Chapter 3

Problem Formulation and Solution Strategies

The formulation of the research problem based on the extensive literature review and various planned solution strategies for the identified problems is presented in this chapter. The limitations of existing approaches are brought out in the previous chapter, chapter 2. The uncertainties along with their treatment, in early prediction of software reliability are discussed. The solution strategies for the stated limitations in chapter 2 regarding the early prediction of software reliability are also discussed.

3.1 Introduction

The Control and Instrumentation systems are widely based on software in today's era and software is the root cause of most of today's system problems. Failures occur because of the occurrence of fault in software. Reliability assurance or assessment attempts to study, characterize, measure, and analyze the failure and repair (in case the system is repairable) of the systems in order to increase their quality and longevity, thereby minimizing the catastrophic failures. The early assessment of reliability of software systems is an effective strategy for the management of risk and decision making for safe, economical and efficient design and operation of safety critical computer based systems like medical systems, aeronautical systems, NPPs and defense systems.

There are six phases in SDLC, namely requirements analysis, software design, implementation, testing and deployment and maintenance. Requirements analysis, software design and testing are the vital tasks. The fault in any of these phases may lead to an unreliable product, which can lead to catastrophic disasters. The fault can get embed in any of the vital phases of SDLC, starting from requirements analysis, software design, software implementation to software testing and hence there are the techniques through which software reliability can be assured in all these phases. Typical procedure to compute the reliability estimates is shown in figure 3.1.

In spite of potential benefits of these approaches, the uncertainties associated with models, parameters, phenomena and assumptions put limitation on its usage in the industries. From our studies, we conclude the following necessary requirements that need to be addressed. These requirements lie in the scope of these three important phases of SDLC: (i) requirements analysis phase (ii) software design phase (iii) testing phase.

Researchers, academicians and practitioner engineers are continuing to propose models for reliability prediction of software system during architectural design. Some of these models are based on the Markov chain. There is randomness or uncertainty involved in Markov reliability models. These uncertainties include the verification of incorporation of all the requirements in the model, verification of the correctness of the model itself even after all the requirements have been taken care of.

Next, the existing Markov reliability models, at the level of architectural design, are incapable to model the probability of changes in the system other than exponential distribution but the current software systems need not follow such restriction. Further the uncertainty in input parameters is involved in such models, which are transition probabilities in between the states of the Markov chain, of which the reliability of the system is a function.

Further, the reliability of a software system is the function of the reliabilities of all the composed components. To take preventive action during the architectural phase, the impact analysis of every component's reliability on the reliability of its associated components and the overall system. This also helps in taking corrective action, if there

is a provision to know the reliability change of any component of the software system during operational phase. Reliability depends on the failure rate, hence may change during operational phase.

3.2 Problem Formulation

The issues that are identified during different stages of SDLC, for probabilistic models of reliability assessment of a software system and for reliability estimates, need proper treatment to ensure the reliability of the software system. After critical literature review regarding the software reliability prediction, the research problem is four folded as shown in figure 3.1, explained in the subsequent sections.

3.2.1 Uncertainty in Markov reliability models

Software faults are likely to get embedded because of ambiguous, inconsistence and incomplete requirements, leading to improper design and implementation and the end result will be a unreliable software system. The unreliability in the software system may lead to the catastrophic failures. Hence, an early prediction reliability model must contain precise and complete requirements, which should be validated by the client who is the source of requirements and should be explainable to all the other stakeholders. The stakeholders have different skill set and hence there should be a common language to model the software system reliability for early prediction.

3.2.2 Uncertainty in input parameters in Markov reliability models and their limitations

Many early reliability prediction models exploit the architectural design making use of Markov chain for the purpose. Since Markov reliability model contains two types of states, behavioral and failure, the reliability of the software system can be represented as a function of the transition probabilities in between the states of the Markov chain. In all the existing approaches, the authors have either assumed these probabilities based

on some assumptions or computed them using the corresponding operational profile [23-27]. The computation of transition probabilities based on assumptions is known as an analytical approach and fails to give accurate reliability prediction. On the other hand, it may not be possible to collect an operational profile before the final development of the software. Hence transition probabilities, based on operational profile may not be useful in providing an early estimate of reliability. In addition to it, it may not be possible to model concurrency with Markov modeling. Further, Markov Model analysis is typically limited to modeling the probability of changes in the system with the exponential distribution, while it may not necessarily applicable to all kinds of software. Hence the perfection of this modeling approach to cover these deficiencies is essentially required.

3.2.3 Impact analysis of component reliability on the reliability of a software system for preventive action

A software system is composed of many components. Each component performs some function and contributes to the overall functioning of a system. The criticality of these functions need not necessarily be same and hence the reliability requirements of all the components also need not necessarily be same. During the early prediction, it is useful to know the sensitivity of the component's reliability on the overall system reliability as well as its associated components' reliability to take preventive design action like incorporating redundant components, diverse components.

3.2.4 Updation of components reliability during testing / operational phase for corrective action

Some safety critical systems are operational round the clock and hence there is a need to assess the failure possibilities of such a software system during operation. It is possible only if we keep learning from the actual operational profile in terms of failure data of components of the software system in operation. In case something can be done for improving the reliability (i.e. minimizing the failure probability) of critical components

then the survivability of the system can be ensured through such improvements. Further, through impact analysis, it will be possible to assess the reliability of the overall software system to verify whether it meets with the target reliability and take corrective action accordingly like repair, replacement of the faulty software components. Noticeably, it will be very useful if a software system contains COTS components, whose reliability assessment is not possible during early phases of SDLC.

3.3 Solution Strategies

3.3.1 Strategy for dealing with the uncertainty in Markov reliability models

For the requirement of modeling the software system reliability for early prediction, that should be explainable to all the stakeholders, an approach is proposed to support software reliability engineering from requirements to deployment level, through proper analysis and suitable mappings. UML has proved to be a general-purpose modeling language in the field of software engineering that can be understood by all the stakeholders [28]. UML has an ability to model the scenarios of the system. Authors have extended the UML to incorporate dynamic aspects for Schedulability, Performance and Time Specification [29]. Based on this, it is possible to extend the UML further to make it appropriate for reliability modeling for software reliability quantification. A method to predict system reliability from scenario specifications is proposed, by transforming them into behavioral models for each system component, which is then extended to model the probabilistic occurrence of component failure. The resultant reliability model is a Markov model, which can predict the overall reliability of the system according to Cheung's user-oriented reliability model [25]. The proposed approach is validated on a control logic component of a running safety critical system of Indian Nuclear Power Plant. An empirical sensitivity analysis of the system reliability is also carried out, which is a function of (i) the components' reliability and (ii) the transition probability between scenarios for this case study.

3.3.2 Strategy for dealing with the uncertainty in Markov reliability models and their limitations

To overcome the limitations of Markov reliability model, Petri net is used to model the software system. Petri net violates the restriction to follow the exponential property of Markov model. This is required because every software system does not necessarily follow the exponential property. Also, modern software systems require a rigorous analysis to ensure their performance because of their complex nature. The complexity lies in their nature of concurrency, synchronization, communication and cooperation among subsystems. It is possible to model these complex software systems through Petri net, rather than Markov chain. A tool TimeNET[30, 31] is used for steady state distribution of the transition rates in between the states of Markov model, through which the transition probabilities have been derived. To validate the proposed approach, a case study of running safety critical software system, known as Test Facility is taken. The case study is explained in the appendix 1. The reliability is quantified, based on predicted transition probabilities and compared with the computed reliability using Ramamoorthy and Bastani [32] model, which is well suited for safety critical systems. The dependability studies are commonly centered on reliability evaluation. When prediction of reliability of software is done the requirements document and architectural propositions are the only things that are available for the purpose. In case of software system that is already implemented the test data available can be utilized for estimation of reliability. This measure gives a confidence in the system in terms of its failure behavior in the future.

3.3.3 Strategy for dealing with impact analysis of component reliability on the reliability of a software system for preventive action

An approach based on Bayesian theory is proposed to do the impact analysis of the reliability of any components on its associated components and hence on the overall software system. The software components in a software system can be arranged in series or in

parallel. The proposed methodology works on all kinds of software systems, irrespective of the arrangements of the software components; however, it has been demonstrated on a series system, on the same case study, Test Facility of Indian NPP.

3.3.4 Strategy for updation of components reliability during testing/operational phase for corrective action

When the state of any software component changes, the state of other software components and hence the state of whole software system can be updated using BN through backward propagation. The change in the component's reliability is informed using this approach, as new information of reliability of one or more node becomes available, on the basis of the test or operational profile data. The proposed method is experimentally validated taking Test Facility of Indian NPP, as a case study.

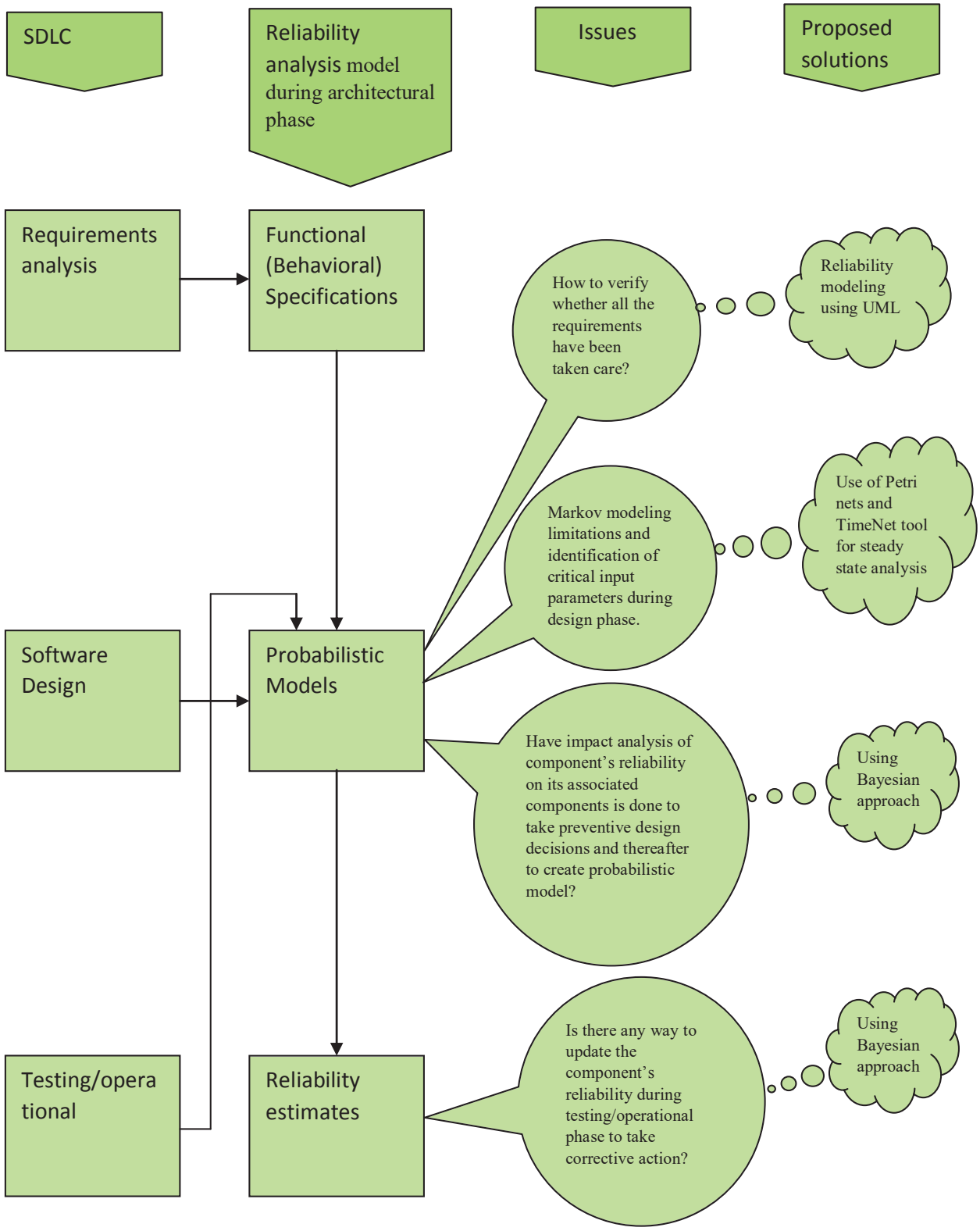

Figure 3.1: Issues and proposed solutions for uncertainty in probabilistic models of software reliability and its estimates.