

Berichte aus der Wirtschaftsinformatik

Michael Grottke

**Modeling Software Failures
during Systematic Testing**

The Influence of Environmental Factors

D 29 (Diss. Universität Erlangen-Nürnberg)

Shaker Verlag
Aachen 2003

Bibliographic information published by Die Deutsche Bibliothek

Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the internet at <http://dnb.ddb.de>.

Zugl.: Erlangen-Nürnberg, Univ., Diss., 2003

n2

Copyright Shaker Verlag 2003

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN 3-8322-2035-6

ISSN 1438-8081

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen

Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9

Internet: www.shaker.de • eMail: info@shaker.de

Acknowledgments

This work is based on the research results I attained while I was employed at the University of Erlangen-Nuremberg for working on the project PETS (Prediction of software Error rates based on Test and Software maturity results), partially funded by the IST (information society technologies) program of the European Union.

I am grateful to Professor Dr. Ingo Klein of the Chair of Statistics and Econometrics for letting me devote my time to this tightly-scheduled project; otherwise, meeting the milestones would not have been possible. He also gave me the academic freedom to focus on those approaches and techniques that I deemed most promising.

Furthermore, my thanks go to Professor Dr. Michael Amberg of the Chair of Information Systems III for accepting the task of second reviewer of this dissertation.

The European Union did not only finance my position as a research assistant, but it also enabled the acquisition of the latest literature as well as my participation at various conferences on software quality around the world. Through its support preliminary results of the PETS project could be discussed with international researchers.

I am indebted to the employees of the industrial partners Acron Communications (Prague), AS Arote (Tartu), imbus AG (Möhrendorf) and Procedimientos-Uno, SL (Málaga) for their willingness to endure my questions, my presentations and the additional burden of data collection. Their input ensured the practical relevance of the research carried out.

Moreover, I would like to extend my thanks to my colleagues at the Chair of Statistics and Econometrics, especially for their understanding concerning my intermediate position between research and practice, or between project work and affiliation with the Chair.

Special appreciation is offered to PD Dr. Susanne Rässler of the Chair of Statistics and Econometrics and to Dipl.-Inf. Thomas Rossner of imbus AG for their valuable comments on chapters of my manuscript.

Finally and most importantly, I wish to thank my parents and my brother for their encouragement over the past few years. Without their support this work could not have been written.

Michael Grottke
September 2003

Contents

List of figures	viii
List of tables	x
Frequently used notation	xi
1 Introduction	1
2 Software reliability models and testing	3
2.1 Introduction to the chapter	3
2.2 Important concepts in software reliability engineering	4
2.3 Operational versus systematic testing	6
2.3.1 Operational testing	6
2.3.1.1 Brief method description	6
2.3.1.2 Advantages and disadvantages of operational testing	7
2.3.2 Systematic testing	9
2.3.2.1 Black-box techniques	10
2.3.2.2 White-box techniques	10
2.3.2.3 Advantages and disadvantages of systematic testing	13
2.4 A survey of software reliability growth models	14
2.4.1 Jelinski-Moranda model	14
2.4.2 Goel-Okumoto model	15
2.4.3 Models with a time-varying testing effort	16
2.4.4 Musa basic execution time model	18
2.4.5 Characteristics of the fault exposure ratio	19
2.4.6 Musa-Okumoto model	21
2.4.7 Enhanced non-homogeneous Poisson process framework	22
2.4.8 Approach by Piwowarski et al.	24
2.4.9 Rivers-Vouk model	26
2.5 A model framework	30

2.6	Testing techniques and structural coverage	33
2.7	Concluding remarks	35
3	The partial redundancy models	37
3.1	Introduction to the chapter	37
3.2	The basic partial redundancy model	38
3.2.1	Approximated expected structural coverage	39
3.2.2	The distribution of the number of code constructs covered	42
3.2.3	Exact expected structural coverage	46
3.3	Extended partial redundancy models	48
3.3.1	The first extended partial redundancy model	49
3.3.1.1	The approximated expected number of failure occurrences	50
3.3.1.2	The distribution of the number of failure occurrences	50
3.3.1.3	The exact expected number of failure occurrences	53
3.3.2	The second extended partial redundancy model	54
3.3.2.1	The approximated number of expected failure occurrences	55
3.3.2.2	The distribution of the number of failure occurrences	56
3.3.2.3	The exact expected number of failure occurrences	61
3.4	Including the models in the model framework	63
3.5	Model estimation	64
3.5.1	General remarks	64
3.5.2	Least squares estimation	66
3.5.2.1	Fitting the cumulative number of failure occurrences	66
3.5.2.2	Fitting the number of failure occurrences per test case	67
3.5.3	Maximum likelihood estimation	68
3.5.3.1	Interpreting the partial redundancy models as NHPP models	68
3.5.3.2	Maximizing the likelihood implied by the model setup	69
3.5.4	Staged estimation procedures	73
3.5.4.1	Stage 1: Estimating the basic partial redundancy model	73
3.5.4.2	Stage 2: Estimating the extended partial redundancy models	75
3.6	Analyzing model performance	77
3.6.1	The models and the estimation procedures	77
3.6.2	The data sets	78
3.6.3	The criteria	78
3.6.3.1	Measures of the deviations of predictions	78
3.6.3.2	Measures of noise	80
3.6.3.3	Measures based on the likelihood function	81

3.6.3.4	The estimated fraction of detectable faults discovered	82
3.6.4	The results	83
3.7	Concluding remarks	88
4	Software process maturity models	91
4.1	Introduction to the chapter	91
4.2	The Capability Maturity Model for Software	92
4.2.1	History	92
4.2.2	The reference model	93
4.2.3	How to conduct appraisals	97
4.2.4	Expected benefits of increased process maturity	98
4.2.5	Criticisms of the SW-CMM	99
4.2.6	Empirical evidence	100
4.2.6.1	Case studies and surveys	100
4.2.6.2	Statistical analyses of reliability	101
4.2.6.3	Statistical analyses of validity	102
4.3	Emerging standard ISO/IEC 15504	103
4.3.1	History	103
4.3.2	The reference model	104
4.3.3	An assessment model and how to perform assessments	108
4.3.4	Expected benefits of increased process capability	109
4.3.5	Criticisms of the emerging standard ISO/IEC 15504	110
4.3.6	Empirical evidence	110
4.3.6.1	Statistical analyses of reliability	111
4.3.6.2	Statistical analyses of validity	113
4.4	Assessing process capability in the PETS project	113
4.4.1	Basic decisions	113
4.4.2	Capability questions in the PETS questionnaire	115
4.4.3	Analyzing the reliability of the capability questions	117
4.4.4	Analyzing the validity of the capability questions	120
4.5	Concluding remarks	123
5	The influence of environmental factors	125
5.1	Introduction to the chapter	125
5.2	The data sets	127
5.3	Influence on the estimated fault density	129
5.3.1	Bivariate analyses of software development process capability	129
5.3.2	Bivariate analyses of other environmental factors	131

5.3.3	Multivariate analyses	136
5.4	Influence on the estimated redundancy level	140
5.4.1	Bivariate analyses	140
5.4.2	Multivariate analyses	147
5.5	Concluding remarks	152
6	Conclusions and outlook	153
A	Proofs and derivations	155
A.1	Mean value function of the model by Yamada et al.	155
A.2	Mean value function of the discrete Rivers-Vouk model	156
A.3	Recursive formulations of expected values	156
A.3.1	First extended partial redundancy model	156
A.3.2	Second extended partial redundancy model	162
A.4	Relationship between $E(M_i)$ and $E(\Xi_i)$	164
B	Information on the data sets	167
B.1	Project A	168
B.2	Project B	169
B.3	Project C	172
B.4	Project D	173
B.5	Project E	174
B.6	Project F	176
B.7	Project G	177
B.8	Project H	178
B.9	Project I	179
B.10	Project J	180
B.11	Project K	181
B.12	Project L	182
B.13	Project M	183
B.14	Project N	185
B.15	Project O	188
B.16	Project P	189
C	Tables of model performance measures	191
C.1	Project A	193
C.2	Project B	194
C.3	Project C	195

C.4 Project D	196
C.5 Project E	197
C.6 Project F	200
C.7 Project G	201
C.8 Project H	202
C.9 Project I	203
C.10 Project J	204
C.11 Project K	205
C.12 Project L	206
C.13 Project M	207
C.14 Project N	210
C.15 Project O	216
C.16 Project P	217
D Elements of semantic data modeling	221
E The PETS questionnaire	223
Bibliography	251

List of figures

2.1	<i>Models included in the continuous variant of the model framework</i>	32
2.2	<i>Models included in the discrete variant of the model framework</i>	33
2.3	<i>Expected structural coverage growth in the approach by Piwowski et al. and in the Rivers-Vouk model for $G=1000$ and $p=10$</i>	34
3.1	<i>Structure of the basic partial redundancy model</i>	39
3.2	<i>“Probability mass functions” of Q_4 for $G = 100$, $p = 10$ and $r = 1$ calculated via the closed form expression (l) and recursively (o)</i>	45
3.3	<i>Probability mass functions of Q_{100} for $G = 1000$, $p = 10$ and various redundancy levels r</i>	46
3.4	<i>Expected structural coverage growth in the basic partial redundancy model for $G = 1000$, $p = 10$ and various redundancy levels r</i>	48
3.5	<i>Structure of the first extended partial redundancy model</i>	49
3.6	<i>Probability mass functions of M_{100} according to the first extended model for $G = 1000$, $p = 10$, $u_0 = 100$ and various redundancy levels r and activation probabilities s</i>	53
3.7	<i>Structure of the second extended partial redundancy model</i>	54
3.8	<i>Probability mass functions of M_{100} according to the second extended model for $G = 1000$, $p = 10$, $u_0 = 100$ and various redundancy levels r and activation probabilities s</i>	59
3.9	<i>Expected number of failure occurrences in the first and second extended partial redundancy model for $G = 1000$, $p = 10$, $u_0 = 100$, $r = 0.9$ and $s = 0.5$</i>	62
4.1	<i>Maturity levels and related key process areas in the SW-CMM</i>	95
4.2	<i>Structure of the SW-CMM reference model</i>	96
4.3	<i>Expected benefits of increased software process maturity</i>	99
4.4	<i>Capability levels and related process attributes in the ISO/IEC TR 15504 reference model</i>	106
4.5	<i>Structure of the ISO/IEC TR 15504 reference and assessment models</i>	107

4.6	<i>Box-and-whisker-plot of the deviations of the mean scores from the normalized summative scores</i>	122
B.1	<i>Project A, failure data</i>	168
B.2	<i>Project B, "raw" failure data</i>	169
B.3	<i>Project B, testing effort and number of failure occurrences per test case</i>	170
B.4	<i>Project B, adjusted failure data</i>	171
B.5	<i>Project C, failure data</i>	172
B.6	<i>Project D, failure data</i>	173
B.7	<i>Project E (second and third integration test cycle), failure data</i>	174
B.8	<i>Project E (first system test cycle), failure data</i>	175
B.9	<i>Project E (second system test cycle), failure data</i>	175
B.10	<i>Project F, failure data</i>	176
B.11	<i>Project G, failure data</i>	177
B.12	<i>Project H, failure data</i>	178
B.13	<i>Project I, failure data</i>	179
B.14	<i>Project J, failure data</i>	180
B.15	<i>Project K, failure data</i>	181
B.16	<i>Project L, failure data</i>	182
B.17	<i>Project M, failure data</i>	183
B.18	<i>Project M, coverage data</i>	184
B.19	<i>Project N (all plug-ins), failure data</i>	185
B.20	<i>Project N (all plug-ins), coverage data</i>	186
B.21	<i>Project N (plug-in S only), failure data</i>	186
B.22	<i>Project N (plug-in S only), coverage data</i>	187
B.23	<i>Project O, failure data</i>	188
B.24	<i>Project P, failure data</i>	189
B.25	<i>Project P, coverage data</i>	190
D.1	<i>Cardinalities in the crow'sfoot notation</i>	221
D.2	<i>Example for a relationship type</i>	222
D.3	<i>Example for an inheritance relationship</i>	222

List of tables

3.1	<i>Unified functional forms of $\mu(i)$ and $E(M_i)$ for both extended partial redundancy models</i>	65
3.2	<i>Interpretation of the parameters in the unified functions</i>	65
3.3	<i>Mean ranks of the seven approaches based on failure data only</i>	84
3.4	<i>Mean ranks of twenty-two approaches, including staged procedures, omitting the estimates based on the first 4 data points</i>	86
3.5	<i>Mean ranks of twenty-two approaches, including staged procedures, omitting the estimates based on the first 15 data points</i>	87
4.1	<i>Number of projects (out of 14) for which the processes were not applicable or for which ratings were not available</i>	118
4.2	<i>Correlations matrix for the mean ratings of all applicable development processes</i>	119
4.3	<i>Processes evaluated in an assessment and with the PETS questionnaire</i>	120
5.1	<i>Parameter estimates obtained for the first extended partial redundancy model via the ML-NHPP procedure</i>	129
5.2	<i>Correlations between all process scores and (the logarithm of) the estimated fault density</i>	130
5.3	<i>Environmental factors related to software development and the software product</i>	132
5.4	<i>(Average) Correlations between the environmental factors related to software development and the product</i>	134
5.5	<i>(Average) Correlations between environmental factors and (the logarithm of) the estimated fault density</i>	135
5.6	<i>Fitted models of estimated fault density</i>	137
5.7	<i>Estimated fault densities according to the first extended partial redundancy model and average fitted values of the models (I) and (II)</i>	138
5.8	<i>Fitted models of the logarithm of the estimated fault density</i>	139
5.9	<i>Influence of testing capability on the estimated redundancy level</i>	143
5.10	<i>Environmental factors related to the test process</i>	144
5.11	<i>(Average) Correlations between the environmental factors related to testing</i>	146

5.12	<i>Influence of the environmental factors related to testing on the estimated redundancy level</i>	147
5.13	<i>Results of fitting the zero inflated binomial model</i>	151
C.1	<i>Explanation of entries in the tables of results</i>	191
C.2	<i>Model performance measures for project A</i>	193
C.3	<i>Model performance measures for project B</i>	194
C.4	<i>Model performance measures for project C</i>	195
C.5	<i>Model performance measures for project D</i>	196
C.6	<i>Model performance measures for project E (second a. third integration test cycle) 197</i>	
C.7	<i>Model performance measures for project E (first system test cycle)</i>	198
C.8	<i>Model performance measures for project E (second system test cycle)</i>	199
C.9	<i>Model performance measures for project F</i>	200
C.10	<i>Model performance measures for project G</i>	201
C.11	<i>Model performance measures for project H</i>	202
C.12	<i>Model performance measures for project I</i>	203
C.13	<i>Model performance measures for project J</i>	204
C.14	<i>Model performance measures for project K</i>	205
C.15	<i>Model performance measures for project L</i>	206
C.16	<i>Model performance measures for project M</i>	207
C.17	<i>Model performance measures for project N (all plug-ins)</i>	210
C.18	<i>Model performance measures for project N (plug-in S only)</i>	213
C.19	<i>Model performance measures for project O</i>	216
C.20	<i>Model performance measures for project P</i>	217

Frequently used notation

Functions and random variables

$\Gamma(x)$	gamma function
$\Delta^n f(x)$	n^{th} forward difference of $f(x)$
$\exp(x)$	exponential function
$E(X)$	expected value of random variable X
$F(x)$	distribution function
$I_A(x)$	indicator function for $x \in A$
$\ln(x)$	natural logarithm of x
\mathcal{L}	likelihood function
$P(H_1)$	probability of the event H_1
$P(H_1 H_2)$	probability of the event H_1 conditional on the event H_2
$P_{ m }$	probability of the occurrence of exactly m events
S_m	sum of the probabilities of the simultaneous occurrence of at least m events

Goodness of fit measures and other metrics

AIC_j	Akaike's information criterion calculated based on the first j observations
α	Cronbach's α
C	condition number
$d_{(p)}$	p -quantile of the observations of d
$d_{(0.5)}$	0.5-quantile of the observations of d (= median of d)
κ	Cohen's κ
$\lambda_{\min}, \lambda_{\max}$	smallest and largest eigenvalue of a matrix
R^2	coefficient of determination
R^2_{adj}	adjusted coefficient of determination
R^2_{AN}	Aldrich's and Nelson's pseudo R^2 measure
R^2_{MF}	McFadden's pseudo R^2 measure
R^2_{MZ}	McKelvey's and Zavoina's pseudo R^2 measure
ρ	empirical correlation coefficient

Miscellaneous

\mathbb{N}	integer numbers
\mathbb{N}_0	integer numbers including zero
\mathbb{R}_0^+	positive real numbers including zero
$\lfloor x \rfloor$	largest integer less than or equal to x
$\binom{n}{k}$	binomial coefficient = $\frac{n!}{k!(n-k)!}$
∞	infinity
\propto	proportional to

Abbreviations

4 GL	fourth generation programming language
CAF	CMM Appraisal Framework
CASE	computer aided software engineering
CBA IPI	CMM-Based Appraisal Framework for Internal Process Improvement
CM	configuration management
CMM	Capability Maturity Model
CPU	central processing unit
CUS	customer-supplier process category
DDIF	development difficulty
DEFF	development effort
DEPI	development effort performance index
DMSK	development team manager's skill level
DRPI	development runtime performance index
DRUN	development runtime
DTSI	size of the development team
ENG	engineering process category
FDEN	fault density
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
LS-Cum	Least squares estimation based on the cumulative number of failure occurrences
LS-Delta	Least squares estimation based on the number of failure occurrences per test case
MAN	management process category
MARE	mean absolute relative error
MIS	management information system

ML-NHPP	Maximum likelihood estimation based on the likelihood following from interpreting the model as a non-homogenous Poisson process model
ML-SetupC	Maximum likelihood estimation based on the likelihood implied by the model setup
ORG	organization process category
PAUT	proportion of automated test cases
PETS	Prediction of software Error rates based on Test and Software maturity results
PGSK	programmers' general skill level
PNDT	proportion of new members in the development team
PNTT	proportion of new members in the test team
PRCH	proportion of requirements changed after the specification phase
PRCO	proportion of reused code
PREJ	proportion of rejected failure messages
PSSK	programmers' specific skill level
PTED	proportion of testers with a special education as test engineers
QA	quality assurance
SARE	short term absolute relative error
SCAP	selective capability rating
SCE	Software Capability Evaluation
SEI	Software Engineering Institute
SICC	size of the compiled code
SPICE	Software Process Improvement and Capability dEtermination
SSE	error sum of squares
SST	total sum of squares
SUP	support process category
SW-CMM	Capability Maturity Model for Software
TCAP	testing capability rating
TDIF	testing difficulty
TEFF	testing effort
TEPI	testing effort performance index
TGSK	testers' general skill level
TMSK	test team manager's skill level
TRPI	testing runtime performance index
TRUN	testing runtime
TSSK	testers' specific skill level
TTSI	size of the test team

Software reliability models / software failure models

$c(\tilde{t})$	deterministic relative code coverage function
E	code construct state “eliminated”
EC	code construct state “eliminated and correct”
EF	code construct state “eliminated and faulty”
$g(\tilde{t})$	testing efficiency function
g_i	testing efficiency at the \tilde{i}^{th} stage of testing
G	total number of code constructs
$G_{A,i}$	number of code constructs that are located in state A after execution of the i^{th} test case
$G_{A,p,i}$	number of code constructs that are located in state A before the i^{th} test case execution and that are exercised by this test case
$G_{A \rightarrow B,i}$	number of code constructs residing in state A before the i^{th} test case execution and in state B afterwards
$G_{TF \rightarrow T,p,i}$	number of already tested, faulty code constructs that are tested and replaced during the i^{th} test case
$G_{TF \rightarrow TF,p,i}$	number of already tested, faulty code constructs that are tested and replaced without activating the fault during the i^{th} test case
i	number of test cases executed
i_j	test case at which the j^{th} measurement was taken
i_t	total number of test cases in the test plan
\tilde{i}	generic discrete measure of testing progress
K	fault exposure ratio
$\kappa(\tilde{t})$	expected relative code coverage function
$\lambda(\tilde{t})$	failure intensity function
$\lambda_{\tilde{i}}$	failure intensity at the \tilde{i}^{th} stage of testing
m_i	number of failures experienced / faults detected during the first i test cases
Δm_i	number of failures experienced / faults detected during the i^{th} test case
Δm_j^*	number of failures experienced / faults detected during the j^{th} observation period, i.e. between the $(i_{j-1} + 1)^{th}$ and the i_j^{th} test case
$M(\tilde{t})$	random variable denoting the cumulative number of failures experienced by time \tilde{t}
$M_{\tilde{i}}$	random variable denoting the cumulative number of failures experienced during the first \tilde{i} stages of testing
$\mu(\tilde{t})$	mean value function, $E(M(\tilde{t}))$
N	expected number of inherent faults
ν_d	expected number of inherent detectable faults

p	number of code constructs executed per test case
q_i	number of code constructs exercised during the first i test cases
Δq_i	number of code constructs exercised during the i^{th} test case
Δq_j^*	number of code constructs exercised during the j^{th} observation period, i.e. between the $(i_{j-1} + 1)^{\text{th}}$ and the i_j^{th} test case
$Q(\tilde{t})$	random variable denoting the cumulative number of code constructs exercised by time \tilde{t}
$Q_{\tilde{i}}$	random variable denoting the cumulative number of code constructs exercised during the first \tilde{i} stages of testing
r	redundancy level
s	fault activation probability
t	testing effort
t^*	calendar time
\tilde{t}	generic continuous measure of testing progress
T	code construct state “tested”
TC	code construct state “tested and correct”
TF	code construct state “tested and faulty”
τ	CPU execution time
u_0	number of inherent faults
U	code construct state “untested”
UC	code construct state “untested and correct”
UF	code construct state “untested and faulty”
$w(t^*)$	instantaneous testing effort at calendar time t^*
$W(t^*)$	cumulative testing effort until calendar time t^*
$X_{I,i}$	random variable denoting the number of faulty code constructs exercised at least once during the first i test cases
$\Delta X_{I,i}$	random variable denoting the number of faulty code constructs exercised for the first time by the i^{th} test case
$\Delta X_{II,i}$	random variable denoting the number of faulty code constructs exercised by the i^{th} test case
Ξ_i	random variable denoting the number of faulty constructs either corrected or eliminated during the first i test cases
z_i	probability with which a certain previously not eliminated code construct is executed by the i^{th} test case
$z_a(\tilde{t})$	per-fault hazard rate at time \tilde{t}
$z_{a,\tilde{i}}$	per-fault detection rate at the \tilde{i}^{th} stage
$z(\Delta \tilde{t} \tilde{t}_{n-1})$	hazard rate of the application after the $(n - 1)^{\text{th}}$ failure occurrence