December 2006

Analysis of Software Quality via a Goal Programming Approach

Young Chun
Louisiana State University

Follow this and additional works at: http://aisel.aisnet.org/amcis2006

Recommended Citation
http://aisel.aisnet.org/amcis2006/457
Analysis of Software Quality
via a Goal Programming Approach

Young H. Chun
Department of Information Systems and Decision Sciences
E. J. Ourso College of Business
Louisiana State University
Baton Rouge, LA 70803, USA
chun@lsu.edu

ABSTRACT
Complex software systems often fail because of errors undiscovered in the design stage of the development process. Detecting these errors early in the process would eliminate many downstream problems. The so-called capture-recapture model used by biologists in estimating the size of wildlife populations has also been used to estimate the number of software design errors. However, one simplifying assumption in capture-recapture models is that the inspections performed by various inspectors are statistically independent from each other. In the paper, we propose a novel method that is based on the correlation matrix of multiple inspectors. In a numerical analysis, we show that our method outperforms other traditional models that are based on the independence assumption.

Keywords
Software reliability, software inspections, capture-recapture model, quality management.

INTRODUCTION
Sometimes complex software systems fail because of errors or faults introduced in the design stage of the development process. Design reviews can remove some of these errors, but often a few errors remain undetected until the software is developed further. It is important to eliminate these design errors at an early stage because they become much more expensive to fix as a software system proceeds through development.

In the design stage, software engineers usually produce documents that are reviewed by their peers. Document reviews serve as the gateway to the next stage in the process. The current industry practice is for each document to be reviewed by a small group of software engineers before it passes to the next stage. The engineers individually read the document and note “issues” that they believe should be resolved before the software feature is developed further. We refer to these issues as “errors” or “fault” in the paper. At the review meeting, data are collected showing which software engineers discovered which faults. These data are then used to estimate the number of errors that remain undetected. These estimates can then be used in managing the software quality at the design stage of the development process.

Since it is impossible to count the total number of errors in a software system before it has been in operation for a while, it is necessary to build estimation models of the number of defects in a software artifact. One of the techniques that have been widely used to estimate the software design faults is a class of statistical methods known as “capture-recapture models”.

This technique is most commonly used by biologists in estimating the size of wildlife populations in wildlife research.

LITERATURE REVIEW
Capture-recapture models with two inspectors
Suppose that animals are captured, marked, and released on several trapping occasions. If an animal bearing a mark is captured on a subsequent trapping occasion, it is said to be recaptured. Based on the number of marked animals that are recaptured, one can estimate the total population size using statistical models and their estimators. When many marked animals are recaptured, for example, one can argue that the total population size is small, and vice versa.
For example, consider the problem of estimating the number of fish in a lake (Schnabel, 1938). A commonly used approach is to first catch a number \( n_1 \) of fish, tag or mark them, and release them again into the lake. Some time later, after giving the tagged fish an opportunity to mix well with the remaining fish in the lake, a second catch of \( n_2 \) fish is made, and the number \( n_{12} \) of tagged fish is observed. What is the total number \( N \) of fish in the lake?

From Table 1, one can derive an intuitive estimator of the population size \( N \) based on the assumption that the ratio of marked to total animals in the second sample \( (n_{12}/n_2) \) should reflect the same ratio \( (n_1/N) \) in the population. Thus, the so-called Lincoln-Peterson estimator of \( N \) can be derived as follows (Seber, 1982):

\[
\hat{N} = \frac{n_1 \times n_2}{n_{12}},
\]

From a formal statistical point of view, the Lincoln-Peterson estimator has the drawback of having an “infinite bias”, because there is always a finite probability that \( n_{12} = 0 \) (i.e., no marked animals are caught on the second occasion). Chapman (1951) proposed an adjusted equation to circumvent this drawback:

\[
\hat{N} = \frac{(n_1 + 1) \times (n_2 + 1)}{(n_{12} + 1)} - 1.
\]

Now the idea behind using capture-recapture models for software engineering inspections can be illustrated by the following puzzle (Wells, 1992, p. 146):

“Two proof readers are checking two copies of the same manuscript. The first finds thirty errors, and the second finds only twenty-four. When their completed proofs are compared, it turns out that only twenty errors have been spotted by both of them. How many errors would you suspect remain, not detected by either of them?”

Applying the Lincoln-Peterson estimator to the data given in the problem, we see that the estimate of \( N \) is 36, and thus the answer to the puzzle is 2.0 (=\( N-n_1-n_2+n_{12} \)). Note that the Chapman’s modification of the Petersen-Lincoln estimator in (2) gives \( \hat{N} = 35.90 \).

Readers are referred to Pollock (1991) and Seber (1986) for more detail about the modeling of capture, recapture, and removal statistics in the estimation of demographic parameters of fish and wildlife populations, and to Briand, El Emam, Freimut, and Laitenberger (2000) for a comprehensive analysis of capture-recapture models in the field of software inspections.

### Capture-recapture models with multiple inspectors

The basic methodology of capture-recapture models can be easily extended to the case in which there are more than two inspectors. Suppose that there are \( m \) inspectors, who work independently on identical copies of the product, or who work in some sequence on a single product, identifying, but not removing, the errors which they find. Suppose also that the inspection process is such that (1) each error present has the same probability of being detected by a given inspector, and (2) the probability that inspector \( i \) will find any given error is \( p_i \), independent of previous errors found by inspector \( i \) or by any other inspector.

Let \( n_i \) denote the total number of errors detected by inspector \( i \) \((i = 1, 2, \ldots, m)\), and \( u_i \) be the number of errors not detected by inspector \( i \) but detected by any of the remaining inspectors. Likewise, let \( n_{ij} \) denote the number of errors detected by both inspector \( i \) and inspector \( j \), and \( u_{ij} \) denote the number of errors detected by neither inspector \( i \) nor inspector \( j \) but detected by any of the remaining inspectors. Furthermore, let \( n_T \) be the total number of errors discovered by any of the inspectors and \( n_0 \) be the number of errors not discovered by any inspectors. Thus, the total number of errors \( N \) in the software design document is \( n_T + n_0 \) as shown in Table 2.

<table>
<thead>
<tr>
<th>First Catch</th>
<th>Marked</th>
<th>Unmarked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marked</td>
<td>( n_{12} )</td>
<td>( n_1-n_{12} )</td>
</tr>
<tr>
<td>Unmarked</td>
<td>( n_{2-n_{12}} )</td>
<td>( N-n_{1-n_{22+n_{12}}} )</td>
</tr>
</tbody>
</table>

Table 1. Capture-recapture history of fish in the lake
Chun

Analysis of Software Quality via a Goal Programming Approach

Proceedings of the Twelfth Americas Conference on Information Systems, Acapulco, Mexico August 04th-06th 2006

<table>
<thead>
<tr>
<th>Inspector</th>
<th>Detected</th>
<th>Undetected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n_{ij}</td>
<td>n_i - n_{ij}</td>
</tr>
<tr>
<td></td>
<td>n_{ij}</td>
<td>n_i + n_0</td>
</tr>
<tr>
<td></td>
<td>n_i</td>
<td>n_i + n_0</td>
</tr>
</tbody>
</table>

Table 2. Capture-recapture history in a multiple inspector case

The most commonly used estimator of \( N \) in the software design document is the maximum likelihood (ML) estimator \( \hat{N} \), which maximizes the following likelihood (Eick et al., 1993a):

\[
L(N) = \left( \frac{N}{n_T} \right)^m \prod_{i=1}^{m} p_i^{n_i} (1 - p_i)^{N-n_i}.
\]  

(3)

The exact ML estimator of \( N \) does not exist in closed form. The relevant log-likelihood function for estimating the parameters \((N, p_1, p_2, \ldots, p_m)\) is given by

\[
\ln L(N) = \ln \binom{N}{n_T} + \sum_{i=1}^{m} n_i \ln p_i + \sum_{i=1}^{m} (N-n_i) \ln(1 - p_i).
\]  

(4)

Given the value of \( N \), the ML estimators \( p_i(N) \) are obtained by taking the first-order derivative of (4) with respect to \( p_i \), setting it equal to zero, and solving it for \( p_i \):

\[
\hat{p}_i = \frac{n_i}{N}, \quad i = 1, 2, \ldots, m.
\]  

(5)

By replacing \( p_i \) in (4) with \( \hat{p}_i \) in (5), we have the following the log-likelihood function:

\[
\ln L(N) = \ln \binom{N}{n_T} + \sum_{i=1}^{m} n_i \ln \frac{n_i}{N} + \sum_{i=1}^{m} (N-n_i) \ln \left( \frac{N-n_i}{N} \right).
\]

\[
= \ln N! - \ln(N-n_T)! + \sum_{i=1}^{m} (N-n_i) \ln(N-n_i) - Nm \ln N + \text{constant}.
\]  

(6)

The ML estimator \( \hat{N} \) that maximizes (6) can be obtained numerically with the constraint \( N \geq n_T \). For a large value of \( N \), it is computationally more convenient to use the log-gamma function \( \ln \Gamma(N+1) \) in place of \( \ln N! \) in (6), assuming that \( N \) is a continuous variable; In the numerical analysis, we used the log-gamma function, GAMMALN( ), in Microsoft Excel 2000.

An alternative estimator of \( N \) is Darroch’s approximate ML estimator \( \tilde{N} \), which is the unique solution of the following equation (Darroch, 1958):

\[
1 - \frac{n_T}{N} = \prod_{i=1}^{m} \left( 1 - \frac{n_i}{N} \right), \quad \text{where } N \geq n_T.
\]  

(7)

The left side of (7) estimates the probability of not being able to detect an error using the fact that the group of inspectors as a whole discovered a total number of \( n_T \) errors. The right hand side of (7) estimates the same probability using the number of errors \( n_i \) detected by each individual inspector. It can be easily shown that, for the special case of \( m=2 \), Darroch’s estimator in (7) reduces to the Lincoln-Peterson estimator in (1).

Both the maximum likelihood estimator and the Darroch’s approximation method require a fast and iterative method, such as the Newton-Rapson method or the steepest-descent method, to find the optimal solution. However, Chao (1989) proposed an estimator of \( N \) that is in closed form as follows:
\[
\hat{N} = n_T + \frac{\sum_{i=1}^{m} \sum_{j=1}^{m} n_{i(j)} n_{j(i)}}{s_2 + 1},
\]

where \( n_{i(j)} \) is the number of defects detected only by inspector \( i \) and \( s_2 \) is the total number of defects that have been detected by exactly two inspectors. For the special case of \( m=2 \), Chao’s estimator in (8) reduces to the Chapman’s modification of the Petersen-Lincoln estimator in (2).

Like all other mathematical models, capture-recapture models in software inspection as well as in biology make certain simplifying assumptions so as to render the mathematics tractable. One simplifying assumption is that the inspectors are working independently from each other. The statistical independence between two inspectors can be formally explained as follow.

For a certain error, let the random variable \( X_i \) be defined as follows:

\[
X_i = \begin{cases} 
1 & \text{if detected by inspector } i \\
0 & \text{if undetected by inspector } i
\end{cases}
\]

Furthermore, let \( X_i \) be a Bernoulli random variable with parameter \( p_i \) as shown in Table 3. If and only if two inspectors are statistically independent, the joint probability \( p_{12} \) that the error is detected by both inspectors is expressed as \( p_{12} = p_1 p_2 \). If two inspectors are working together to find many errors in common, then \( p_{12} > p_1 p_2 \). If they have different specialties or educational backgrounds, then \( p_{12} < p_1 p_2 \).

<table>
<thead>
<tr>
<th>Inspector 1</th>
<th>Inspector 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_1 = 1 )</td>
<td>( p_{12} )</td>
</tr>
<tr>
<td>( X_1 = 0 )</td>
<td>( p_2 - p_{12} )</td>
</tr>
</tbody>
</table>

\[ p_1 \]

\[ 1 - p_2 \]

\[ 1.0 \]

\[
\text{Table 3. Probability table for two inspectors}
\]

Suppose that the data in Table 1 is observed from \( N \) Bernoulli trials. Then, the maximum likelihood estimator of \( p_{12}, p_1, p_2 \) are \( n_{12}/N \), \( n_1/N \), and \( n_2/N \), respectively. If two inspectors are statistically independent, then we have \( p_{12} = p_1 p_2 \) or

\[
\frac{n_{12}}{N} = \frac{n_1}{N} \cdot \frac{n_2}{N}.
\]

From (10), the Lincoln-Peterson estimator in (1) can be easily derived and the number of undetected errors \( n_0 \) is estimated as follows:

\[
\hat{n}_0 = \frac{(n_1 - n_{12})(n_2 - n_{12})}{n_{12}}.
\]

In many occasions, however, this independence assumption might be violated particularly when (a) inspectors collaborate (resulting in more defects found in common), or (b) specialize on certain defect types (resulting in fewer defects found in common). Collusion occurs when multiple inspectors work together, resulting in more common faults than would be expected. On the other hand, inspectors specializing in different areas may be intentionally chosen to ensure nothing is overlooked. Since each inspector may focus on certain parts of the document relating to his or her area of expertise, specialization may result in fewer common faults than would otherwise be expected. In either case, the cornerstone of capture-recapture models is no longer valid.

In the paper, we propose a novel approach that is distinct from the concept of capture-recapture models. The idea is to consider the correlation matrix of multiple inspectors. Since the correlation coefficients are expressed as a function of \( N \), we can estimate \( N \) if the correlation matrix of multiple inspectors is given a priori or can be estimated based on past inspection history.

**CORRELATION MATRIX OF MULTIPLE INSPECTORS**

The correlation coefficient is a standardized measure of the relationship between two variables \( X_1 \) and \( X_2 \), which is defined as follows:
As shown above, the correlation coefficient is computed by dividing the covariance by the standard deviations of the two variables, $X_1$ and $X_2$. This division results in the correlation coefficient having a standardized range from -1 to +1. The correlation coefficient provides us with a good indication of relationship because it varies in the range from -1 to +1 and its value directly indicates (1) the strength and (2) the direction of the relationship between two variables.

For the binary random variables $X_1$ and $X_2$ in Table 3, the covariance between $X_1$ and $X_2$ is shown to be

$$
\text{Cov}(X_1, X_2) = E[X_1X_2] - E[X_1]E[X_2] = p_{12} - p_1p_2,
$$

and thus the correlation coefficient between the two binary variables is

$$
\rho_{12} = \frac{p_{12} - p_1p_2}{\sqrt{p_1(1 - p_1)p_2(1 - p_2)}}.
$$

From the contingency table in Table 1, the observed correlation coefficient is

$$
r_{1,2}(N) = \frac{n_{12}N - n_1n_2}{\sqrt{n_1(N - n_1)n_2(N - n_2)}} = \frac{n_{12}N - n_1n_2}{\sqrt{n_1(N - n_1)n_2(N - n_2)}},
$$

which is also expressed as a function of $n_0$ as follows:

$$
r_{1,2}(n_0) = \frac{n_{12}n_0 - u_1u_2}{\sqrt{n_1(u_1 + n_0)n_2(u_2 + n_0)}}.
$$

Since $n_0$ and $r_{1,2}$ in (16) are in one-to-one correspondence, we can estimate the total number of undetected errors $n_0$ for any given value of the correlation coefficient $\hat{\rho}_{1,2}$. In other words, there exists a unique solution because $r_{1,2}(n_0)$ is a monotone function of $n_0$. If the inspections are statistically independent as in capture-recapture models, for example, the correlation coefficient in (16) is assumed to be zero and the number of undetected errors is estimated as

$$
\hat{n}_0 = \frac{u_1u_2}{n_{12}},
$$

which is consistent with (11). Thus, the Lincoln-Peterson estimator is a special case of (16) in which the correlation coefficient is zero.

Suppose that there are $m$ inspectors. From Table 2, the observed correlation coefficient between inspector $i$ and inspector $j$ is obtained as follows:

$$
r_{ij}(n_0) = \frac{n_{ij}(u_i + n_0) - (n_i - n_{ij})(n_j - n_{ij})}{\sqrt{n_i(u_i + n_0)n_j(u_j + n_0)}},
$$

which is a function of $n_0$. For the $m \times m$ correlation matrix, there are $m(m-1)/2$ number of elements $r_{ij}(n_0)$, all of which will be changed simultaneously as we change the value of $n_0$.

Suppose that all of $\rho_{ij}$ in the correlation matrix are known a priori or can be estimated from the past inspection history. Then, there are “deviations” between the given correlation coefficients $\hat{\rho}_{ij}$ and the observed correlation coefficients $r_{ij}(n_0)$. The given correlation coefficients $\rho_{ij}$ are considered multiple “goals” that should be achieved by changing $n_0$, and thus the problem of estimating the number of undetected errors $n_0$ can be formulated as a “goal programming” model in operations research. In the next section, we estimate the number of undetected errors via the goal programming approach.

**GOAL PROGRAMMING APPROACH**

For $m$ inspectors, there are $m(m-1)/2$ number of goals $\hat{\rho}_{ij}$ to be achieved and the same number of attained values $r_{ij}(n_0)$.
obtained from the inspection data. Note that $r_j(n_0)$ are all functions of a single variable $n_0$. As we change $n_0$, all $r_j(n_0)$ are changed simultaneously. Thus, each of the attained value $r_j(n_0)$ may be lower or higher than a given goal $\hat{p}_{ij}$. Let $d_{ij} = \hat{p}_{ij} - r_j(n_0)$ be a “deviational variable”, which is the difference between the goal and the attained value. Using the deviational variable, we can formulate a goal program that minimizes the total cost incurred in deviating from the goals.

As in many other goal programming models, we use the additive objective function for its simplicity: the total cost is the weighted sum of individual costs as follows:

$$\min_{\{n_0\}} z = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} v_{ij} c(d_{ij})$$

subject to

$$\hat{p}_{ij} - r_j(n_0) = d_{ij}, \quad i = 1, 2, \ldots, m-1 \text{ and } j = 2, 3, \ldots, m$$

$$-\infty < d_{ij} < +\infty; \quad n_0 \geq 0,$$

where $c(d_{ij})$ is a cost function and $v_{ij}$ is its weight.

The weight $v_{ij}$ should be determined such that the most important goal has the largest weight, and so on. In practice, we suggest that the weight $v_{ij}$ be determined based on $n_i$, $n_j$, and $n_{ij}$; e.g.,

$$v_{ij} = n_i + n_j - n_{ij}.$$

For the cost function $c(d_{ij})$, we could use the one that minimizes the weighted sum of squared deviations,

$$c(d_{ij}) = d_{ij}^2,$$

or that minimizes the weighted sum of absolute deviations

$$c(d_{ij}) = |d_{ij}|.$$  

In practice, the goals or targets, $\hat{p}_{ij}$, may not be known a priori. In that case, we recommend the following heuristic method.

The attained correlation coefficient $r_j(n_0)$ in (18) is a monotonically increasing function of $n_0$. Thus, the lower limit $r_{ij}^{\min}$ is obtained when $n_0 = 0$,

$$r_{ij}^{\min} = r_j(n_0 = 0) = \frac{n_j u_{ij} - (n_i - n_{ij})(n_j - n_{ij})}{\sqrt{n_i u_{ij} + n_j u_{ij}}}.$$

and the upper limit on $r_{ij}$ is achieved when $n_0 \to \infty$,

$$r_{ij}^{\max} = \lim_{n_0 \to +\infty} r_j(n_0) = \lim_{n_0 \to +\infty} \frac{n_j (u_{ij} + n_0) - (n_i - n_{ij})(n_j - n_{ij})}{\sqrt{n_i (u_{ij} + n_0) n_j (u_{ij} + n_0)}}$$

$$= \lim_{n_0 \to +\infty} \frac{n_j (u_{ij} / n_0 + 1) - (n_i - n_{ij})(n_j - n_{ij}) / n_0}{\sqrt{n_i (u_{ij} / n_0 + 1)(u_{ij} / n_0 + 1)}} = \frac{n_j}{\sqrt{n_i n_j}}.$$

Since the unknown goal $\hat{p}_{ij}$ is somewhere between $r_{ij}^{\min}$ and $r_{ij}^{\max}$, we may set the goal or target $\hat{p}_{ij}$ as a weighted average of the lower bound $r_{ij}^{\min}$ and the upper bound $r_{ij}^{\max}$ as follows:

$$\hat{p}_{ij} = w r_{ij}^{\min} + (1-w) r_{ij}^{\max}.$$

As a practical matter, we suggest that equal weight should be put on the lower and the upper bound because we lack any theory that suggests otherwise. In our empirical research, we also found that the weight $w$ of 0.5 or 0.6 gives a good estimate.
of $\hat{\rho}_j$.

**PERFORMANCE EVALUATIONS**

In our goal programming approach, the goal is a weighted average of the lower and the upper bounds of the observed correlation coefficients $r_{ij}(n_0)$. Thus, the weight $w$ should be specified in our model. In addition, the form of the cost function $c(d_{ij})$ and its appropriate weight $v_{ij}$ in the objective function should be also specified in advance. In the numerical analysis, we set $w$ equal to 0.6, which is the most appropriate value that we found from our empirical research. The objective is to minimize the weighted average of the absolute deviations, in which the weight $v_{ij}$ is determined from (21).

Suppose that there are exactly 100 ($=N$) errors in the system, among which 80 ($=n_f$) errors have been detected by $m$ inspectors. For all the possible cases, we found the estimates of $N$ using the SOLVER function in Microsoft Excel and then computed the mean square error,

$$\text{MSE} = \frac{\sum_{j=1}^{k} (\hat{N}_j - 100)^2}{k}$$

and the mean absolute percentage error,

$$\text{MAPE} = \frac{\sum_{j=1}^{k} |\hat{N}_j - 100|}{100k}$$

where $k$ is the number of possible cases considered in the numerical analysis.

First, we consider the situation in which there are two inspectors. As shown in Table 4, we consider 24 possible cases in which exactly 80 errors have been found by the two inspectors and 20 errors are still remaining in the system. In the first row in Table 9, for example, both Inspector 1 and Inspector 2 have detected 70, respectively. If their inspections are statistically independent, the number of common errors detected by both inspectors should be 49. Since they have detected 60 common errors, it implies that their inspections are positively interrelated with each other.

<table>
<thead>
<tr>
<th>$n_1$</th>
<th>$n_2$</th>
<th>$n_{12}$</th>
<th>MLE</th>
<th>Darroch (1958)</th>
<th>Chao (1989)</th>
<th>Chun</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>70</td>
<td>70</td>
<td>60</td>
<td>80.968</td>
<td>81.667</td>
<td>81.639</td>
</tr>
<tr>
<td>2</td>
<td>65</td>
<td>70</td>
<td>55</td>
<td>81.980</td>
<td>82.727</td>
<td>82.679</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>70</td>
<td>50</td>
<td>83.186</td>
<td>84.000</td>
<td>83.922</td>
</tr>
<tr>
<td>4</td>
<td>55</td>
<td>70</td>
<td>45</td>
<td>84.657</td>
<td>85.556</td>
<td>85.435</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>70</td>
<td>40</td>
<td>86.493</td>
<td>87.500</td>
<td>87.317</td>
</tr>
<tr>
<td>6</td>
<td>45</td>
<td>70</td>
<td>35</td>
<td>88.852</td>
<td>90.000</td>
<td>89.722</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
<td>70</td>
<td>30</td>
<td>91.997</td>
<td>93.333</td>
<td>92.903</td>
</tr>
<tr>
<td>8</td>
<td>35</td>
<td>70</td>
<td>25</td>
<td>96.399</td>
<td>98.000</td>
<td>97.308</td>
</tr>
<tr>
<td>9</td>
<td>30</td>
<td>70</td>
<td>20</td>
<td>103.001</td>
<td>105.000</td>
<td>103.810</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>70</td>
<td>15</td>
<td>114.003</td>
<td>116.667</td>
<td>114.375</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
<td>70</td>
<td>10</td>
<td>136.006</td>
<td>140.000</td>
<td>134.545</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>70</td>
<td>5</td>
<td>202.011</td>
<td>210.000</td>
<td>188.333</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>60</td>
<td>40</td>
<td>88.996</td>
<td>90.000</td>
<td>89.756</td>
</tr>
<tr>
<td>14</td>
<td>55</td>
<td>60</td>
<td>35</td>
<td>93.141</td>
<td>94.286</td>
<td>93.889</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>60</td>
<td>30</td>
<td>98.667</td>
<td>100.000</td>
<td>99.355</td>
</tr>
<tr>
<td>16</td>
<td>45</td>
<td>60</td>
<td>25</td>
<td>106.401</td>
<td>108.000</td>
<td>106.923</td>
</tr>
<tr>
<td>17</td>
<td>40</td>
<td>60</td>
<td>20</td>
<td>118.003</td>
<td>120.000</td>
<td>118.095</td>
</tr>
<tr>
<td>18</td>
<td>35</td>
<td>60</td>
<td>15</td>
<td>137.338</td>
<td>140.000</td>
<td>136.250</td>
</tr>
<tr>
<td>19</td>
<td>30</td>
<td>60</td>
<td>10</td>
<td>176.006</td>
<td>180.000</td>
<td>170.909</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
<td>60</td>
<td>5</td>
<td>292.008</td>
<td>300.000</td>
<td>263.333</td>
</tr>
<tr>
<td>21</td>
<td>20</td>
<td>50</td>
<td>20</td>
<td>123.003</td>
<td>125.000</td>
<td>122.857</td>
</tr>
<tr>
<td>22</td>
<td>15</td>
<td>50</td>
<td>15</td>
<td>147.338</td>
<td>150.000</td>
<td>145.625</td>
</tr>
<tr>
<td>23</td>
<td>10</td>
<td>50</td>
<td>10</td>
<td>196.006</td>
<td>200.000</td>
<td>189.091</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>50</td>
<td>5</td>
<td>342.007</td>
<td>350.000</td>
<td>305.000</td>
</tr>
<tr>
<td>Mean squared error</td>
<td>5359.346</td>
<td>5817.843</td>
<td>4031.396</td>
<td>40.419</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean absolute percentage error</td>
<td>42.408%</td>
<td>44.067%</td>
<td>38.134%</td>
<td>5.240%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Numerical analysis for two inspectors: $N = 100$
In this case, the total number of errors is estimated to be 80.968, 81.667, and 81.639, respectively, by the maximum likelihood estimator, Darroch’s approximate maximum likelihood estimator, and Chao’s estimator. Note that Darroch’s estimate and Chao’s estimate are the same as the Lincoln-Peterson estimate and Chapman’s estimate, respectively, when \( m = 2 \). Our estimate of \( N \) based on the goal programming approach is 86.667, which is slightly closer to the parameter \( N = 100 \). In terms of the MSE and MAPE criteria, our method performs much better than the traditional capture-recapture methods as shown in the last two rows in Table 9. Particularly when the inspectors are negatively interrelated as in case 24, our goal programming approach outperforms the traditional methods remarkably.

The performance of our method is reaffirmed in the three inspector cases. As shown in Table 5, we considered 42 possible cases of \( m = 3 \), systematically changing the number of errors detected by each inspectors to simulate various situations. As shown in last two rows in Table 10, the MSE and the MAPE of our method are much smaller than those of the traditional capture-recapture models.

We did not perform the numerical analysis for the cases of more than three inspectors, because there are too many possible permutations. For \( m \) inspectors, for example, we need to simulate \( 2^m - 1 \) different inspection history. However, we expect that our method would also outperform the traditional models that are based on the independence assumption.

<table>
<thead>
<tr>
<th>( n_1 )</th>
<th>( n_2 )</th>
<th>( n_3 )</th>
<th>( n_{12} )</th>
<th>( n_{13} )</th>
<th>( n_{23} )</th>
<th>( n_{123} )</th>
<th>MLE</th>
<th>Darroch (1958)</th>
<th>Chao (1989)</th>
<th>Chun</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>50</td>
<td>81.081</td>
<td>80.583</td>
<td>84.688</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>65</td>
<td>40</td>
<td>40</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>50</td>
<td>81.824</td>
<td>81.250</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>65</td>
<td>40</td>
<td>55</td>
<td>55</td>
<td>40</td>
<td>50</td>
<td>82.673</td>
<td>82.026</td>
<td>81.630</td>
</tr>
<tr>
<td>4</td>
<td>65</td>
<td>50</td>
<td>40</td>
<td>55</td>
<td>55</td>
<td>40</td>
<td>50</td>
<td>82.673</td>
<td>82.026</td>
<td>81.630</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>50</td>
<td>40</td>
<td>55</td>
<td>40</td>
<td>40</td>
<td>50</td>
<td>83.475</td>
<td>82.791</td>
<td>81.630</td>
</tr>
<tr>
<td>6</td>
<td>35</td>
<td>50</td>
<td>40</td>
<td>55</td>
<td>40</td>
<td>40</td>
<td>50</td>
<td>83.475</td>
<td>82.791</td>
<td>81.630</td>
</tr>
<tr>
<td>7</td>
<td>65</td>
<td>35</td>
<td>40</td>
<td>55</td>
<td>40</td>
<td>40</td>
<td>50</td>
<td>83.475</td>
<td>82.791</td>
<td>81.630</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>50</td>
<td>40</td>
<td>55</td>
<td>40</td>
<td>40</td>
<td>50</td>
<td>83.475</td>
<td>82.791</td>
<td>81.630</td>
</tr>
<tr>
<td>9</td>
<td>65</td>
<td>65</td>
<td>50</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>50</td>
<td>84.688</td>
<td>84.688</td>
<td>84.688</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>65</td>
<td>50</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>50</td>
<td>84.688</td>
<td>84.688</td>
<td>84.688</td>
</tr>
<tr>
<td>11</td>
<td>35</td>
<td>65</td>
<td>65</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>85.577</td>
<td>84.756</td>
<td>81.230</td>
</tr>
<tr>
<td>12</td>
<td>50</td>
<td>65</td>
<td>65</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>85.577</td>
<td>84.756</td>
<td>81.230</td>
</tr>
<tr>
<td>13</td>
<td>65</td>
<td>35</td>
<td>65</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>85.577</td>
<td>84.756</td>
<td>81.230</td>
</tr>
<tr>
<td>14</td>
<td>50</td>
<td>65</td>
<td>65</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>85.577</td>
<td>84.756</td>
<td>81.230</td>
</tr>
<tr>
<td>15</td>
<td>35</td>
<td>65</td>
<td>65</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>85.577</td>
<td>84.756</td>
<td>81.230</td>
</tr>
<tr>
<td>16</td>
<td>50</td>
<td>65</td>
<td>65</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>85.577</td>
<td>84.756</td>
<td>81.230</td>
</tr>
<tr>
<td>17</td>
<td>65</td>
<td>35</td>
<td>65</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>85.577</td>
<td>84.756</td>
<td>81.230</td>
</tr>
<tr>
<td>18</td>
<td>50</td>
<td>65</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>86.484</td>
<td>85.948</td>
<td>82.758</td>
</tr>
<tr>
<td>19</td>
<td>35</td>
<td>65</td>
<td>65</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>86.484</td>
<td>85.948</td>
<td>82.758</td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>50</td>
<td>65</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>87.385</td>
<td>86.849</td>
<td>83.258</td>
</tr>
<tr>
<td>21</td>
<td>65</td>
<td>50</td>
<td>65</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>87.385</td>
<td>86.849</td>
<td>83.258</td>
</tr>
</tbody>
</table>

Mean squared error 228.996
Mean absolute percentage error 13.049%
CONCLUDING REMARKS

In the paper, we proposed a new method that estimates the number of undetected errors in software design documents. The idea is to use the correlation matrix of multiple inspectors and to formulate the estimation problem as a goal program. To use our method in practice, the goal or target $\rho_{ij}$ must be given in advance or should be estimated based on the past data or experience. If the goal $\rho_{ij}$ is not available, we may first find, using the observed data, its lower and upper bounds, and then estimate the goal as a weighted average of the two bounds. The optimal weight could be obtained from the training data, or we suggest $w=0.5$ or 0.6 if there is no prior information at all.

We also illustrated our method with an empirical data and compared its performance with those of traditional capture-recapture models that are based on the independence assumption. Although our method outperformed other capture-recapture models, it would be more informative to use our method along with other traditional models in a specific inspection environment.

The capture-recapture model was first introduced by ecologists as a means of estimating the size of wildlife populations. However, the basic methodology has been used in different scientific areas such as demography, engineering, and epidemiology. We expect that our method can be successfully applied to any areas in which we suspect the independence assumption among inspectors.

REFERENCES


