CHAPTER 3

SYSTEM STUDY AND REQUIREMENTS SPECIFICATION
SYSTEM STUDY AND REQUIREMENTS SPECIFICATION

3.1 SOFTWARE QUALITY

The system requirements and specifications have to be clearly defined to achieve the goal of quality assessment. A normal SDLC consists mainly of the following phases- Requirements Analysis, Design, Coding, Testing and Operation / Maintenance. For flight software, if any discrepancy is noticed during flight, the anomaly is analyzed during post flight data analysis and suitable corrections are incorporated in the next mission after detailed studies and reviews. Hence operation phase is not considered separately. The SDLC in flight software is the iterative waterfall model- so that in each iteration a working software system is available.

This research work depicts how an analysis of the defect data from different iterations can provide valuable information for preventing defects in subsequent developments. If the number of defects can be predicted, they can be prevented and defect prevention can improve quality and productivity. The premise behind this is - if the causes are understood, then efforts can be made to eliminate them, thereby reducing the number of defects getting injected subsequently in the next phases of development and also in new missions.

Also in a product, if the number of defects that are likely to occur during different phases of SDLC are known, it is possible to assess its quality through neuro-fuzzy logic and define the risk associated in using that software.
3.2 Software Quality Assessment Model

This study is a step in the direction of achieving a more complete picture of software quality and a broader perspective of software quality using available measurable data. Managers and developers should understand what aspects of software quality are important to them and to users, so that they can ensure that developers of the software implement those features.

Quality of software is usually measured by the number of defects detected in the final product. Poor quality eventually affects cost and schedule because software requires tuning, re-coding, or even redesign to meet original requirements. Cost and schedule overruns are common because serious problems are often not discovered until the system integration phase.

Minimizing the number of defects, requires thorough testing of the software. Testing phase requires approximately 50% of the whole project schedule. This means testing is the most expensive, time and resource consuming phase of the SDLC. Also testing comes very late in the development life cycle.

If it is possible to develop a predictive model which can predict the number of defects before any code is written, it will help the project managers and testing agencies to focus their efforts on more error prone phase / products. As with any predictive model, there is a need for training data. In this case the data available are the defect counts obtained over different phases of software development. An analysis of the defects, could bring a
correlation to the development as well as verification and validation practices including the environment, in which the software is developed and tested. The data obtained over different projects was collected and a training data set is prepared. This was used for training the neural network which can predict the number of defects in each phase. From the defects predicted, using neuro-fuzzy rules a crisp value can be obtained for the quality of the software product delivered.

### 3.3 Software Quality Model

A software quality model is useful for identifying quality factors from a set of direct measures [93]. A quality model takes as input a set of values for a software product and produces a quality factor which is identified in the model. Software product metrics are very popular as indirect metrics of quality. Most models measure both design and code attributes but there is no unique set of product metrics that all the models use. Although the requirements to measure product and process from the viewpoint of quality are stated, in many models there is little guidance as to what should be measured and how the measures can be used to support the development of high quality software. An additional problem is that measurements cannot be defined in a totally context independent manner.

The research work defines a quality model for flight software. The main attributes or quality factors- Functionality, Reliability, Efficiency, Maintainability and Portability are defined. The quality model helps in focussing the evaluation efforts by identifying the parameters that are important for the software under consideration. While evaluating these quality factors, all the non-compliances are classified as defects and they are
analysed and root causes are identified. The quality model is useful in identifying the inputs and outputs that can be used for building a defect prediction model.

3.4 Defect Prediction Based on Neural Network

As stated earlier, the research focuses on developing a quality assessment model based on neural network and fuzzy inference concepts. A Feed Forward Neural Network is used to predict the defects. The defects detected during evaluation of the quality attributes during the different phases are analyzed and based on the root causes, the environment or conditions that caused these defects to occur in the four phases of SDLC mainly—Requirements, Design, Coding and Testing are identified. The environment or conditions that are likely to cause these defects are defined as the input factors and are used for training the network. The defects in different phases are the outputs used for training the neural network. Four back propagation network (BPN) models – one for each phase respectively are trained using the data obtained from different versions of software developed for previous Launch Vehicle flights. If a Neural Network model is trained with the data available, it can be used as an estimator for predicting the number of defects in the software that is developed for a new Launch Vehicle Project [72] [73] [109].

3.5 Fuzzy Inference Logic

When the numbers of defects are predicted from the Neural Network, this can be fed into a quality estimator based on Fuzzy rules [125]. Different fuzzy rules are implemented based on the impact of errors on quality in each phase, and the quality value of the corresponding phase can be obtained. Using a fuzzy inference system employing fuzzy
It is possible to model the qualitative aspects of human knowledge and reasoning processes without employing precise quantitative analysis [141] [143] [144]. One of the limitations of fuzzy logic is while transforming human knowledge into rule base, no standard methods exist. Also standard methods are not available for tuning the membership functions so as to minimise the output error. In the case of flight software, all the combination of input conditions could not be captured as fuzzy rules. Hence the possibility of using neural network techniques to generate the fuzzy rules by means of an Adaptive Neuro-Fuzzy Inference System to assess the quality of flight software was explored.

### 3.6 Neuro-Fuzzy Model for Quality Assessment

Adaptive Neuro-Fuzzy Inference System (ANFIS) is a fuzzy inference system implemented in the framework of adaptive networks. Using an input/output data set, the ANFIS method constructs a fuzzy inference system whose membership function parameters are tuned using a back propagation gradient descent and a least square type of method. A neuro-fuzzy model will help in solving the two main problems faced in fuzzy reasoning- lack of a definite method for determining the membership functions and the lack of a learning function for self tuning inference rules. Thus for solving the problem of quality assessment of software from the number of predicted defects from the different phases of software development, a neuro-fuzzy model was adopted.

In this model, the neuro-fuzzy system provides the fuzzy system with the kind of automatic tuning methods typical of neural networks but without altering their
functionality. The neural network provides the connectionist structure and learning abilities to the fuzzy logic systems, and the fuzzy logic systems provide the neural networks with a structural framework with the high level fuzzy IF-THEN rule thinking and reasoning [62] [63] [65]. These integrated systems can learn and adapt. It is proposed to build a quality assessment model based on ANFIS by generating a set of fuzzy rules which can assess the quality and are optimized by training strategies originating from neural network. The Neuro-Fuzzy Quality Assessment Model is based on predicting defects based on neural network and assessing the quality from predicted defects based on adaptive neuro-fuzzy inference system.

The development of a Quality Assessment model is split into three parts:

- Defining a Software Quality Model to identify the quality attributes thereby enabling the evaluation efforts to be focused and productive
- Developing a Defect Prediction Model based on neural networks
- Developing a Quality Index Computing Model based on neuro-fuzzy inference logic.

Thus the Quality Model helps in clearly defining the quality attributes important for flight software and helps in focusing all verification and validation activities suitably to evaluate these attributes. From the evaluation of the Quality Model we get the defects. By analysing the defects, it is possible to identify the input factors which resulted in these defects. These input defects and output factors can be used for developing a Defect Prediction Model based on neural network. From the defect prediction model we get the predicted defects which can be fed into an Adaptive Neuro-Fuzzy Model to generate the
Quality Index for new software that is developed. Figure 3.6.1 gives the block schematic of the Quality assessment model

**Figure 3.6.1 Quality Assessment Model**

Thus the development of quality assessment model is based on the following:

- a. Identify the quality attributes of flight software from a quality model
- b. Evaluate the quality attributes
- c. Collect the defects from each phase of software development
- d. Analyze the defects and find the root cause of these defects
- e. Determine the inputs causing these defects
- f. Build a defect prediction model based on neural network using Back propagation algorithm using the input and output data
- g. From the defects make an assessment of quality using Fuzzy Logic by generating the quality index
- h. Overcome the limitations of fuzzy logic by adopting adaptive neuro-fuzzy inference system for assessing the quality index from the defects predicted
- i. Assess the quality of the software based on the quality index computed and classify the software as high risk, medium risk, low risk and no risk.
The steps involved in development of the Neuro-Fuzzy Model for Quality Assessment as shown in Figure 3.6.2.

Figure 3.6.2 Development of Neuro-Fuzzy Quality Assessment Model

1. Analyze the defect Data
2. Determine Inputs of Neural Network based Defect Prediction Model
3. Train Defect Prediction Model with data
4. Predict the defects based on inputs for a new software
5. Fuzzify the inputs (Defects)
6. Develop Fuzzy Rule database
7. Infer the Quality of the software
8. Assess the Quality Index
9. Categorize based on risk