Capture-Recapture Methods

Dr. Mark C. Paulk

SE 4367 – Software Testing, Verification, Validation, and Quality Assurance

Defects and Reliability

Defect prediction models

- **predict the number of defects in a module or system**
- **predict which modules are defect-prone**

Reliability models

- **predict failures (usually mean-time-to-failure MTTF)**

Who Uses?

Reliability models can be used during testing to determine where the software is ready to release.

Reliability models can be used to understand the quality of the operational software.

Defect prediction models are used during development.

- **by project management and the development team**
- **to focus effort on the parts of the system that need the most attention**
- **to understand the impact of selected processes, techniques, and tools on quality**

Predicting Reliability

Stochastic reliability growth models can produce accurate predictions of the reliability of a software system providing that a reasonable amount of failure data can be collected for that system in representative operational use.

Unfortunately, this is of little help in those many circumstances when we need to make predictions before the software is operational.

N. Fenton and M. Neil, "Software Metrics: Successes, Failures, and New Directions," The Journal of Systems and Software, July 1999.

Challenges in Using Defect Prediction Models

Difficult to determine in advance the seriousness of a defect

Great variability in the way systems are used by different users, resulting in wide variations of operational profiles

Difficult to predict which defects are likely to lead to failures (or to commonly occurring failures)

- **33% of defects led to failures with a MTTF greater than 5,000 years**
- **proportion of defects which led to a MTTF of less than 50 years was around 2%**

Be wary of attempts to equate fault densities with failure rates!

Explanatory Variables for Predicting Defects

Size measures (LOC)

Complexity measures

- **McCabe cyclomatic complexity**
- **Halstead software science: effort**
- **count of procedures**
- **Henry and Kafura's Information Flow Complexity**
- **Hall and Preisser's Combined Network Complexity**

OO structural measures (Chidamber and Kemerer)

Code churn measures

- **amount of change between releases**

Process change and fault measures

- experience
- number of developers making changes
- number of defects in previous releases
- number of LOC added/changed/deleted

Causal Factors for Defects

Difficulty of the problem

Complexity of designed solution

Programmer/analyst skill

Design methods and procedures used

N.E. Fenton and M. Neil, “A Critique of Software Defect Prediction Models,” IEEE Transactions on Software Engineering, September/October 1999.

Limits of Using Size and Complexity Measures to Predict Defects

Models using size and complexity metrics are structurally limited to assuming that defects are solely caused by the internal organization of the software design and cannot explain defects introduced because

- **the “problem” is “hard”**
- **problem descriptions are inconsistent**
- **the wrong “solution” is chosen and does not fulfill the requirements**

Techniques Used

Regression models

- **multicollinearity is a problem**

Factor analysis / principal component analysis

Bayesian belief networks

Artificial neural networks

Capture-recapture

Capture-Recapture

Uses the overlap between the sets of defects found by different reviewers to estimate residual defects

Assumptions
- **reviewers work independently of each other**
- **searching is performed before, and not during, an inspection meeting**

If the overlap is large, few defects are left to be detected.

If the overlap is small, many faults are undetected.

History of Capture-Recapture

First known use of capture–recapture was by Laplace (1786), who used it to estimate the population size of France

In biology, capture–recapture is used to estimate the population size of animals in an area

Lincoln-Petersen Method

$N = M C / R$

N – estimate of total population size

M – total number of animals captured and marked on the first visit

C – total number of animals captured on the second visit

R – number of animals captured on the first visit that were then recaptured on the second visit

Example

Capture 10 specimens on a first visit and mark them

Capture 15 specimens on a second visit

- **5 are marked from the first visit**

N = M C / R = (10) (15) / 5 = 30

Chapman Estimator

A less biased estimator for small samples

N = [(M + 1) (C + 1) / (R + 1)] – 1

var(N) = [(M + 1) (C + 1) (M – R) (C – R)] /
[(R + 1) (R + 1) (R + 2)]

Example

- **N = [(10 + 1) (15 + 1) / (5 + 1)] -1 = 29.3**
- **var(N) = (11*16*5*10) / (6*6*7) = 34.9**
- **std(N) = sqrt(34.9) = 5.9**

Capture-Recapture Models in Software Engineering

Basic model (M0) assumes that all faults are equally probable to be found and that all reviewers have equal abilities to find faults

Mh model – the probabilities of fault detection vary

Mt model – abilities of reviewers vary

Mth model – both the probabilities of fault detection and the abilities of reviewers vary

Capture-Recapture Estimators

M0

- **M0–ML – maximum likelihood (Otis, 1978)**

Mt

- **Mt–ML – maximum likelihood (Otis, 1978)**
- **Mt–Ch – Chao's estimator (Chao, 1989)**

Mh

- **Mh–JK – Jackknife (Burnham, 1978)**
- **Mh–Ch – Chao's estimator (Chao, 1987)**

Mth

- **Mth–Ch – Chao's estimator (Chao, 1992)**

Goodness of Capture-Recapture

For four reviewers and more, Mh–JK is preferable

Mt–Ch is the best estimator for two reviewers

Most models underestimate, but false positives inflate the estimate

Reinspections

If a reinspection is made, knowledge grows about the artifact

Biffl and Grossman (2001) approaches to using additional information

a) first combine the data from the inspections and then estimate
b) add the number of faults detected in the first inspection to an estimate of the reinspection
c) estimate the first inspection and the reinspection separately and then add their results

Best approach is (a), which improved estimators significantly

Criteria for Establishing Confidence in a Defect Prediction Model

Prediction criteria

- **Is a prediction model reported?**
- **Is the prediction model tested on unseen data?**

Context criteria

- **Is the source of the data reported?**
- **Is the maturity of data reported?**
- **Is the application domain of data reported?**
- **Is the programming language of data reported?**

T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "Developing Fault-Prediction Models," IEEE Software, November/December 2011.

Model criteria

- **Are the independent variables clearly reported?**
- **Is the dependent variable clearly reported?**
- **Is the granularity of the dependent variable reported?**
- **Is the modeling technique used reported?**

Data criteria

- **Is the fault data acquisition process described?**
- **Is the independent variables data acquisition process described?**
- **Is the faulty/nonfaulty balance of data reported?**

Performance of Defect Prediction Models *(Hall 2011)*

Precision – proportion of units predicted as faulty that were faulty

Recall – proportion of faulty units correctly classified

F-Measure – harmonic mean of precision and recall

- **(2 * recall * precision) / (recall + precision)**

Most models peak at about 70% recall.

Models based on naïve Bayes and logistic regression seem to work best.

Models that use a wide range of metrics perform relatively well.

- **source code, change data, data about developers**

Models using LOC metrics performed surprisingly well.

Successful defect prediction models are built or optimized to specific contexts.

Questions and Answers