MATLAB® @Work

CODING CHECKLIST

Richard Johnson

GENERAL

DESIGN

Do you have a good design?

Is the code traceable to requirements?

Does the code support automated use and testing?

Does the code have documented test cases?

Does the code adhere to your designated coding style and standard?

Is the code free from Code Analyzer messages?

Does each function perform one well-defined task?

Is each class cohesive?

Have you refactored out any blocks of code repeated unnecessarily?

Are there unnecessary global constants or variables?

Does the code have appropriate generality?

Does the code make naïve assumptions about the data?

Does the code use appropriate error checking and handling?

If performance is an issue, have you profiled the code?

UNDERSTANDABILITY

Is the code straightforward and does it avoid “cleverness”?

Has tricky code been rewritten rather than commented?

Do you thoroughly understand your code? Will anyone else?

Is it easy to follow and understand?

CODE CHANGES

Have the changes been reviewed as thoroughly as initial development would be?

Do the changes enhance the program’s internal quality rather than degrading it?

Have you improved the system’s modularity by breaking functions into smaller functions?

Have you improved the programming style--names, formatting, comments?

Have you introduced any flaws or limitations?

LAYOUT

PROGRAM

Does the program's layout show its logical structure?

Does the formatting scheme improve code readability?

Is the formatting scheme used consistently?

Are blank lines or section breaks used to separate blocks of code?

Have relatively independent groups of statements been moved into their own functions?

FLOW CONTROL

Are control sequences simple and clear?

Does the normal case follow the `if` rather than the `else`?

Is nesting minimized?

Have you removed unreachable code?

INDIVIDUAL STATEMENTS

Is there enough white space to easily parse each line?

Are lines short enough to read without scrolling?

Does each line contain one statement?

Have Boolean expressions been simplified by using additional variables and functions?

SELF-DOCUMENTING CODE

FUNCTIONS

Does each function name describe what it does?

Is each function interface obvious and clear?

CLASSES AND OBJECTS

Does each class and object name describe what it represents?

Are the properties and methods named well?

DATA NAMES

Are variables named well?

Are variables used only for the purpose for which they are named?

Are named constants used instead of magic numbers or magic strings?

DATA ORGANIZATION

Are extra variables used for clarity when needed?

Are data containers simple so that they minimize code complexity?

COMMENTING

Are comments up to date, clear, and correct?

Is the header information adequate to use the program correctly?

Are code and comments clear enough to understand the program?

Does every comment add value?

Is the commenting style easy to maintain?

Do comments precede the code referred to?

Are the comments indented the same as the code?

Are units on data variables commented?

Are limitations on input data commented?

Are flag variables documented?

Should any comments be enforced by assertions?

Have you removed commented out code?

This checklist was inspired by a general list written at Construx in 2002.