PREFACE

Rapid advances in computers and network technologies have expanded the size and complexity of real-time software systems. With this almost exponential growth of computing technology human dependence on software has exceeded way beyond the limits of failure tolerance.

Software Reliability, a major software quality attributes, quantifies the probability of correct service delivery. In current practice, conventional software reliability models are parameter-based mathematical or statistical models. The models estimate software reliability using post-failure data along with assumptions regarding the nature of software failure process. However, estimates by these models are often far from the actual software usage. Additionally, despite the vast work in the field of software reliability assessment, much research is still required to ensure accurate software reliability estimation. Despite many varied models for software reliability estimation, software failures are no exception.

Addressing the above software reliability concerns, we postulate that software reliability challenge can be overcome by using actual runtime software specifications to control software execution and hence software reliability. Prevalent conventional software reliability models are black-box oriented techniques for software reliability modelling. These models though mathematical and statistical creations, are improper to model the breakdown of current component-based, heterogeneous software.

Software Reliability as opposed to Hardware Reliability is the operational reliability of a program. We thus require a model that can help ensure correct software operation. To achieve this goal predicting reliability of software taking into consideration their runtime architecture has become absolutely essential.

Research presented in this thesis is an extension of the Finite State Machine (FSM) concept to the probabilistic framework. We contribute a novel automata-based software reliability estimation model that takes into account the fact that any software during execution is an automaton. The model then uses the
characteristics of the automaton to track and control software execution at runtime. By doing this, we contribute towards systematically integrating reliability tracking and control from the early to the late stage of software life cycle.

In the process of establishing the need of the proposed model, we also discuss how conventional software reliability estimation models are inappropriate for actual software. We also show how the assumptions of the reliability models cause problems when applied to software.

This thesis proposes a technique to reliability that makes it probable, if not currently probable, to monitor software execution so as to precisely and safely establish software reliability.

Additionally we enable integration of the theory of software reliability with the formal theory of automata. By means of this mapping the reliability of a software application can be controlled during each execution instance. Further this information can be utilized for feasibility analysis of software during any phase of software development. To facilitate acceptance of this reliability control model over its conventional counterparts, we also discuss different conventional software reliability estimation models. We critically evaluate the characteristics of these traditional reliability estimation methodologies and mark the reasons why they have failed in accurate software reliability estimation.

Finally we evaluate our software reliability control tool with a case study on Java-Based Line and Chart Generator application. The purpose of this case study is to evaluate the efficacy of our analysis technique and compare it with other software reliability estimation approaches.