CHAPTER 1

INTRODUCTION
INTRODUCTION

1.1 SOFTWARE IN INTELLIGENT SYSTEMS

Software is an integral part of many of the modern intelligent systems used for critical applications like satellite launch vehicles, strategic missiles, aircrafts, nuclear power plants and patient monitoring systems. As software systems become more complex to design and develop, intensive studies are carried out to increase the chance of the software performing satisfactorily in operation. Software engineering has been the fastest developing technology during the last century, but even now there is no complete, scientific measure to assess the quality of software. In order to increase the performance of the software product and to improve the software development process, a means of assessing the reliability and quality of the software must be established. This research work focuses on deriving a model for quality assessment of the software used in intelligent systems.

1.2 DEFINITION

A few definitions pertaining to the research work is given below.

Software Defect/Error:

a) The difference between a computed, observed or measured value or condition and the true correct value or condition

b) It is also an incorrect step, process or data definition

c) Not meeting the specified requirements.
Quality: The degree to which a system, component or process meets specific requirements or customer/user needs or expectations.

Quality Attribute: A feature or characteristic that affects an item’s quality

Software Requirements Specification (SRS): Documentation of the essential requirements (functions, performance, design constraints and attributes) of the software and its external interfaces.

Software Design Description (SDD): A representation of the software created to facilitate analysis, planning, implementation and decision making.

The term ‘defect’ and ‘error’ are used interchangeably throughout the thesis to denote the definition given above.

1.3 SOFTWARE QUALITY IN INTELLIGENT SYSTEMS

With software embedded into many systems, like any engineering subsystem, software also fails; and software failures have been one of the major causes of failures of satellite launch vehicles and fighter aircrafts. The causes have ranged from poorly designed user interfaces to direct programming errors. The risks associated with such software failures are very high - economically and strategically, and can even cause loss of life. Hence the quality of software in these systems has to be very high, as any error in the software can jeopardize the whole mission. Achieving a high level of quality and reliability in the software systems realized for various applications has been a challenge for software developers, system integrators, and users. Quality assessment and improving the quality,
is a difficult task as there is no model, which can accurately predict the quality of software in intelligent systems. In this thesis, the flight computers of a Satellite Launch Vehicle have been taken as a typical intelligent system. The embedded software denoted as flight software, residing in the flight computers has the critical task of guiding and controlling the vehicle to precisely inject the satellite(s) into the desired orbit. The success of a Launch Vehicle mission thus depends to a great extent on the quality and reliability of the flight software. Hence it becomes all the more critical to have a quality assessment model, which can be used for assessing and improving the flight software. Due to the inherent differences of a software product from a hardware product, the well-developed theories of reliability engineering cannot be directly applied to a software product.

1.4 LAUNCH VEHICLE FLIGHT COMPUTERS CONFIGURATION

The flight computers or onboard computer system used in Launch vehicles is a real time distributed processing system. The flight computers control the launch vehicle from lift off till the satellite is injected into the specified orbit. The major components of the onboard computer system are two Advanced Mission Computer Processors which acquires data from sensors and does Navigation, Guidance & Control computation and gives necessary commands through Stage Processing Systems and Control Electronics. A dual redundant hot standby system is realized so that error handling is done through failure detection, isolation and re-configuration. For design diversity, primary processor is the indigenous processor and the redundant one is a bought out processor. The software resident in flight computers of a satellite launch vehicle has to perform the critical
function of putting a satellite into a desired orbit meeting stringent orbital specifications. An error or failure in the flight computer or software can cause catastrophic mission failure. The configuration of the flight computers is given in Figure 1.4.1.

**Figure 1.4.1 Flight Computers Configuration**

The prime chain will be the controlling chain, with the redundant chain working as hot standby. Since all elements in prime chain are connected to redundant chain also, when any one element fails, a reconfiguration isolating that element alone from the controlling chain is possible through software. This configuration can tolerate two non-identical processor or link failures in the data loop.

The software components which do the mission critical functions of navigation, guidance and control called application software is realized through Ada codes. The real time executive which does the software/hardware interfaces and resource management and scheduling of tasks is realized in the respective processor assembly language to optimize the execution time and memory used.
The flight computers works as an Intelligent System because of the various error handling features that are built into it. An extensive redundancy management scheme is introduced in the design of the flight computers through hardware and software to handle all sorts of interface related errors and failure of any one hardware component. Apart from ensuring module level performance in the design, each module is designed robust enough to handle unforeseen nature of input conditions. In the event of an unanticipated input, the software is designed to take the previous valid data input and proceed with computations until a valid input is received. With this feature, the module is expected to function even beyond the operating domain of inputs. All algorithm level singularities are identified and handled in the software design. The mission performance is monitored during the course of the flight, and in case of severe performance deviations, salvage options which are built in the software are invoked to redirect the vehicle to the nearest better target orbit to avoid catastrophic mission failure. Thus the flight computer of a Launch Vehicle is an Intelligent System. The quality assessment of the mission critical software resident in the flight computer is attempted in this thesis using neuro-fuzzy computing techniques.

1.5 Current Scenario for Assessing Quality of Flight Software

At present software quality is achieved through a well-defined development process and thorough defect detection and defect prevention techniques appropriately applied at all phases of software life cycle. For this, rigorous verification and validations steps are followed, and strict process control is achieved through reviews, inspections and audits. The dictum of building the quality into the product from the beginning is followed,
instead of inspecting and trying to build quality into the realized product. No specific quality metrics are identified and the quality improvement is not quantified. At the end of the current flight software quality evaluation process, if the final product meets the mission requirements, it is cleared for flight use.

1.6 LIMITATIONS OF PRESENT METHODOLOGY OF QUALITY ASSESSMENT

As stated above, current methodology of Quality Assessment is based only on correcting the errors detected and ensuring the software meets the requirements through a well-defined Verification and Validation (V&V) scheme. No systematic analysis of the causes of defects is conducted, and when software for a new project has to be developed in a similar environment, it is not possible to predict the errors that are likely to occur in that software.

There are different software prediction models – software defects prediction models and software quality prediction models. Existing empirical validation studies collect product metrics as independent variables and aim at predicting software faults or software development costs as dependent variables by applying a variety of software prediction models. A software quality prediction model maps metrics values to a defect or a quality attribute. For statistical validation, the quality attributes have to be assessed quantitatively as well as independently of the metrics and quality models.

One method of assessing the quality of software is by software reliability prediction. The application of statistical modelling techniques has been pursued for this purpose.
Different models have been proposed for characterising software reliability. A study of the various models available reveals, no single statistical model can predict the reliability of critical software in intelligent systems. Various defect prediction models were also studied. The general criteria used to compare the available models were - ability to predict future failure behaviour from present and past failure behaviour, capability of the model to estimate with satisfactory accuracy the qualities needed by software managers and engineers in planning and managing software development projects, the quality of assumptions made and the feasibility of evaluating the model from the viewpoint of consistency. The model should be judged on the basis of its applicability across software products that vary in size, structure, function and different operational environment. Also the model should be simple and permit usage of existing / available inexpensive data.

1.7 Motivation

It is necessary in any software product, that it performs its intended function satisfactorily. For that, it is essential that the errors detected during different phases of Software Development Life Cycle (SDLC) are analyzed to establish the root cause, and suitable corrective and preventive actions are taken to prevent these defects from recurring. Evaluation, prediction and improvement of the software quality have been of concern to project managers and quality assurance teams, wherever software is used for critical applications. Lack of a model for assessing the quality of flight software used in flight computers of a Satellite Launch Vehicle has been a stumbling block in making predictions about the software quality and focusing the Verification and Validation efforts on error prone products and processes. Also the need for a model which can make
an assessment of the quality of flight software has been felt for a very long time. Especially in the scenario of multiple launches with international commitments, timely deployment of manpower with focused efforts is crucial to meet the schedule in a cost effective manner.

1.8 Original Contributions of the Thesis

The objective of the study is to develop a model for the assessment of the software quality of the intelligent systems like those used in Satellite Launch Vehicles. The research work carries out a study of the defects detected during different phases of the software development life cycle, analyze the root cause of these defects, and quantify them as factors that can be fed into the quality assessment model. In fact it is observed, that the complexity of the problem, type of documentation, programmer expertise and process adequacy - all play a crucial role in the quality of the software product realized. None of the statistical models studied is able to capture the complex interrelationships between the four factors cited above and arrive at a quality factor. The environment in which the software is developed cannot be captured in a mathematical model. Also varying one factor and studying its effect on the total quality is also not possible with a mathematical model. Only non-mathematical models like Neural Networks can capture the above two aspects.

If the environment characteristics and defects obtained are used for training Neural Networks, the number of defects that are likely to occur can be predicted. The correctness with which the number of errors can be predicted depends on how correctly the
development environment is characterized and how effectively the complex uncertain relationship between software quality external factors and internal factors are captured. In the case of flight software which consists of different software components, at component level sufficiently large data is available to train the Neural Network. Also with Neural Network, it is possible to vary one input parameter and study its effect on the system, and thus identify the most important input factor which can affect the number of errors introduced into the software developed. Identifying the most critical input and taking appropriate preventive and corrective action is possible using a Neural Network based defect prediction model.

Based on the defects predicted in different phases of software development like requirements, design, coding and testing phases, an assessment about the quality of the software can be made. For assessing the quality from the number of defects predicted, it is proposed to build a quality assessment model based an Adaptive Neuro-Fuzzy Inference System. The thesis work will deliver both research insights and practical methods for quality assessment of software used in intelligent systems, considering the flight software embedded in Launch Vehicles as a typical example.

On the research side we expect three major contributions:

1. We define a Quality Model for flight software from which it is possible to establish methods or techniques to assess the extent to which the software possesses the required quality attributes. The proposed model uses the defects detected during different phases of software lifecycle while quality attributes like functionality,
reliability, maintainability, efficiency and portability are evaluated. The causes for these defects are analysed and it is observed that there exist many factors that have an impact on the quality of the target software product.

2. Based on the quality model defined, it is possible to identify the inputs which affect the quality of the software. These input factors, their relationship and software quality attributes is a complex, non-linear and multivariate problem. With the data available for the inputs and outputs, it is possible to train Artificial Neural Network for each phase to predict the number of defects that are likely to occur in that phase. We develop a Defect Prediction Model using Artificial Neural Network.

3. From the number of defects predicted, a Neuro-Fuzzy model for quality assessment was developed. Given the defects detected during different phases of development, it is possible to arrive at a quality index by generating a set of fuzzy rules which are optimized by training strategies originated from a neural network. An improvement in software development process and the quality of the software produced depends on predicting defects and thus the quality, to a very high degree of accuracy.

The Neural Network alone can be used, if the objective is to predict the number of defects in different phases. The number of defects is an indication of the quality of the product - the more the number of errors, the more the probability of residual defects. Also the Neuro-Fuzzy Inference Engine alone can be used to make an assessment of the
software quality if the defects detected during the Verification and Validation cycle are fed into the Fuzzy Inference engine.

Thus together 1, 2 & 3 can be used as a Quality Assessment Model in any general purpose intelligent system since they provide the theoretical basis for quality management assessing industrially standardized software quality factors in an effective and efficient way.

1.8.1 Development of a Quality Model for Flight Software

A quality model specifying the relevant quality factors- Functionality, Testability, Reliability, Efficiency and Portability and their sub-factors was defined for flight software. Based on the analysis of the causes of the defects in different phases of the software development life cycle, the various input factors which can cause errors in the Requirements, Design, Coding and Testing phases are identified using the quality models. The attributes corresponding to these factors were finalized from the evaluator’s view. The data pertaining to these attributes were collected from software used in previous flight development and suitably decoded for training. The main factors causing defects during different phases were identified as complexity of the problem, adequacy of documentation, experience of the programmer/tester and the effectiveness of the processes employed. The sub factors for each phase were identified and the four main factors feeding into these errors were derived as a function of these sub factors. The sub factors/main factors for each phase were identified from the data.
1.8.2 Using Neural Network for Defect Prediction

The advantage of having a software defect prediction model is that, the management could identify fault prone artifacts or modules in advance, which will help in allocating the limited resources available for software verification and validation more efficiently in terms of quality improvement. Existing statistical models typically rely on assumptions about development environment, nature of software errors and the probability of individual failure occurring. These reliability growth models exhibit different predictive capabilities at different life cycle phases, within a project and across projects.

Artificial neural network and Fuzzy logic are the two key technologies in computational intelligence that have received growing attention in solving real world, non linear complex problems. Because of their learning and / or reasoning capabilities, these techniques do not require a mathematical model of the system under study, which may be difficult to obtain for complex systems.

Neural Networks are able to capture the interrelationships between various factors identified in the Quality Model which can cause the defects during different phases of Software Development Life Cycle. For each phase of the software development life cycle, a neural network was trained with the input data identified as causes for the defects in that phase. There are four neural nets for Requirements, Design, Coding and Testing phases respectively. Neural Networks due to the non-mathematical modeling techniques can be trained to predict the defects if the software is developed in a particular environment. Availability of an accurate model is essential in flight software
development for meeting the schedule of multiple launches with international commitments in a cost effective and timely manner.

Feed forward network is set up and back propagation algorithms are used for training. Based on the environment or input conditions under which the flight software is developed, and the errors detected under different phases of SDLC which are used for training the Neural Network, it is possible to predict the errors that will occur for new software developed under that conditions.

1.8.3 Neuro-Fuzzy Model for Quality Assessment

Fuzzy logic is a technology for developing intelligent control and information systems. Fuzzy logic achieves machine intelligence by offering a way for representing and reasoning about human knowledge that is imprecise by nature. The main advantage of a Fuzzy Inference is the ability to model the dynamics of a problem domain using a linguistic model which is essentially a fuzzy rule base consisting of a set of IF-THEN rules that can be easily comprehended by the users. Fuzzy logic provides an assessment of the quality of the software based on the number of defects detected during different phases of SDLC. Numerical values of errors or quality are not acceptable in scientific terms. Fitness for application in terms of risk is usually an acceptable method of assessing quality of software. For assessing the quality of flight software from the number of defects predicted, 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 (eg fuzzification,
defuzzification, inference engine, and fuzzy logic base). In neuro-fuzzy systems, neural networks are used in augmenting numerical processing of fuzzy sets, such as membership function elicitation, and realization of mappings between fuzzy sets that is utilised as fuzzy rules. Hence Neuro-Fuzzy logic is used to assess the quality of the software from the defects predicted in each phase by the Neural Network. The ultimate output of the Neuro-Fuzzy Model is a quality index using which the quality of the software is assessed as say excellent, good, moderate or poor. And based on the output, it is possible to assess the risk in using that software for an application. All simulations of the Artificial Neural Network and Neuro-Fuzzy Logic are simulated using Matlab tool kits.

1.9 LAYOUT OF THE THESIS

The thesis begins by introducing the subject and defining the aim and scope of the present research work in Chapter-1. As a typical example of an intelligent system the flight software residing in a launch vehicle is taken for study. The perceived contributions arising out of the present work are also highlighted. The introduction is followed by a literature survey on the subject in Chapter-2, which establishes the current state-of-the-art as well as provides a basis for the work undertaken in the present thesis. The summary of work related to the quality assessment and defect prediction models published in open literature and the limitations of the methodologies with respect to the aim of the thesis are given in this chapter. Chapter-3 gives the system requirements and specification. Chapter-4 gives the development of a Quality Model suitable for flight software residing in a launch vehicle defining the quality attributes to be evaluated. Chapter-5 describes the identification of input and outputs for the Defect Prediction Model based on analysis
Chapter-6 proposes a Neural Network approach for predicting the defects in software. It also gives the details of the software defect prediction models for each phase of the software life cycle obtained by training the neural networks. Chapter-7 gives the reasoning behind using Fuzzy logic and then adopting Neuro-Fuzzy Expert system for assessing the software quality from the defects predicted. In the conclusion, the research work is summarized and indicates the scope for future work.