<<一种推导广义软件度量模型的方法>>

图书基本信息

书名：<<一种推导广义软件度量模型的方法>>

13位ISBN编号：9787811353792

10位ISBN编号：7811353792

出版时间：2009-8

出版时间：暨南大学出版社

作者：陈嘉贤

页数：277

内容概要

dilates on its underlying research which aimed to innovate a new general methodology for deriving generalized software metrics models from past empirical software metrics data. These genera/ised software metrics models are to predict the target software metric（s） of any future software project from the project's predictor software metric（s） always in a best-effort, best-accuracy and best-consistency manner whether all, only some or even none of these required predictor software metric（s） is/are available for that future project and/or projects in the past empirical software ware metrics data. A software project's predictor software metric（s） indicate（s） or measure（s） the project's relevant software engineering factor（s）. This' general methodology was illustrated in the research by deriving, from the empirical software metrics data of past software projects sourced in this research, two real-world generalized software metrics models with the productivity and work effort measures as the target software metrics respectively. These two real-world generalized software metrics models' prediction accuracy and consistency were also assessed in this research. Additionally done was an analysis of the impact of the various aforesaid software engineering factor（s） of software projects on the projects' productivity and work effort. All these works are presented in this book.

<<一种推导广义软件度量模型的方法>>

书籍目录

Table of ContentsForewordTable of ContentsList of IllustrationsList of TablesAcknowledgementsAbbreviations1 Introduction 1.1 Software Metrics 1.2 Software Metrics Models 1.3 Importance of Software Metrics Models 1.4 Existing Software Metrics Models 1.4.1 Function Point Analysis 1.4.2 Inferential Statistics 1.4.3 Neural Networks 1.4.4 Fuzzy Logic Systems 1.4.5 Hybrid Neuro-fuzzy Systems 1.4.6 Rule-based Systems 1.4.7 Case-based Reasoning 1.4.8 Regression and Classification Trees 1.5 Generalisedversus Existing Software Metrics Models 1.5.1 For Developers of Software Metrics Models 1.5.2 For End-users of Software Metrics Models 1.6 Overview of the Approach of this Research2 Terminologies and Typography 2.1 Terminologies 2.2 Typography3 The Objectives of the Research4 Summary of the General Methodology Innovated in this Research5 Theories Underlying the General Methodology Innovated in this Research 5.1 The "EM Algorithm fbr the General Location Model". 5.1.1 Theory 5.1.2 Application to this General Methodology 5.1.3 What Ifa Future Project Has No Missing Values 5.2 Transformation on the Continuous Software Metrics and Testing Multivariate Normality 5.2.1 Power Transformations on Individual Continuous Software Metrics 5.2.2 Testing Univariate Normality of the Individual Continuous Variables 5.2.3 Testing Multivariate Normality of the Continuous Variables 5.3 Detection and Elimination of the Multivariate Outlier(s) in Respect of the Continuous Variables. 5.3.1 Theory 5.3.2 Application to this General Methodology 5.4 Linear LS Regression of Each Continuous Independent Variable on All Other t2ontmuous IndependentVariable(s) 5.4.1 Theory 5.4.2 Application to this General Methodology 5.5 Coefficient of Determination R2 for the Linear LS Regression of the Dependent Variable on the Continuous Independent Variable(s) 5.5.1 Theory 5.5.2 Application to this General Methodology 5.6 Data Splitting, Mean Magnitude of Relative Error and Pred 5.6.1 Theory 5.6.2 Application to this General Methodology 5.7 The Bootstrap Method 5.7.1 Bootstrap Procedures 5.7.2 Bootstrap Analysis 5.7.3 Confidence Intervals 5.7.4 Test of Hypotheses 5.8 Plots of the Residuals versus the Predicted Dependent Variable6 Findings and Results 6.1 Stage 1 : Data Sourcing 6 1.1 Data Sources 6.1.2 ISBSG 6.1.3 Content of the ISBSG 6 Data 6.2 Stage 2: Rectification of Software Metrics Data 6.2.1 Short listing Software Metrics 6.2.2 "Filtering" Software Projects 6.2.3 Transformation on the Continuous Software Metrics and Testing Multivariate Normality 6.2.4 Detection and Elimination of the Multivariate Outlier(s) in Respect of the Continuous Variables 6.3 Stage 3: Constructing the Candidate Models 6.3.1 For the Intended Model with the PDR as the Target Metric 6.3.2 For the Intended Model with the "Summary Work Effort" as the Target Metric.. 6.4 Stage 4: Selecting and Optimizing Candidate Models 6.4.1 For the Intended Model with the PDR as the Target Metric 6.4.2 For the intended Model with the "Summary Work Effort" as the Target Metric.. 6.5 Stage 5: Analysis of Software Engineering Factors 6.5.1 For the Intended Model with the PDR as the Target Metric 6.5.2 For the Intended Model with the "Summary Work Effort" as the Target Metric.7 Discussion on the Findings and Results 7.1 Limitations of the Findings and Results 7.1.1 Unavoidably Biased Sampling 7.1.2 Unavailability of"Ideal" Software Metrics 7.1.3 Evolving Software Engineering/Development Technologies, Tools and Equipment 7.1.4 Extrapolation 7.1.5 No Causality Relationship Established 7.2 Comparision between the Existing Software Metrics Models and Generalised Models 7.2.1 Empirical Prediction Accuracy and Consistency of the Existing Software Metrics Models 7.2.2 Empirical Prediction Accuracy and Consistency of the Generalised Models 7.2.3 Summarising the Comparison 7.3 Foreseeable Improvement Areas lbr the GeneralisedModels of this Research 7.3.1 Complexity Measurement 7.3.2 Number of Bootstrap Samples 7.4 Other Comments on the GeneralisedModels8 ConclusionAppendix A The Sweep OperatorAppendix B Listing of the Scripts Implemented in this ResearchReferences

章节摘录

1 Introduction 1.1 oftware Metrics Based on the work of Fenton in 1991 and summarised by MacDonell and Gray in 1997, software metrics are measurements relating to a software system (i.e. the "product") or to the process of developing the system (i.e. the "process"). Examples of "product" software metrics are the size of a system (perhaps in the number of lines of code or the number of screens and reports), the number of defects in a system remaining after testing, some measure of the complexity of a system, etc. Examples of "process" software metrics may include the number of developers involved in the software project to develop the system, the work effort required for various stages of the project, and the experience of the developers, etc. Note that the number of screens or reports is usually classified as a functionality-based measure.

Traditionally, such software metrics were used as part of a formally specified model, such as "function point analysis" (Garmus and Herron 1995 and 1.4.1), which could be calibrated to a specific organisation and/or environment. Alternatively, they were used as variables in other models that established relationships between them for different software systems or projects. Regression equations are typical examples of these models. These relationships relate a target software metric (to be defined in 2.1) or target metric, in short, with one or more predictor software metric (s) (to be defined in 2.1) or predictor metric (s), in short. For example, the work effort (e.g. in the number of programmer hours) required for testing a particular system or a series of its modules might be the dependent variable in a regression equation with a functionality-based measure of the size of the concerned system, the systems complexity and the developers experience as the independent variables. All such models of this type are known as software metrics models. Note that "dependent variable" is a term in statistical regression corresponding to the target metric here whilst "independent variable" is a term in statistical regression corresponding to the predictor metrics here. 1.2 Software Metrics Models Means to derive software metrics models relating various software metrics have been sought since the advent of the study of software metrics (Zhou and Leung 2006, 1988, Shepperd 1988, Gremillion 1984 and DeMarco 1982). Typically, these models relate the work effort required for software projects to develop different software systems with a functionality-based measure of the sizes of the systems, their complexities and their developers experience. Once derived, software metrics models can be used to predict one or more target metrics, given the predictor metrics. The aforesaid work effort is a typical example of target metrics while typical examples of predictor metrics may include the aforesaid functionality-based measure of the sizes of the systems, their complexities and their developers experience. In his inspired assertion of software science (Halstead 1977) which was unfortunately seen subsequently as somewhat flawed, Halstead, one of the founders of software measurement, included an equation to predict the work effort as the target metric for program development. The equation was based on the fundamental algorithm size which itself composed several predictor metrics. Software metrics models have become a pivotal mechanism to control aspects of the "process" of the project to develop a system. Examples of these aspects may include the work effort, resources, costs required to develop the system, etc. Software metrics models should have real-world practical values and uses to software proiects.

版权说明