**Coding Review Document**

**Reviewer Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Code Title and Author: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Source Code reviewed (number of files): \_\_\_\_\_\_\_\_\_\_**

This is a checklist that is prepared by the instructor of 3150 to assist you in the peer review process.

Coding Style Review[[1]](#footnote-1)

# Documentation:

\_\_\_\_ Comment block exists at the beginning of the source file containing at least the

 following information: original author’s name, file creation date, development group, and a brief statement of the purpose of the software in the file.

\_\_\_\_ Each subroutine or function in the file is preceded by a comment block which provides the following information: routine name, original author’s name,

routine’s creation date, purpose of the routine, a list of the calling arguments

(their types and what they do), a list of required files and/or database tables, the routine’s return value, error codes and exceptions processed by the routine, and a modification history indicating when and by whom changes were made.

# Programming Standards

\_\_\_\_ Consistent indentation of at least 3 spaces is used to distinguish conditional or control blocks of code. TABS NOT USED FOR INDENTATION.

\_\_\_\_ Inline comments are frequently used and adequately describe the code.

\_\_\_\_ Structured programming techniques are adhered to.

\_\_\_\_ Subroutines, functions, and methods are reasonably sized.

\_\_\_\_ The routines in each source file shall have a common purpose or have interrelated functionality. Methods in a class support its functionality.

\_\_\_\_ The name of the file, script or class represents its function.

\_\_\_\_ Function and method names contain a verb, that is, they indicate an action.

\_\_\_\_ Meaningful variable names are used.

\_\_\_\_ Variables are initialized prior to use.

\_\_\_\_\_ Braces are used consistently

\_\_\_\_\_ There are no compiler warnings

# Programming Guidelines

\_\_\_\_ Spacing is used correctly to enhance the source code’s readability.

\_\_\_\_ When continuing lines of code on new lines, they are broken after a comma or an operator. Higher level breaks are used instead of lower level breaks.

\_\_\_\_ Wrapped lines of code are aligned with the beginning of the expression at the same level on the previous line.

\_\_\_\_ Program statements are limited to one per line.

\_\_\_\_ Nested program statements are avoided.

\_\_\_\_ Parentheses are used to remove operator precedence ambiguity and to improve code readability.

\_\_\_\_ Inline comments constitute approximately 20% of the total lines of code in the program, excluding the file and routine documentation blocks.

\_\_\_\_ The software reflects a balance of coding for efficiency and coding for readability.

\_\_\_\_ Meaningful error messages are used.

\_\_\_\_ System calls which acquire system resources are tested for error returns.

\_\_\_\_ Routines and methods contain no more than 200 executable lines.

\_\_\_\_ The number of routines in a source file is kept to a minimum for procedural languages.

# Any other Comments on style:

Coding Design Review[[2]](#footnote-2)

# Performance:

Comment on the time required to respond to stimuli (user or internal events).

*Typical Design/Architectural principles to look for:*

* Connection pooling - reducing the execution time overhead associated with establishing database connections by establishing a shared pool of connections
* Load balancing – spreading the load evenly between a set of resources
* Distributed processing
* Caching – using a local copy of data to reduce access time
* Lazy instantiation
* Transaction Concurrency
* Process isolation between OLTP and OLAP
* Replication of data

*Typical unit of measurement you could use:*

* Transactions per unit time
* Amount of time it takes to complete a transaction

# Reliability:

The ability of the system to keep operating over time in the context of application and system errors and in situations of unexpected or incorrect usage (to perform in a predictable manner).

*Typical unit of measurement you could use:*

* Mean time to failure

# Modifiability:

The ability to make changes to the system quickly and effectively.

*Typical Design/Architectural principles:*

* Independence of interface from implementation – This mechanism allows architects to substitute different implementations for the same functionality.
* Separation – This strategy separates data and function that address different concerns. Since the concerns are separate, we can modify one concern independently of another. Isolating common function is another example of a separation strategy. (High cohesion).

*Typical unit of measurement:*

* Using specific changes as benchmarks and recording how expensive those changes are to make

# Functionality:

The ability to the system to do the work it is designed to do.

*Typical unit of measurement:*

* Number change required

# Extensibility:

The ability of the system to handle new feature implementation/replacement of components with improved versions and the removal of unwanted or unnecessary features or components.

*Typical unit of measurement:*

* Easy, incremental addition of functionality (time, budget, etc.)
* Coupling/cohesion

# Maintainability:

The ability to quickly identify problems and fixing them within the system quickly

*Typical unit of measurement:*

* Easy localization
* Readability of the code
* Understandability of the code
* Ripple effects of change

# Reusability:

The ability to use components of the software or system within the system and outside.

Additional comments

Any other additional comments that the reviewer feels need to be addressed.

1. These review checklists were borrowed from OHD General Programming Standards and Guidelines Peer Review Checklist [↑](#footnote-ref-1)
2. These review checklists were borrowed from http://www.codeproject.com/KB/architecture/SWArchitectureReview.aspx [↑](#footnote-ref-2)