

Preface

Software systems are at once the most complex and the least reliable technological systems human beings construct. A large software system can have over 10^{20} states, and the reliability of software is infamously poor. Software engineers must usually make assertions about the reliability of software systems after having observed only an insignificant fraction of the possible states of the system. New mechanisms and techniques for inferring the overall quality and reliability of software systems are needed. In this book, we will describe three investigations into the use of computational intelligence and machine learning for software quality assurance, which lead toward such mechanisms.

Our first contribution is the use of chaos theory for software reliability modeling. Software reliability growth models (SRGM) are used to gauge the current and future reliability of a software system. Virtually all current SRGMs assume that software failures occur randomly in time, an assumption that has never been experimentally tested despite being criticized by a number of authors in the field. We have used nonlinear time series analysis to ascertain whether software reliability data from three commercial software projects come from a stochastic process, or from a nonlinear deterministic process. Evidence of deterministic behavior was found in these datasets, lending support to the idea that software failures may be *irregular* in nature. This is a qualitatively different form of uncertainty than randomness, one that is best modeled using the techniques of fractal sets and chaos theory rather than probability theory.

Our second contribution is the use of fuzzy clustering and data mining in software metrics datasets. Software metrics are measures of source code, which are intended as a basis for software quality improvement. Literally hundreds of metrics have been published in the literature, but no generally applicable regression model relating metrics and failure rates has been found. Instead of statistical regression, we use unsupervised machine learning, in the form of the fuzzy c-means algorithm, to analyze three collections of software metrics from commercial systems. This investigation highlights additional challenges for machine learning in the software metrics domain, one of which is skewness. The most common machine learning approach to overcoming skewness is to resample the dataset; however, this has never been attempted in the software metrics domain. Hence, our third contribution is the use of resampling algorithms to calibrate a decision tree to preferentially recognize high-risk classes of modules. We consider how the calibration process, as well as the operational decision tree, can be woven into an iterative software development process.

This book will primarily be of interest to researchers in the areas of computational intelligence or software engineering, and particularly those interested in interdisciplinary research between those two fields. It will also be suitable for use as a textbook in an advanced graduate class in either field, or to practicing software engineers interested in how computationally intelligent technologies may be used to aid their work. The original research material in this book is supported by an extensive review of both software engineering and computational intelligence, covering over 300 references.

Scott Dick
Abraham Kandel

February 2005