator is a quality assurance program used to check Hypertext Markup Language (HTML) markup elements for syntax errors validator can be useful tool for HTML user who received data electronically from a variety of input sources.

Syntax error such as open tags, extra spaces, or forgotten quotation marks, can cause a web page to look drastically different than creator intended or render correctly in one browser, but not in another.

Verifying the integrity of HTML elements manually is repetitive and tedious work and it becomes difficult when different set of rules such as CSS and XML are included in the page.

By using validation program to systematically flag errors, the HTML user can choose whether to correct the code mistake on case by case basis or correct them globally by using find and replace mechanism within the program.

5) **CSS Validator:**

CSS is an evolving language, and it is considered by many that "CSS" is a single grammar with a number of properties and acceptable values defined in various profiles. In future version of validator, the default behavior may be to check style sheet against the latest "CSS grammar" and the cloud of all CSS properties and values.

The W3C CSS validator is written in java language. You can browse the code online and follow the instruction to get the CSS file validated

Code Quality Health Report:

Quality Metrics	Number of Warning	Number of errors
Code Analyzer	0	0
Code Metrics	0	0
JS Lint	0	0
HTML Validator	0	0
CSS Validator	0	0

The above statistics state that the code has maintained a code quality standard with **"0" warning and "0" errors**

4. ALGORITHM AND FLOWCHART

Step1: Start

Step 2: Get the project code path on which the filter or code quality tool is to be run

ISSN: 2278 – 7798

Step 3: Code Analyzer scan the code on basis of code quality standard and generated a XML file on the drop Location

Step 4: Code metrics scan the code on basis of code metric parameters and generate a XML file on the drop Location

Step 5: JS Lint scans the code as per JavaScript standards and generated the excel file stating list of warnings and error

Step 6: **A consolidated table structure text is generated stating number of warning and error under each quality tool**

Step 7: The Mailer operation will attach a consolidated table structure in mail body and gets all attachment from drop location for user reference

Step 8: Mailer function will send mail to respective user

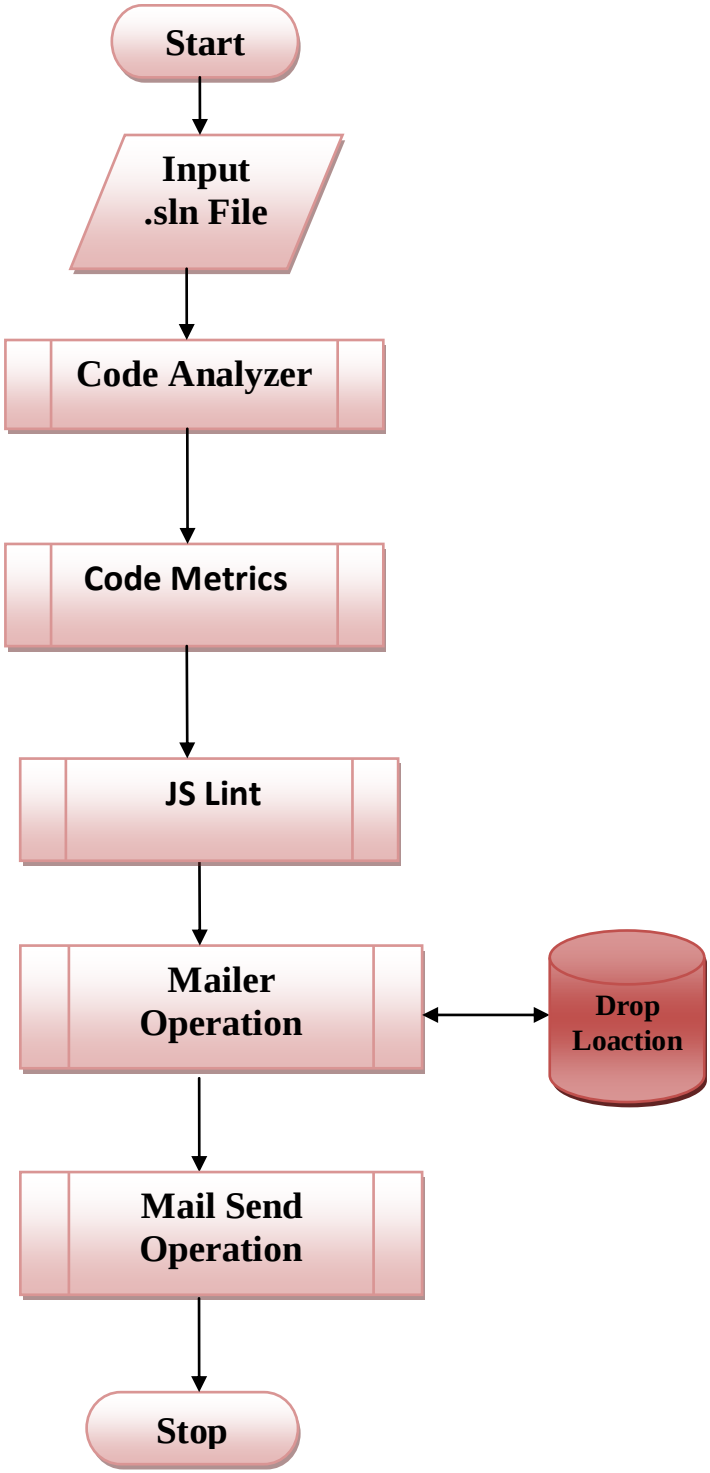

Fig 3 Flowchart of the Project

ISSN: 2278 – 7798

5. CONCULSION

Code Quality Assurance technologies have been associated generally with very costly top applications. Today the core technologies have evolved and the cost of equipments is going down dramatically due to the integration and the increasing processing power.

Reviewing each other's code is a necessary step in any modern development team's workflow. Just like a book author wouldn't publish a book without an editor's review, a developer should never release work without having it reviewed first.
Code Review with Beanstalk is built to make this process seamless and encouraging. Your team has a common goal; get your work out to your users on time and bug-free. Having a solid code review process is the most important step to get that accomplished.

6. FUTURE ENHANCEMENTS

a) Further version will more focus on getting HTTP Request for HTML and CSS Validator

b) Improvised Approach for setting the path of solution file

c) Service level implementation for user interaction with the help of GUI

d) For any organization where number of projects are implemented we can use a SharePoint list to get the project path and remove App.config dependency

7. REFERENCES

[1] Code quality Open source perspective ,Diomidis Spinellis
[2] A Tutorial onCode Quality tool, Lindsay Smith.
[3] http://blog.smartbear.com/peer-review/defining-code-quality/
[4] http://www.wikipedia.org/
[5] http://www.codeproject.com/

ISSN: 2278 – 7798