Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 70 (2014) 24211-24214

ISSN: 2229-712X

Academic Coding Guideline Model OCG - Open Coding Guidelines For Windows (An Organized Code Creation Model)

Hareesh Gu and Sanjay Kumar Dubey
Amity University, Sector 125, Noida, India.

ARTICLE INFO

Article history:
Received: 6 December 2013;
Received in revised form:
25 April 2014;
Accepted: 10 May 2014;

Keywords
Academic coding guidelines,
Source code reuse,
Writing clean code,
OCG.

ABSTRACT
Every software company has its own set of guidelines that increases code readability, reuse and helps in its storage in organized storage structure .These guidelines also help in the determination of various attributes of code such as complexity, duplicity, warnings, memory leaks, coverage etc. However coding guidelines/approaches as such are not present in the field of academics. Scholars of educational institutions generally concentrate mostly on logic (brute force quality approach) while coding, against writing documentable clean code or understanding code's attributes. Also end evaluation/delivery in a software company is dependent on clean code with various attributes such scenario is not possible in academic framework all together and mostly is done manually. The paper tries to bridge this gap by analyzing the role of an organized code creation model and its effect on various stakeholders in academic Eco-system and suggesting one such open source Code Guideline Model OCG and providing its implementation through a template.

Introduction

Coding conventions/Guidelines are a set of guidelines for a specific programming language that recommend programming style, practices and methods for each aspect of a piece program written in a particular language. These conventions usually cover the following file organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices, programming principles, programming rules of thumb, architectural best practices, Folder storage hierarchy, tools for analysis of code parameters etc[9].

Coding conventions are for software structural quality and are not enforced by compilers [1].Software programmers are highly recommended to follow these guidelines to help improve the readability of their source code, make software maintenance easier and help in easier code delivery [1to4].

Fig. 1. Unclean Code Injection Sectors Pie Chart

When CG's are not followed it leads to development of poor quality code whose attributes are questionable or unknown .The whole software thus written can be classified as brute force quality developed. Every software company has it own set of guidelines that increases code readability, reuse and helps in its storage in organized storage structure. These guidelines also help in determination of various attributes of code such as complexity, duplicity, warnings, memory leaks, coverage etc. However coding guidelines/approaches as such are not present in the field of academics. Data collected from various sources suggest code written for academic purposes like projects, practice, examination etc have the lowest quality against code written in industries for product and services development.

Fig. 2. Sample of code written with its un-organized storage structure for project purpose in academic environment

The paper would attempts to bridge this particular gap, the paper has two aspects primarily the paper summarizes a set of best practices in a open template format called OCG finally the paper proposes organized code development as a change in thought process in academic environment and does a stakeholder effect analysis when Coding Guidelines are used.

Proposed Model OCG

The proposed model is in the form of a template .OCG template incorporates Various basic industrial approved aspects of organized code writing like (File organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices, programming principles, programming rules of thumb, architectural best practices) along with some advanced concepts such as evaluating complexity, Duplicity ,Basic Code coverage using an open source Continuous Inspection framework (sonar).

Tele:
E-mail addresses: hareeshg.pillai@gmail.com

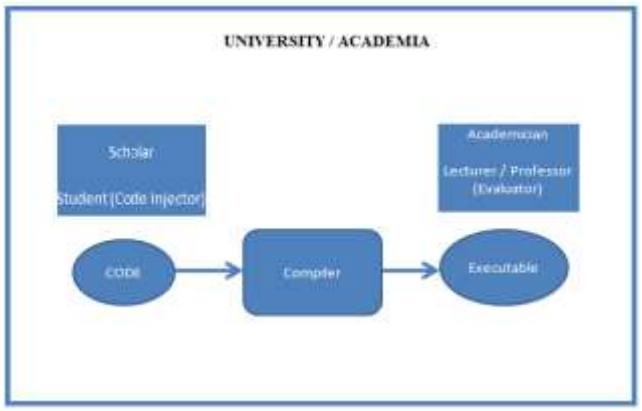

Fig. 3. Current source code injection process in academic eco-system

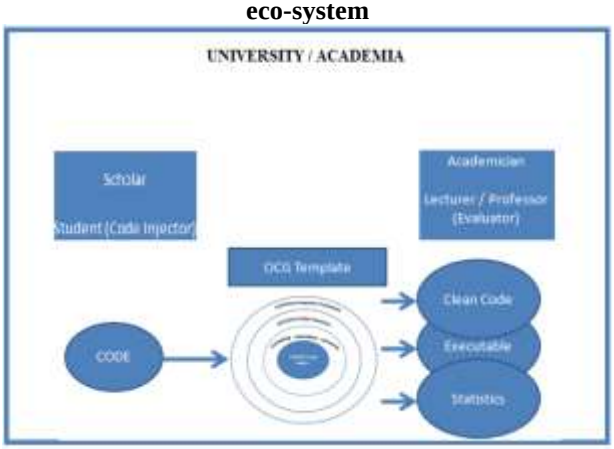

Fig. 4. Proposed source code injection process in academic eco-system

The proposed model takes 3 basic assumptions on platform and language at the moment.

1. Operating system – windows
2. Programming language – c
3. Compiler being using – Turbo / Borland (efforts would be made to overcome these assumptions in the later versions of model). OCG template has 3 layers which helps in making code cleaner stored in a traceable folder hierarchy and helps to qualify code aspects. Using the template in the proposed model is very hassle free as the scholar now can inject code in the proposed template directly.

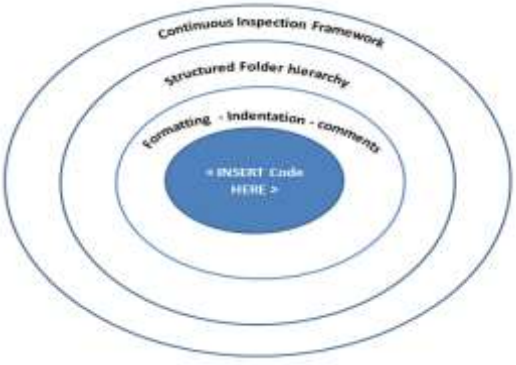

Fig. 5. OCG template graphical representation

OCG template contains 3 working areas Formatting, indentation and commenting related best practices incorporation.

In computer programming, a comment is a programming language construct used to embed programmer-readable annotations in the source code of a computer program. Those annotations are potentially significant to programmers but typically ignorable to compilers and interpreters. Comments are usually added with the purpose of making the source code easier to understand. [9&11] The syntax and rules for comments vary and are usually defined in a programming language specification. In OCG template many commenting techniques were analyzed and the below commenting techniques are embedded on template. The selection of the below commenting style is based on the amount of accuracy, clarity and brevity of information the style helps to convey.

Header comments

```
/*************************************************************************
*   Filename    : header.h
* Description contains all header and macro related information
*   Date   Name
Reference
    Reason
*<header_creation_date<name_of_creator>
<reference><reasons_if_any>
**      Copy Right Information
*************************************************************************/
```

Program level comment styling

```
/*************************************************************************
*   Filename    : <program_name.c>
* Description : <program_name.c> has abc  functionality
*Input       :  <over all program inputs eg. Any command line arguments etc >
*Output  :  <end result of the .c file>
*   date   name  reference
    reason
*<header_creation_date<name_of_creator>
<reference><reasons_if_any>
**      Copy Right Information
*************************************************************************/
```

Function level comment styling

```
/*************************************************************************
  *   * Function name  :      Fuction_name()
  *Description   :      <what the function does >
  * Input : <input information for the function eg. Character array>
  * Returns   <return information of  function eg. void  >

*************************************************************************/
```

Variable level or individual block level

```
/************************  <insert  comment  here  > ***************/
```

Horstmann style

```
while (x == y)
{    something();
     somethingelse();
     //...
     if (x < 0)
     {    printf("Negative");
          negative(x);
     }
     else
     {    printf("Non-negative");
          nonnegative(x);
     }
}
finalthing();
```

Fig. 6. Snap shot of code written in Horstmann style

In computer programming, an indent style is a convention governing the indentation of blocks of code to convey the program's structure. Indentation is used to show the relationship between control flow constructs such as conditions or loops and code contained within and outside them[13]. There are various styles of indentations available.OCG models template incorporates Horstmann style by Cay S. Horstmann adapts Allman by placing the first statement of a block on the same line as the opening brace.

Example

Folder hierarchy being followed in OCG template

A Folder is an organizational unit, or container, used to organize sub folders and files into a hierarchical structure. Having an organized folder hierarchy helps in developing and maintaining cleaner code and helps in easier storage and deployment[3to5].

Proposed folder hierarchy in OCG template is as follows.

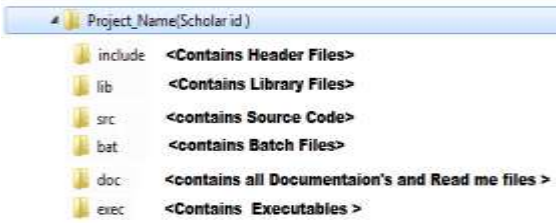

Fig. 7. Graphical Representation Of Proposed Folder Hierarchy

Program_name_<scholar_id> EG. Bubble_sort<13086>
Include -> Contains header files and all files to be included.
Lib -> Contains Library files.
Src -> Cotains source files (.c etc)
Bat -> Starter file (equivalent to make utility in linux).
Exec ->Contains final executable file.
Doc ->Contains Documentation, Read me files etc

Continuous Inspection Tool integration

Continuous Inspection requires tools to automate data collection, to report on measures such as complexity, duplicity, warnings, memory leaks, coverage etc. and to highlight hot spots and defects. Sonar is currently the leading "all-in-one" Open source Continuous Inspection engine. A Continuous Inspection engine can be seen as an Information Radiator dedicated to make the source code quality information available at anytime to every stakeholder [14]. OCG's template's outer layer uses Sonar CI web interface to run diagnostics on injected code within the template. Since Sonar is a web based (runs on http://) Continuous Inspection platform with support for more than 40 programming languages via plug-ins sonar tools incorporation makes OCG a platform, device and programming neutral model. Sonar helps to generate real time diagnostics on every line of code injection (complexity, duplicity, warnings, memory leaks, coverage)there by qualifying source code injected.

Fig. 8. Sonar Dash Board Snap Shot Overview

Fig. 9. Sonar Dash Board Snap Shot showing individual code attributes

Fig. 10. Sonar Dash Board Snap Shot Showing individual code attributes

Algorithm For Template Usage

1. For each Source_file_inserted_in_OCG_template
2. Open OCG template
3. Add necessary header files in INCLUDE folder
4. Add necessary library files in LIB folder
5. Add source code to .C template file SRC folder
6. Add documentation related to code inserted in step5 to DOC folder
7. Add source code's main function file references and compiler related configuration information to .bat file in BAT folder
8. Execute bat file for testing
9. Logon to sonar dashboard to see detailed attributes of injected source code in step 5
10. output
11. End

Stake Holders In Academic Eco-System

In any academic framework the major stake holders are student(scholar), academician (teacher), academia (university) and in a broader sense it can be analyzed that coding done in this context falls under one of these purpose Practice or Project or Exam

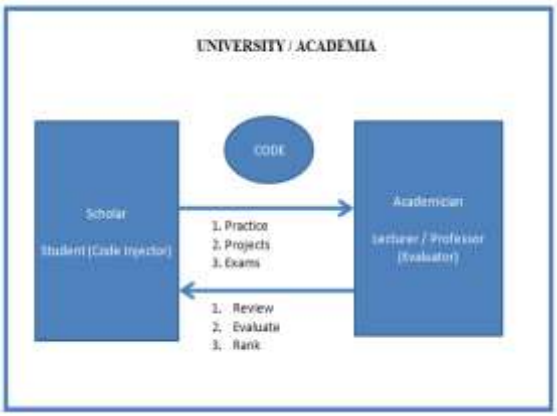

Fig. 11. Academic Eco-system Graphical Representation

The broader effects of using an organized coding guideline model on the above classified stake holder are as follows

Scholar (Code injector) level IMPACT

Scholars using model would have better understanding of organized code writing, understanding of various elements /attributes of code. The model could create a competitive environment creation for better quality code creation among scholars.

Academician (Evaluator) level IMPACT

Code manually evaluated/reviewed initially could now be evaluated in an automated manner by academician. The template also provides ample automated review support

Academia /University (Environment) level IMPACT

The model when used in academia would empower the university (Higher management) to get its evaluation process certified against various available models/quality metrics like CMM ,TQM, Sigma etc. The university also gets a view of practical Empowerment being created or skills being taught in a 360degree graphical view which was earlier done manually or done using feedback form collections.

Advantages of proposed model

The model being proposed in the form of a template helps in creation of structured code (clean code). The model helps scholars in better understanding of code being injected and their corresponding attributes in a better manner using minimum extra effort . It helps the academicians in code analysis in a holistic manner and empowers than in better ranking and automates the evaluation process. The model helps the academia to achieve better visibility of the testing process being followed it also provides a platform for quality certifications of evaluation. since the model is open source in nature academic institutions can extend the model (add delete modify) models structure create their own variant and try field testing models efficiency.

Fig. 12. Model Extensibility Diagram

Conclusion

The model proposed is a change in thought process in the field of academia Stake holder effects and ways of achieving it are outlined in the paper. Further clarity on the proposal can be achieved by field testing the model.

Future improvements

The model needs to be field tested with the basic assumption set and real time bugs have to be documented and fixed. Apart from it effort can be made to overcome these basic assumptions and model can be made both operating platform and programming language neutral. Finally the models alignment with IDE (Integrated Development Environment) builders or RAD (Rapid Application Development) tools are unknown .These effects need to be analyzed and documented

References

[1] "Code Conventions for the Java Programming Language : Why Have Code Conventions". Sun Microsystems, Inc. 1999-04-20.
[2] Robert L. Glass: Facts and Fallacies of Software Engineering; Addison Wesley, 2003.
[3] Staplin, George Peter (2006-07-16). "Why can I not start a new line before a brace group". 'the Tcler's Wiki'.
[4] "C Programming FAQs: Frequently Asked Questions". Addison-Wesley, 1995. Nov. 2010.
[5] "Why have trailing commas in resources?", Puppet Cookbook, Dean Wilson
[6] http://www.mono-project.com/Coding_Guidelines
[7] Hoff, Todd (2007-01-09). "C++ Coding Standard : Naming Class Files".
[8] https://www.haiku-os.org/development/coding-guidelines
[9] http://en.wikipedia.org/wiki/Coding_conventions
[10] http://en.wikipedia.org/wiki/Programming_style
[11] http://en.wikipedia.org/wiki/GNU_coding_standards
[12] http://www.methodsandtools.com/PDF/mt201001.pdf
[13] http://en.wikipedia.org/wiki/Indent_style
[14] http://en.wikipedia.org/wiki/SonarQube