SYNOPSIS

I. INTRODUCTION

Software design, development and testing have become very intricate with the advent of modern highly distributed systems, networks, middleware and interdependent applications. The demand for complex software systems has increased more rapidly than the ability to design, implement, test, and maintain them, and hence the reliability of software systems has become a major concern. Today software is being deployed in safety applications due to the advancement of technology. In nuclear power plants (NPP), many systems are being used in safety critical and safety related applications, which demand a very high reliability [1]. As software becomes an increasingly important part of many different types of systems that perform complex and critical functions in many applications, such as defense, nuclear reactors, etc., the risk and impacts of software-caused failures have increased. There is now a general agreement on the need to increase software reliability by eliminating errors made during software development and maintenance.

Software is a collection of instructions or statements in a computer language. It is also called a computer program, or simply a program. A software program is designed to perform a set of specified functions. Upon execution of a program, an input state is translated into an output state. An input state can be defined as a combination of input variables or a typical transaction to the program. When the actual output deviates from the expected output, a failure occurs. It is estimated that 60-90% of current computer errors is from software faults [2].
Software reliability is defined as the probability of failure-free software operations in a specified environment [3]. The software reliability field discusses ways of quantifying it and using it for improvement and control of the software development process. A number of standards have emerged in the area of developing reliable software consistently and efficiently [4]. The Software Engineering Institute has established a standard called the software Capability Maturity Model (CMM) that scores organizations on multiple criteria and gives a numeric grade from one to five.

The software faults are most often caused by the requirement and design faults. The requirement fault can be incomplete requirement or interpreted in different or wrong way. The incomplete / missing of requirement may be covered under “Adequacy” check. An ambiguous statement may lead to wrong interpretation. It generally happens because of user’s “implicit specification”, i.e., user assumes it is obvious. Design faults occur when a designer either misunderstands a specification or simply makes a mistake. Software faults are common for the simple reason that the complexity in modern systems is often pushed into the software part of the system. Software reliability is operationally measured by the number of field failures, or failures seen in development, along with a variety of ancillary information. The ancillary information includes the time at which the failure was found, part of the software where it was found, the state of software at that time, the nature of the failure, time of deployment, etc. Most of the software quality improvement efforts are triggered by lack of software reliability. Thus, software managers recognize the need for systematic approaches to measure and assure software reliability, and devote a major share of project development resources to this. Formally, software reliability engineering is the field that quantifies the operational behavior of software based systems with respect to user
requirements with bearing on reliability. It includes data collection on reliability, statistical estimation, metrics and attributes of product architecture, design, software development, time of deployment and the operational environment. Besides its use for operational decisions like deployment, it includes guiding software architecture, design, development and testing. Most of the testing process is driven by software reliability concerns, and applications of software reliability models are to improve effectiveness of testing [5].

II. MOTIVATION

From the important software disasters, it is clear that software errors cost the country economy in rework, lost productivity and actual damages. Faulty software can also be expensive, embarrassing, destructive and deadly. It is well recognized that assessing the reliability of software applications is a major issue in reliability engineering [6]. Prediction of software reliability is highly involved. Perhaps the major difficulty is that we are concerned primarily with design faults, which is a very different situation from that tackled by conventional hardware theory. The input values to the software modules (functions) either internally or externally may be considered as arriving to the software randomly. So although software failure may not be generated stochastically, it may be detected in such a manner. Therefore, we can use stochastic models of the underlying random process that governs the software failure [7]. Hence, for safety critical software / highly dependable software systems the estimation of software reliability becomes prime important to evaluate the overall system reliability with the data available on hardware reliability. The hardware and software reliability together becomes more meaningful and useful for predicating the system performance and availability [8].
III. OBJECTIVES AND SCOPE OF THE PRESENT WORK

The main objective of the research is the estimation of software reliability for safety systems of Nuclear Power Plants. In case of NPP, the safety is of main concern and systems use mainly “Structured Programming” for safety systems instead of Object oriented or Commercially off the shelf (COTS) components [9].

In the process of assessment of reliability, the following sub-objectives are envisaged since software reliability is more concerned with design, methodologies, practices and tools used in the process of software development.

The sub-objectives are

(a) Development of software life cycle model for safety systems.
Assessing various life cycle models and the identification of a suitable software life cycle model for safety systems, which ensures highly reliable software delivery is one of the tasks of the current research.

(b) Determination of software metrics and development of software metric tool.
The identification of the software metrics that affect the reliability in terms of quality attribute and the development of tool to evaluate the metrics from the software code is very important.
(c) Development of Software Verification and Validation (V&V) methodology.

To evolve a comprehensive methodology to be followed for V&V throughout the life cycle to ensure quality artifacts are generated. The standards to be practiced and the detailed audit procedures and the checklist for each stage of the development, the role of V&V members and their independence in terms of evaluation are explained. This methodology is for the software fused in custom built computer based systems (CBS). This V&V procedure is developed from IEEE standard [10], NPCIL procedures on C&I system [11] and V&V procedure [12] so that it is suited for nuclear power plants.

(d) Development of Software Reliability Model.

Software reliability measurement includes two types of models namely static and dynamic reliability estimation, used typically in the earlier and later stages of development respectively. A key use of the reliability models is in the area of when to stop testing. One purpose of reliability models is to perform reliability prediction at an early stage of software development. This activity determines future software reliability based upon available software metrics and measures. Particularly when field failure data are not available (e.g. software is in the design or coding stage), the metrics obtained from the software development process and the characteristics of the resulting product can be used to estimate the reliability of the software upon testing or delivery.

(e) Procedure for Estimation of Software Reliability.

As a part of the research, a comprehensive procedure to estimate software reliability for a given software program coded in “C” language has been established. Robustness and
validation of the methodology has been demonstrated by applying it to software deployed in safety systems of a fast reactor.

The above said sub-objectives form the basis for the estimation of software reliability for safety systems of NPP. The current work is specifically for the software development on Instrumentation & Control systems employed in NPP. Due to large human contribution in the software development, the reliability calculation will have significant uncertainty. The variations among individuals need to be properly accounted in software development in particular during code development. The unreliability attached with Human Error Probability is assumed to be minimal which is basically the boundary within which the exercise has been carried out.
REFERENCES


