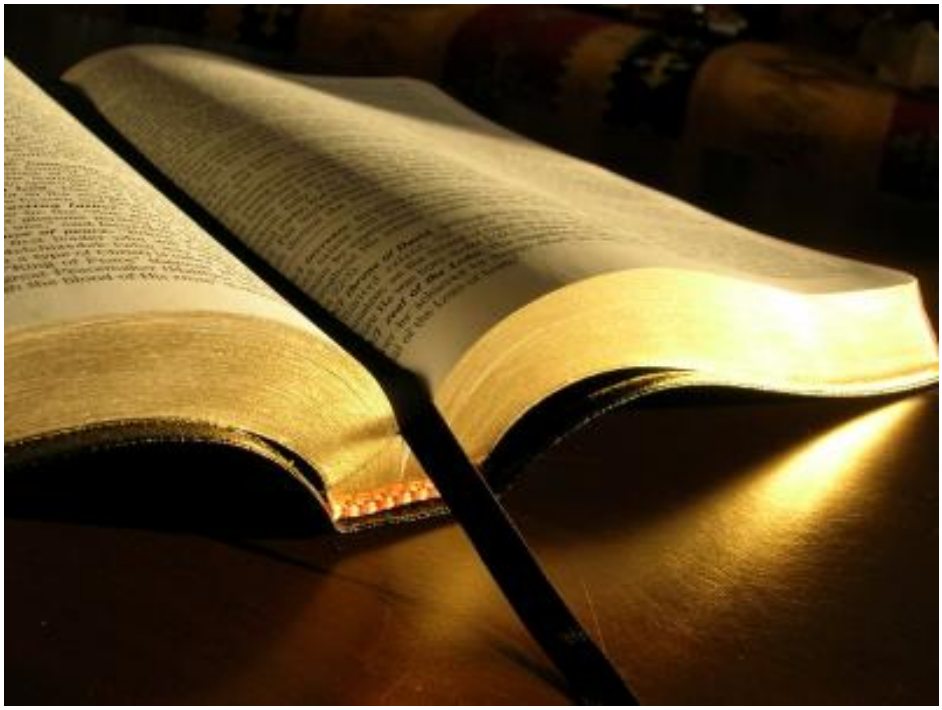
'The Ten Coding Commandments'



By David A. Giles B.Sc.(Hon) M.I.E.T.

Software Quality Measurement using the Gilo Scale

Measuring software quality is particularly difficult as in many instances software can run perfectly well even though it may contain bugs. Those bugs can lay dormant for years until a set of circumstances arise that produce an erroneous situation. In many instances the application can still perform even with the bug present and in many applications this is acceptable. A toaster for example that stays on too long by 10%, will not cause too much distress to the user. For applications like this the software can be upgraded on an ongoing basis without the need for a recall of the shipped product. However, a braking system for a car which takes 10% longer to brake under a software failure condition could endanger life and would be subject to a product recall. For both manufacturers of the products it would be ideal if these bugs were identified earlier on in the process and eliminated before entering the production cycle.

The scale of software reliability is difficult to measure, as there is not a scale currently in existence. For earthquakes we have a 'Richter Scale' which indicates to everyone in one number what the scale of the seismic activity is. Similarly for storms we have the 'Beaufort Scale', which in a number indicates wind strength and the issues it brings. For software however there is no number which we can put against it to reflect the design effort and testing effort that has gone into it. We all appreciate that an application for an aircraft actuator will have had a lot more effort put into in that say a domestic light dimmer. This is primarily due to the standards the overall product has to achieve.

For high integrity application we have standard such as DO-178B (Avionics) and IEC 61508 (Industrial) that many engineers work towards. These standards are for the product as a whole, although most of them do make reference to the software in a dedicated section.

If we were to produce a sliding scale for software development thoroughness on a scale of 0-10, then this would in an instance give some indication as to the level of thought and testing that has gone into the product. A low number indicates a not so thorough process and a higher number indicates a process of higher integrity. A low tech consumer product may only be required to achieve a 2-3 whilst an electric wheel chair would have to be an 8 or 9. Whilst we expect software with dangerous consequences to have been fully tested to the appropriate standards, we sometimes would not expect this of other products, for example a pregnancy tester. Whilst the pregnancy tester may give a faulty reading due to poor software, the implications are not as severe as they are for an aircraft engine designer. In this instance the design team would have to take a subjective view on what quality level to achieve when designing the product. What more how do they convey this to people outside the organisation or to other members within the company that are not part of the design team.

The Gilo scale has been developed primarily for embedded firmware engineer to gauge the amount of effort gone into it producing good software. Whilst non software people may not appreciate its subtleties, it does however convey in a basic number, something which people do understand and that it is a scale of effort. No amount of testing nor scrutiny on software can guarantee that software is error free, but the more of it you do the more your level of confidence increase. For this reason the Safety Integrity Level (SIL) of a complete system is measured in 'Probability of failure demand'.

Methods To Reduce Software Problems

A Software Plan

The specification of the final product should be clearly defined in a document that is available to the software team members and the QA department. The software team should have receipt of a product requirement document from the commissioning team so they can break this down into software requirements. The software requirements can be subdivided down into a software model which can then lead on to a detailed design spec for the module itself.

Each of the detailed designs specifications can then be individually coded by member software team. Each of these modules should have defined functionality such that the document can be given to both a test engineer and a design engineer. The designer should be able to write the code from this spec whilst the test engineer should be able to produce a series of test vectors for that module without the need to see the source code.

As large projects are difficult to manage, there should be a mechanism for communicating the changes that will be made throughout the project development cycle, such that changes can be communicated effectively. Documented changes should be standard practice.

Coding Standards

A coding standard should be adopted for the development of a project. This should be an in house coding standard that uses the existing available industry standards as a basis. The coding standard should be available to view and archived with other related documents. All software that is written to this standard must confirm and exemptions from this must be detailed in supplementary documentation.
Basic standards such as MISRA 98, MISRA 2004, ANSI C, IEC 9899:1999 should be used as a basis for the in house standard.

The in house standard should encompass naming conventions and coding styles also.

Version Control

A version control system should be used to mange documents, such as software files, header files, specifications, software plans, house coding standards etc. Test data should also be archived for future reference.

Static Analysis Testing

A static analyser should be used be used to analyse the produced C code. The compiler itself does provide some syntax checking but this should not be relied on. A third party static analyser should be

used to check the C source code and header files. The analyser should be used to check for the rules as indicated by the coding standard where possible, such as MISRA rules where selected.

Naming conventions and coding style do not need to be checked as it is unlikely for third party tools to do this reliably.

C Function Unit Testing & Software Integration Testing

Each C function should be independently tested in isolation from all the other C functions with a view to achieving 100% line coverage (C0), branch coverage (C1) and MC/DC coverage (C2). Deviations from 100% should be detailed in a separate document, indicating the reason why. In many instances software cannot be 100% MC/DC tested.

Software units whose functionality depends on the correct operation of other software units should be tested together. This is to ensure the integration of the software modules with each other is successful.

Peer Review

A peer review process should be used for those areas of the code that may not lend themselves naturally to unit testing. This would for example be used for blocks of code that use lots of special function registers. A peer review should also be done on all the code as a second pair of eyes can also pick up potential problems.

Test Strategy

The test engineer producing the test cases for the unit testing must be independent from the software code writer. The initial test cases must be generated from the documents provided with only the function prototype initially available to the tester. This may be reviewed following the unveiling of the source code and amended to achieve 100% MC/DC coverage.

Function Test Execution

The unit must be fully exercised with the complete object code present in the system and exercised in accordance with its normal operation. The tests must include typical scenarios that the product may see under fault condition. The testing scenarios must be documented and the tests results archived for future reference.

Tool Qualification

Each of the C compilers and test tools must have a tool qualification pack available for it. For C compilers it is expected that the tools have been validated by third party tool suites such as Plum Hall Validation Suite, and a statement of conformance be available for inspection.

For test tools it is expected that the manufacturers provide a Tool Qualification Pack. In the case of Unit Testing and Static Analysis testing, this TQP should be suitable for the target being used.

Field Trial

The product must be field trialled for a period of two months prior to shipping the first production units.

How the Gilo Scale Works

For each of the items listed above, the user should score 1 point for each of the items detailed. Adding up the total will give the Gilo figure of merit.

Example: 1

A light dimmer manufacturer has designed an 8 pin PIC12xxx into a new consumer unit. The unit is a twin version of a product they already have. The boss of the company instructs the designer to produce a new unit. The designer has neither paperwork, nor a project plan and sets about writing code from day one. He uses the existing in house coding standard and within two weeks he has a working prototype. He elects not to static analyses the code, nor have it unit checked, but does do some functional testing work on the code testing it at various frequencies, temperatures and input voltages. Once satisfied the unit is submitted to the electrical approvals board for approval.

On approval the unit is shipped to customers.

He keeps a back up of the software on a CD, which is kept in another location and this forms a basic archive of his work.

In this case he scores as follows:

A Software Plan	0	No formal plan exists
Coding Standards	1	The developer is using an in house coding style. This document detailing this should be archived with the code.
Version Control	0	No formal version control system is being used, although the user is archiving his finished software.
Static Analysis Testing	0	No static analysis is being done. By using a basic static analyser he can score one additional point easily.
C Function Unit Testing & Software Integration Testing	0	For this product no formal unit testing is required, but could be done simply in a relatively small amount of time.
Peer Review	0	As neither static testing nor unit testing has been done, it would be advisable to have a peer review. He scores no points as this process has been omitted.
Test Strategy	0	No formal test strategy for the software has been done. The product is not unit tested and the engineer will not be able to prove path coverage of the code execution.
Function Test Execution	1	The engineer has developed a functional test to prove the operation of the unit under different load conditions.
Tool Qualification	0	The engineer is reliant on the tool vendors providing adequately qualified products.
Field Trial	0	The engineer has relied on the approvals board for identifying weakness in his product.
Total	**2**	

Example: 2

A blood glucose meter is being developed to compete with a competitor product in the market place. The marketing team know exactly what is required and this is communicated via various documents describing in detail what is required of the product. The system architects have broken this down into manageable sections with the use of a third party modelling tool, which provides a function level overview of the code that is required. Three months are dedicated to writing the function level descriptions and what is required of each function.

A software team of three members is assembled to write the code and as such a version control system is put in place to help manage shared C source files. The customer uses its in house coding standard, which describes in detail the naming conventions of files, functions and variables. The house coding standard implies MISRA98 as a coding standard to which all code must be compliant.

As part of the embedded software quality process, all the software is static analysed using Development Assistant for C. To help with the certification of the product the customer elects to module test the software with TESSY to demonstrate due-diligence in the software testing process.

As the three engineers are all working together on the project a weekly design review of the software is done which facilitates a review of the code and an opportunity to raise concerns.

After completing the initial design a prototype is functionally tested before being submitted to the appropriate medical approvals agency for final approval. After approval the customer fields trials the units with key customers and early adopters before launching the product in the mass market.

The customer does not seek a tool qualification document for the C compiler as he is reliant on Unit testing and functional testing to prove the software works correctly. The customer is happy in the knowledge that a tool qualification pack exists for the testing tool.

In this case he scores as follows:

A Software Plan	1	A formal plan exists, with description of the software down to the function level.
Coding Standards	1	The developer is using an in house coding style which also makes some reference to MISRA98 coding standard.
Version Control	1	Formal version control has been introduced to help with the management of documents, and files.
Static Analysis Testing	1	DAC has been chosen as the static analysis tool.
C Function Unit Testing & Software Integration Testing	1	Unit testing is completed to help build confidence in the software modules and to help demonstrate due diligence has been taken with it.
Peer Review	1	The team members get together once a week to review each other's work.
Test Strategy	1	One member of the software team is nominated to do the unit testing. He is given the original documents for the function descriptions, which he uses to decide on

		test cases. I have given a point here as the engineer is working from documents to generate the test cases rather than just looking at the source code.
Function Test Execution	1	The engineer has developed a functional test to prove the operation of the unit.
Tool Qualification	0	The engineer is reliant on the tool vendors providing adequately qualified products.
Field Trial	1	The marketing team have elected to field trial the product with key customers who can provide feedback on the design.
Total	**9**	