Software Code Review Template

Version: Draft

131 N High St, Suite 640
Columbus, OH – 43215

Email
rfp@devcare.com

Phone
614-221-2277

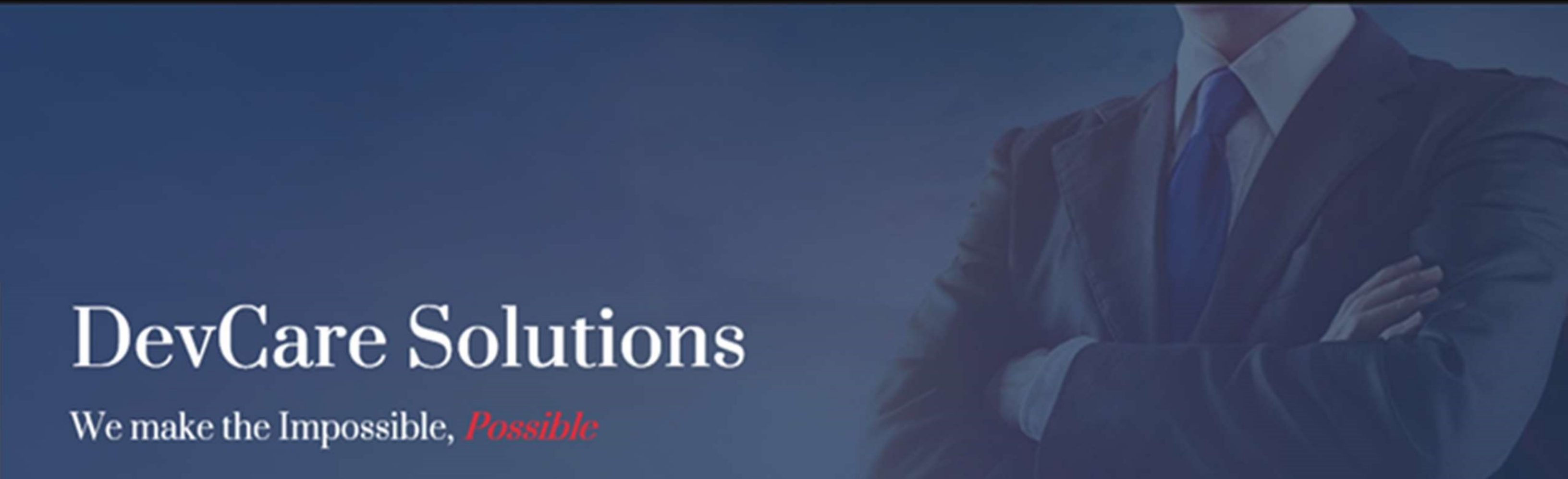

Table of Contents

Scope

The main scope of the document.

Purpose

Specific reason (new feature, bug fix, etc.)

Implementation

Source Code

Security

Stages: The overall code review process is divided into three stages.

Stage	Name	Purpose
1	Code Preparation	Preparer ensures that code adheres to **code review checklist.** Makes non-functional changes as necessary. Notes any functional changes that should be made to adhere to checklist. Pays particular attention to including all items required for header blocks and adding helpful comments.
2	Off-line Code Review	Individually or in small groups as desired, reviewers review the code for all items on the **code review checklist,** covering the areas of proper operation, adherence to coding guidelines, and implementation of risk (and safety hazard) mitigation actions. Reviewers recommend changes.
3	Formal Code Review Meeting	Participants review suggested changes from the off-line code reviews, decide actions, approve code. • The **code review checklists** filled out during off-line review, and any recommended changes to code, will be analyzed. • The **code review record** will be filled out to document any actions, including required changes to the code, and the approval of the code. • The **code review record** and attached **code review checklists** will be filed as the official record of the code review.

Definitions: The following definitions apply.

Code Review	The formal review of software units or modules.
a Unit	One function or routine, starting from its comment header block, to the last line of code or comment in the unit
a Module	The logical grouping of a set of units and its data structures. Normally a module will consist of one or more '.C' source files and it's associated '.H' files.
the Presenter	The person who authored the module or unit(s) for code review.
the Reviewers	The two or more people who are reviewing the module or unit(s)
SRS	Software Requirements Specification – document containing the user/system level requirements this piece of software must fulfill
SDS	Software Design Specification - document containing specific information about the high level design of the Unit or Module being reviewed.

Instructions for Code Preparation

STEP	DESCRIPTION
Check out file to be prepared	The relevant module and any associated header files should be formally 'checked out' of the version control or software code control system for any changes to take place. The most recent checked-in or promoted revision should be used.
Make changes to each unit/module:	In order to prevent unintentional change of functionality, no changes to the actual body of code or its structure should be made prior to a first pass review. Changes which do not affect the functionality (such as the addition of comments) can be made.
Required: ***Add code header***	See **code review checklist** in next section for standards each header must follow. The general categories of information to be included are: • Comments • Data usage (input/output parameters, data structures accessed, changes to global variables, etc.) • Unit processing algorithm/design • Potential failure modes related to system hazards to the patient; required mitigation actions • Miscellaneous notes on critical or risky areas of the design; design assumptions; etc.
Recommended: comment related units	If, when preparing just one unit from a module, it is found necessary to analyze other un-reviewed units, the addition of comments should be made, and any helpful information that could be of future use in the review and/or test of those additional units should be recorded.
Required: Document recommended changes to actual code	Recommended changes may be prepared on a copy of the module, and presented for review. Both the original and the copy with differences clearly indicated should be provided.

Run Lint	Lint must be run on the module or unit(s). Each warning or message produced should be inspected and any real issues corrected or flagged for correction See the **code review checklist** on the following pages for a list of the items Lint must be used to detect. Global wrap-up' output can be discarded and ignored for code review. The final Lint output will be recorded as part of the formal review meeting.
Check in files	The updated module(s) should be checked back into the official version control or software code control system
Prepare diagrams and other supporting information	Data flow, state diagrams or any other useful descriptive information should be prepared to present along with the code for review. This information may be added to the SDS after the review.

Code Review Checklist: Modules (P. 1 Of 1)

Use of this form:

Code Preparation:	Use this checklist as a guideline for preparing the module
Off-line Code Review:	The items on this checklist should be reviewed during Off-line Code Review.
Formal Code Review Meeting:	This form, filled out during the Off-line Code Review, should be brought to the formal meeting, used as a checklist for verification of all items, and attached to the code review record.

CATEGORY	ITEM	PRESENT? Y, N, N/A
Module header	The module must have a module header block containing:	
	File name:	
	Original creator:	
	Date created:	
	Person who last changed code (if different from creator)	
	Code revision number and change history (with dates). NOTE: The name of each changed unit should be listed NOTE: If a change is made to correct a defect, the number or ID of the defect corrected should be entered as well.	
	High level description: (explain the module's purpose, and the name/purpose of key data structures, variables, sub-functions used, etc.)	
	Failure modes and effects analysis: List types of failures which could occur in this module and result in a hazard to the patient. List the types of mitigation actions the software takes to prevent hazards from occurring. If these risks are documented in a separate document, reference it. (Editor's note: for non-medical projects, the corollary would simply be an appropriate analysis of possible software failure modes, and what the software should do in each case.)	

Module definitions and declarations	**Grouping:** Definitions and declarations should be separated into distinct groups, each with a comment header. For example, #defines, #includes, constant definitions, local function prototypes, etc. would all be grouped separately. If required for greater logical clarity, however, related definitions and declarations may be mixed	
	Commenting: Each definition or declaration should have an associated descriptive comment unless the declaration is really obvious.	

CODE REVIEW CHECKLIST: Units (p. 1 of 2)

Use of this form:

Code Preparation:	Use this checklist as a guideline for preparing each unit in the module
Off-line Code Review:	The items on the checklist should be reviewed during Off-line Code Review.
Formal Code Review Meeting:	This form, filled out during the Off-line Code Review, should be brought to the formal meeting, used as a checklist for verification of all items, and attached to the code review record.

CATEGORY	ITEM	PRESENT? Y, N, N/A
Function header block	Every function (Unit) must have a comment header block containing:	
	Function name	
	Change history: List of each change to the unit, with the date of the change and the name of the person making the change. Reference defect numbers or ID if the change was to correct a defect.	
	Purpose: A short description of the unit's purpose. The description should be written such that the unit's purpose in fulfilling the original software requirements in the SRS can be understood.	
	I/O description: A description of the inputs and outputs expected, specifying their acceptable ranges.	
	Return value: A description of the return value	
	External variables: A description of any external variables used, specifying acceptable ranges.	
	Unit design/algorithm: A more detailed description of the unit's processing. Should be detailed enough that reviewers can determine whether the code meets its design, but not so detailed that the description is just pseudo-code.	
	Failure modes analysis: A list of possible failure modes resulting in hazards or error conditions, and any mitigation actions this unit is required to take (for example, range checking a data value before use.)	

Lint results	As noted in the Stage 1 Preparation instructions, Lint should have been run on the module or unit(s). The final Lint output should be recorded as part of the formal review meeting. Each warning or message produced by Lint should have been inspected and any issues corrected. The items listed below must be checked for. 'Global wrap-up' output can be discarded and ignored for code review.	
	• Loop index not modified within the loop	
	• No extraneous code exists	
	• All data references defined, computed, or obtained from external source.	
	• All defined and referenced calling sequence parameters agree.	

CODE REVIEW CHECKLIST: Units (p. 2 of 2)

CATEGORY	ITEM	PRESENT? Y, N, N/A
Code Checks		
	• Descriptive comments are accurate and informative.	
	• Return values (in particular error returns) are not ignored.	
	• Constants and literals are not hard coded.	
	• All variables used have obvious or descriptive names, and correct scope.	
	• Local functions and non-automatic variables are declared static.	
	• System global functions have the module name as a prefix to the unit name.	
	• All functions have prototypes (compiler checks this).	
	• Data structure fields are described and commented clearly.	
	• Code is logically correct (Code performs intended functions, operates correctly)	
	• Numerical methods are sufficient	
	• Accuracy of control outputs to external devices are within tolerance	
	• System I/O mechanisms are consistently used.	
	• Standard module communication techniques are used (e.g. use of message system)	
	• Errors are detected and handled, and processing continued	
	• Error handling conventions are followed (standard use of error handling task, etc.)	
	• Input values (or other data used) are checked for reasonableness before use	
	• Where necessary, critical output parameters or data are checked for reasonableness during processing	
	• Code pays attention to recovery from potential hardware faults (e.g. arithmetic faults, power failure, clock).	

	• Code pays attention to recovery from device errors.	
	• There is no redundant code.	
	• The structure is clean and indentations correct.	
	• Over complication is avoided.	
SDS Check	SDS (Software Design Specification) info for this unit is accurate	

FORMAL CODE REVIEW RECORD (page 1 of 2)

Fill in the following information as the formal record of the review:

Date and time of review	
Presenter	
Reviewers	
Module (or unit) under review	
Module (or unit) revision number reviewed.	
Supporting information used in review	__ SDS __ Diff listings for suggested changes __ New diagrams ___ Code Review checklists __ Lint output ___ Other ___________

Notes on recommended specifics of unit testing

Function name	**Specific notes on recommended unit testing**

Additional hazards/ error conditions and failure modes identified.

Failure mode	**Resulting hazard or error condition the user will see**	**Mitigation action software must implement**	**Does SW need change?**

Outstanding issues	**Assigned to**	**Due Date - resolution**

SDS update required?	Y N	Who:

Sign-off date	

NOTE: ATTACH CODE REVIEW CHECKLISTS (Module: 1 page. Each Unit: 2 pages)

CODE REVIEW RECORD (page 2)

List of suggested and approved changes:

Change	Description of implementation

Other code review comments

NOTE: ATTACH CODE REVIEW CHECKLISTS (Module: 1 page. Each Unit: 2 pages)