CHAPTER 4

QUALITY MODEL FOR FLIGHT SOFTWARE
Chapter 4

QUALITY MODEL FOR FLIGHT SOFTWARE

4.1 QUALITY MODELS

Many of the early designs of quality models have followed a hierarchical approach in which a set of factors that affect quality are defined, with little scope for expansion. A metrics measurement-based framework, linked to a quality model, is a requirement for the effective production and quality of the software. A quality model that is to be used as the foundation for the definition of quality requirements should help in both the specification of quality requirements and the evaluation of software quality [94].

The literature on the subject covers several product quality models, the best known in chronological order of appearance are: McCall, Boehm, FURPS, ISO 9126 etc. [92]. A model is an abstract form of reality, enabling details to be eliminated and an entity or concept to be viewed from a particular perspective. The ISO/IEC 9126-1[59] quality model is defined by means of the general characteristics of software, which are further refined into sub characteristics, which in turn are decomposed into attributes [52]. This results in a multilevel hierarchy, and at the bottom of the hierarchy appear the measurable software attributes.

4.2 DEVELOPMENT OF A QUALITY MODEL FOR FLIGHT SOFTWARE

The embedded software residing in the flight computers of a Satellite Launch Vehicle has to be error free to meet the stringent requirements of a particular mission. As more and more functions are realized by flight software, its quality and reliability become the major
decisive factors in the success of a mission. There are several approaches for specifying quality features. A recent innovation in systems area is to standardize software quality by defining models that combines and relates the software quality characteristics and sub-characteristics. Initially a software development model was constructed and quality steps were identified in each stage.

As noted in ISO/IEC 9126-1, "Software quality should be evaluated using a defined quality model". So the primary step in assessing the software quality is to develop a quality model for flight software. The main benefits of developing a quality model are the following:-

- Identify the factors important for that application: There are many quality factors or attributes available. But the advantage of developing a quality model is that it helps in identifying the critical factors which are relevant for the successful operation of that software.

- Assess the quality of the software with the aid of error detection techniques: Once the quality model is defined, it is possible to establish methods or techniques to assess the extent to which the software complies with the required quality factors. The popular method adopted is extensive Verification and Validation methods through which the errors are detected.

- Root cause analysis of errors: This is the most crucial and most difficult part. A systematic analysis of all probable causes that resulted in the errors will be beneficial in preventing the occurrence of such errors. This helps in establishing the complex interrelationships between various factors by using them to train the neural network.
• Provide guideline for improving the quality of the software: The quality model helps in identifying the input which was the cause for the error in the product. If the culprit for the error is traced, providing guidelines to improve the quality is simple.

4.3 Development Environment of Flight Software

For developing a quality model for flight software, the environment in which it is developed has to be considered. The development of flight software takes place in an environment where the mission requirements are evolving, design parameters are frequently changing and mission constraints are revised from time to time. These problems accompany any project whose development period spans a few years. Hence during the flight software development for a new satellite launch vehicle, different versions of the software emerge, which undergo rigorous software evaluation before the final product is cleared for the satellite launch mission.

The Iterative Waterfall Software Development Life-cycle model is adopted. Different software components are defined to do the different functions like navigation, guidance, control, sequencing etc. The software components developed by different design teams are integrated together to form the launch vehicle software. Various plans are generated to realize a quality product. The Quality Assurance agencies ensure that all reviews and audits take place as per the documented plans. The development and testing of the flight software takes place in a controlled environment. The flight software development model is given in Figure 4.3.1
The development environment also plays a key role in the quality of the flight software. For a new vehicle under development, the final validation of the software can be only during the flight, as all the ground validations are done using simulated flight profiles. Quality software is achieved by following a systematic development and evaluation process, from which a quality model for flight software has been developed.

**Figure 4.3.1 Flight Software Development Model**

- Functional Requirements
- Software Requirements
- Software Detailed Design
- Coding
- Designer Tests
- Code inspection
- Module tests
- Simulations
- Software & Data Finalization
- Final CC Release of code & data
4.4 Quality Characteristics Required for Flight Software

Flight software is defined as high quality software that satisfies the functional, performance and interface requirements and is available at the required schedule, meeting the requirements of reliability, maintainability, testability and correctness. A non-conformance in any of the above causes a schedule delay until the non-conformance is rectified. In order to develop a software quality model that is useful in the quality management process, it is necessary to identify a set of factors that is common to most projects and at the same time meeting the requirements of high quality software. The quality factors applied in flight software are -Functionality, Testability, Reliability, Efficiency and Portability. The quality factors are further subdivided into sub-factors. The Flight Software Quality Model is given in Figure 4.4.1
4.5 EVALUATION OF FLIGHT SOFTWARE

The flight software quality model has been extensively used in defining the development life cycle as well as in the verification and validation activities. Evaluation of the product quality has been extended to encompass system functional requirements specification, software requirements specification, detailed design document, code, test documentation, or reports produced as a result of quality analysis tasks. The different reviews and tests at different life cycle stages of flight software are given in Figure 4.5.1.

After complete evaluation, a Product Assessment Report is issued by Quality Assurance (QA) agency to the Project or end user.
4.5.1 Functionality Evaluation

Functionality is defined as the ability of the software product to provide functions which meet stated or implied needs of the user. The sub factors are correctness, interoperability, suitability, compliance to standards.

The quality factors with the associated sub factors are evaluated from the beginning of the software development life cycle. Though Waterfall lifecycle model is adopted for software development, unlike the conventional model, it is preceded by a stage called Functional Requirements or Algorithm formulation phase.

The Quality agencies verifies the adherence of the Functional Requirements document (FRD) and other documents to approved standards and guidelines (Functionality Compliance). When Software Requirements are analyzed and Software Requirements Specifications (SRS) are generated, the correctness of mapping the Functional requirements and Interface specifications (Interoperability) are reviewed by Design Review Team (DRT). The SRS compliance to standards is verified by QA agencies.

Similarly when Software Detailed Design (SDD) is generated, the correctness of mapping the SRS is reviewed by DRT and compliance to standards is verified by QA agencies. Static analysis of codes is a strong mechanism for verifying the correctness of code. Code inspection is the static analysis technique adopted for code verification. Traceability of code with reference to design and requirements, compliance of code to coding guidelines, presence of unused or undefined variables, and dead codes or
redundant codes etc are some of the aspects addressed in code inspection. The defects detected during Functionality evaluation are spread over different life cycle phases.

4.5.2 Reliability Evaluation

Reliability is defined as the ability of software to maintain a specified level of performance within the specified usage conditions.

Reliability in the context of flight software is defined as the capability to deliver correct results under different perturbation conditions within the acceptable operational range. Conventional software reliability models are not used for the reliability prediction. The reliability of the software is established through dynamic tests- module level tests and different simulation tests. This evaluation is done during testing phase. The sub factors defined are robustness, accuracy, fault tolerance, error detection capability.

Module level tests are carried out for flight software to establish the functional correctness of the module, to qualify the structure of the module, and to establish module level robustness for valid and invalid inputs. The system testing, which involves the integrated system with associated hardware, is carried out in different simulation test beds with different levels of nearness to flight conditions.

4.5.3 Maintainability Evaluation

Maintainability describes the ease with which the software product can be analyzed, changed and tested. All types of modifications, i.e. corrections, improvements and
adaptation to changes in requirements and in environment are covered by this characteristic.

This is the evaluation of the capability of the software product to be tested and modified. This evaluation is carried out during the requirements and design phases and during static analysis done on the code. Testability feature is included in the requirements and design phase itself. Telemetering of important parameters to enable assessment of the correct functioning of the software is built into the requirements itself. Testability is also a criteria while designing the software using the design guideline that McCabe’s complexity should not be more than a specified number.

4.5.4 Efficiency Evaluation

Efficiency is a characteristic that captures the ability of a correct software product to provide appropriate performance in relation to the amount of resources used. Efficiency can be considered as the ability of the product to fulfill its purpose without waste of resources.

Efficiency is evaluated mainly from the execution time and resource utilization point of view. Since the flight software is resident in the real time computer systems, producing the correct output at the correct time is very important. Also a transport delay from sensor acquisition to control actuation is defined, and the timing requirements have to be met under all nominal and off nominal conditions. Simulating different error conditions and ensuring that no task incompletion has occurred or no slippage in the time of
acquisition/posting of commands has occurred is one means of assessing this. Quantifying the worst-case execution time margin in a cycle of computation is also done during validation under different stress cases.

Resource utilization is assessed mainly on the usage of memory for programming. This is done as part of code inspection, where optimization from execution time and memory utilization are specific points addressed by the code inspection team. Also diagnostic utilities to study memory utilization or stack growth are built in.

4.5.5 Portability Evaluation

Portability is a measure of the effort that is needed to move software to another computing platform. For our advanced missions where two different processors are used, this becomes significant.

Evaluation of portability is done mainly to assess the easiness of using the software in a different platform. In the Launch Vehicle flight computers, there are two sets of processors used for mission control. Though the processors are different, hardware compatibility are maintained through common bus architecture. Also the software interfaces are kept common so that interaction with other systems is not affected. These are verified as part of requirements and design review. To maintain adaptability between two processors, more design guidelines are given to the designers while developing software, so that common source code can be used in both the processors. Adherence to these guidelines is verified as part of code inspection (Adaptability). The integration plan
review team assesses \textit{installability}, where the steps for installing software in both the processors are evaluated to keep the procedure as simple as possible.

4.6 \textbf{Evaluation of Quality Factors}

Thus the five quality factors identified for flight software are – Functionality, Reliability, Maintainability, Efficiency and Portability. As seen from the V&V cycle, the different reviews are FRD review, SRS review, SDD review, Code inspection and Testing. The evaluation of these factors is carried out against a checklist during the identified reviews as given in Table 4.6.1

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|c|c|c|}
\hline
Review & Functionality & Reliability & Maintainability & Efficiency & Portability \\
\hline
FRD Review & \checkmark & & \checkmark & & \\
\hline
SRS Review & \checkmark & & \checkmark & & \checkmark \\
\hline
SDD Review & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark \\
\hline
Code Inspection & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark \\
\hline
Testing & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark \\
\hline
\end{tabular}
\caption{Factors evaluated during V&V Cycle}
\end{table}

The quality model meeting all the quality factors is developed and is used for generating the verification and validation plan. The reviews and verification and testing primarily focuses on quality attributes defined by the quality model and when these attributes are not met, they are denoted as defects. The defects detected during different phases are collected for various software products or components.