2nd International Software Quality Week Europe

Technology Track

Conference Day #3 (Thursday, 12 November, 1998)

6T	**Dr. Linda Rosenberg, Mr. Ted Hammer & L. Hoffman,** *GSFC NASA / Unisys* Testing Metrics for Requirement Quality
7T	**Mr. Hans Buwalda,** *CMG Finance BV* Testing with Action Worlds, A Quality approach to (Automated) Software Testing
8T	**Mr. Jon Huber,** *Hewlett Packard* Software Defect Analysis: Real World Testing Implications & A Simple Model for Test Process Defect Analysis
9T	**Prof. Antonia Bertolino & Ms. E. Marchetti,** *CNR-IEI* A Simple Model to Predict How Many More Failures Will Appear in Testing
10T	**Dr. Stacy J. Prowell,** *Q-Labs, Inc.* Impact of Sequence-Based Specification on Statistical Software Testing

A simple model to predict how many more failures will appear in testing

A. Bertolino, E. Marchetti

- Introduction
- Bemar model
- Case studies
- Future work

Static models of software defects

Software metrics are used to estimate the number of defects in the software

General form

$$y = f(x_1, x_2, \cdots, x_n)$$

y is a defect metric, such as:

- number of changes required in the design
- number of errors
- number of program changes

$x_1, \ldots, x_n$ can be:

- product-related
- process-related

Akiyama's study

Metrics used (product-related)

- program size in lines of code (S_s)
- count of decisions (DE)

Total number of defects

$$d_{tot} = 4.86 + 0.018 S_s$$

$$d_{tot} = -1.14 + 0.2 DE$$

Dynamic models of software defects

- The software system is considered as a "black box"
- Tests are developed using the operational profile
- The reliability is estimated without considering the complexity of the program
- Test are developed for:
  - increasing the reliability
  - estimating the reliability

Well known examples are:

- Musa model for the time between failures data
- Yamada S-shaped model for the failures per time period data
- Goel/Okumoto model for both

Bemar model: motivation

- Sometime identifying an operational profile is quite difficult and expensive
- Operational testing is applied to the whole system, or to big-size portions of it
- Operational testing can only start when the software configuration and behavior are fairly stable
- In general, commonly used debug test methods do not exhibit a regular trend in reliability
- Oftentimes for a software producer modifying the test process is not easy

Bemar model: purpose

- To predict the number of remaining failures when:
  - the operational profile can't be identified
  - the test involved single modules or small pieces
  - the test process is in the early phases, which are functional and deterministic
  - the rate of detection of failures remains quite stable
- To provide the software producer with a method that:
  - can be applied without any change to the test process
  - establishes the effectiveness of the tests performed so far
  - establishes a stop criterion for the testing

Bemar model: application

- Collect the data during the test
- Establish a test interval (TI) length
- Group failure data into test intervals
- Apply the Bemar method

Cai's method

Assumptions

- Modules are randomly divided in part 0 and part 1
- There are N=N0+N1 remaining defects in the software:
  - N0 in part 0
  - N1 in part 1

Use

- Perform code review on a randomly chosen module
- Establish to which part (0 or 1) the module that shows a defect belongs
- Use the number of defects discovered to predict how many defects should be detected during the phase of dynamic testing in each part (N0, N1)

Model rationale

Intuition

- The number of failures f can be estimated by:

$$f = n * t$$

  - n the total number of tests executed
  - t is the failure detection rate

Problems

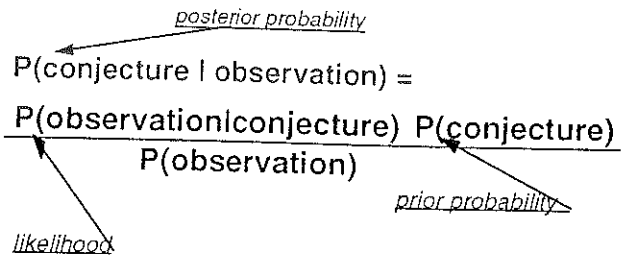
- A distribution should be used instead of a known failure detection rate
- The empirical distribution for t can only be identified after tests are completed

Solution

- A Bayesian approach to derive the distribution of t from the observation of test results.

Bayesian approach

- "subjective" interpretation of probability
- allows consistent deductions from probability statements, and inference from observation
- given *prior* probabilities and new observation, derives updated *posterior* probability:



To predict the failures

A random variable **T**, that takes values in [1..M], is defined as the distance between two successive failed test intervals

First step

Predict the number of failed test intervals N_{FTI}

Second step

Predict the number of expected failures N_F

First step

- Establish a prior distribution for T
- Derive a posterior distribution for T (after a given number k of test intervals):

$$P(T=i \mid F_k) = \frac{p_T(i)\cdot\left(\frac{1}{i}\right)^f\left(1-\frac{1}{i}\right)^{k-f}}{\sum_{j=1}^{M} p_T(j)\cdot\left(\frac{1}{j}\right)^f\left(1-\frac{1}{j}\right)^{k-f}}$$

- Predict the number of failed test intervals

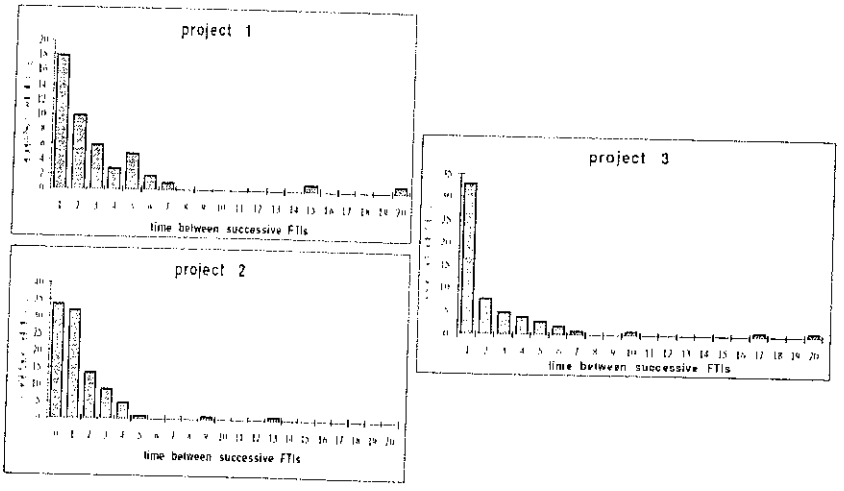
$$N_{FTI,k} = \frac{NTI}{E_k[T]}$$

Second step

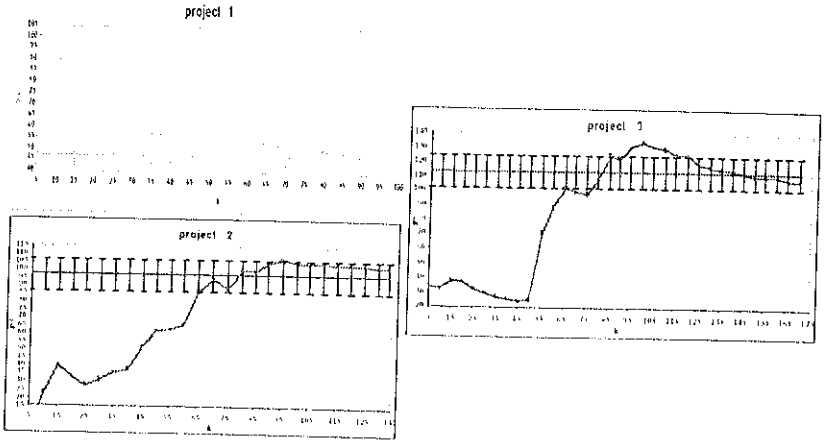
- Derive the mean number of failures observed within a failed test interval: $E_k[F]$
- Predict the number of failures that the product will show at the end of the test

$$N_{F,k} = N_{FTI,k} \cdot E_k[F]$$

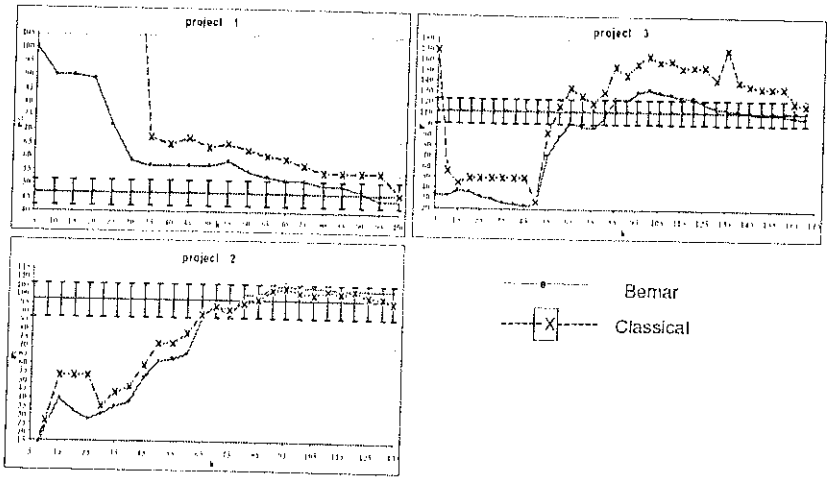
Sets of failure data



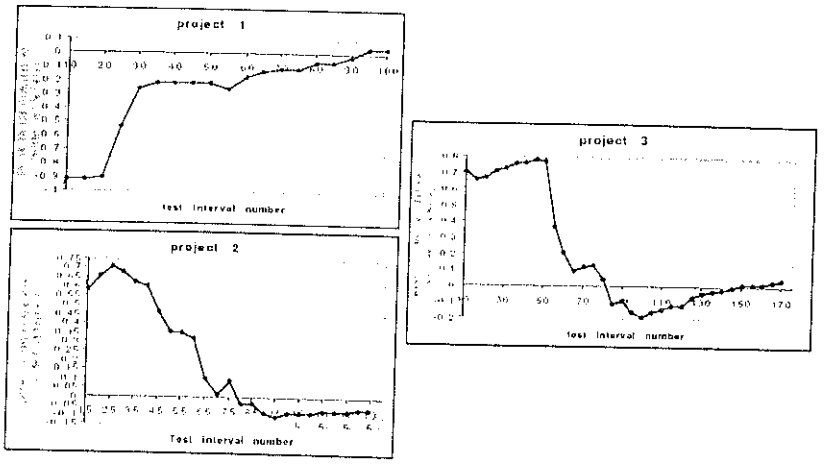
Model predictions



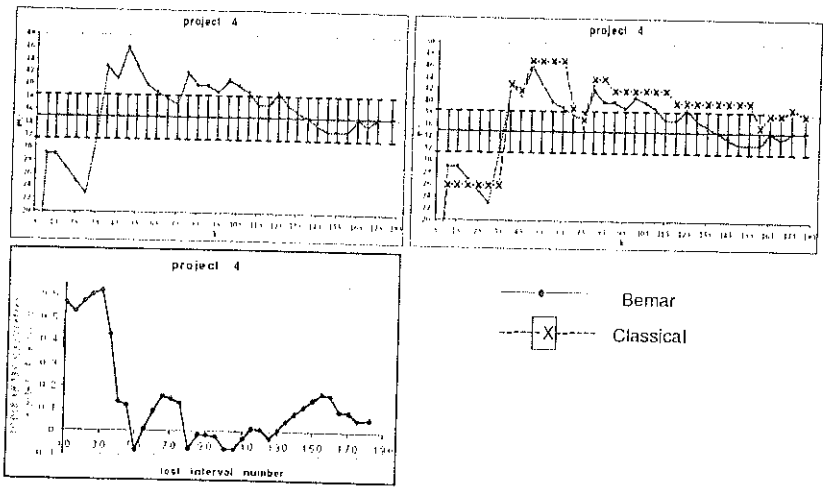
Comparison with a "classical" approach



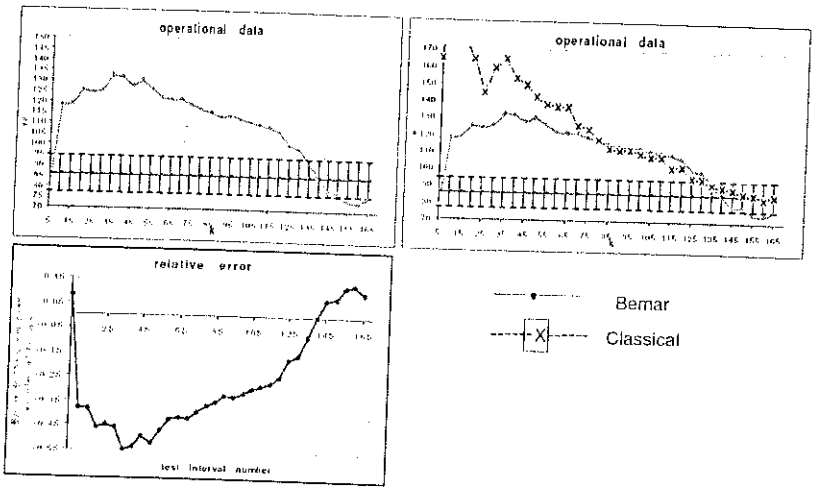
Relative error



Application to a set of operational data



Application to a published set of data



Future work

- Find more efficacious ways to express prior knowledge
- Quantify in advance the goodness of Bemar prediction
- Validate the Bemar method with more data
- Use of Bemar in combination with a reliability growth model

A simple model to predict how many more failures will appear in testing

A. Bertolino, E. Marchetti

Istituto di Elaborazione della Informazione, CNR, Pisa, Italy

bertolino@iei.pi.cnr.it, e.marchetti@iei.pi.cnr.it

Abstract

This paper deals with dynamic models to evaluate how many more failures will be observed in future tests, based on the failures observed so far. The assessment of reliability through testing is now one of the most mature fields in software engineering. There exist tens of reliability growth models, and several tools for applying them. The major assumptions of these models are that test cases are randomly drawn from the operational profile, and that as defects are found and removed, reliability will exhibit an increasing trend. Both assumptions are hardly satisfied in the first stages of the testing process or for the testing of small modules. Besides, there are not reasons why commonly used test methods at this time, such as specification-based testing or branch coverage, should exhibit a regular trend in reliability.

These are the motivations for the work reported here. A dynamic model is introduced that can be applied to predict the number of remaining failures in early test phases. It is called the Bemar model. The Bemar model is quite general and makes no assumption on as to how tests are selected. The most attractive feature is indeed the simplicity of the model: testers have just to collect the detection rates of failures, i.e., the intervals between subsequent failures. No estimation of parameters of the product or of the development process is required.

Keywords: Bayesian approach, defect count models, functional testing, number of expected failures

1. Introduction

In spite of great advances in the software engineering field since the complaints about a software crisis began to spread in the mid-seventies, the state of practice in software development is still such that producing defect-free code remains wishful thinking. On the contrary, coping with software failures, during development and after release, is among the hardest tasks of managers, while testing, debugging and maintenance activities still consume the largest part of development effort and resources. For these reasons, methods to estimate the defect contents of software are of great interest for managers and testers.

Researchers have devoted much attention to this problem and have proposed many models to quantify faults and failures. It is important to distinguish between two different approaches that have been taken. One approach consists of looking at properties of the present or past products, and/or at parameters of the development process and then, using these observations, trying to make a guess of the total number of defects, or faults, in the current product. A different approach

is instead to observe defects, or, more properly, failures, as they show up in testing. Based on the observed behavior, one then uses statistical inference procedures to predict the number or the time of failures expected in future tests or in operation.

Depending on which of the two approaches is followed, defect counting models have been categorized as static or dynamic, respectively [Conte et al., 1986, Chapter 7]. However, the fact that static and dynamic models assess two different entities, namely defects in the code the first and failures to be observed the second, must be underscored.

Static models are very attractive to managers, because they provide "numbers", which the managers are eager of, very early in advance in comparison with dynamic models. The latter can only be used late in the life cycle, i.e., in the testing phases, when it may be too late to efficaciously re-direct development efforts. In fact, static defect models are used to identify more risky modules and consequently re-allocate testing resources or modify design. In addition, static models claim to estimate the total number of defects. As by testing we find and fix failures, then, static models would provide a prediction on how many defects are left in the code, which may seem a very attractive measure at first glance.

On the other hand, a defect can be more or less disturbing depending on whether, and how much frequently, it will eventually show up to the final user (and depending of course on the seriousness of its consequences). Indeed, in many or in few, some defects will inevitably escape testing and debugging. So, in the end, the real important measure to decide whether a product can be released is software reliability; i.e., the number of *failures*, and not of remaining defects, must be estimated. Until they do not cause failures, remaining defects do not trouble neither customers nor producers.

The right position is that static and dynamic models are both useful, but for different objectives. In the front-end phases of the life cycle, managers should use static models to apportion risk among modules and to allocate development time and resources. In the final stages of development, instead, they should use dynamic models in order to evaluate how much disturbing are the defects that are inevitably left, and to decide whether the product is ready for delivery.

This paper deals with dynamic models to evaluate how many more failures are expected to be observed in future tests, based on the failures observed so far. The assessment of reliability through testing is now one of the most mature fields in software engineering [Lyu, 1996]. There now exist tens of reliability growth models, and several tools for applying them, in combination with rather sophisticated techniques to evaluate the accuracy of the measures given by the models, and to select the most appropriate model for a specific data set.

Existing models, though, all share the underlying assumption that the test cases are randomly drawn from the operational profile, and that as defects are found and removed, reliability will exhibit an increasing trend. Both assumptions are hardly satisfied in the first stages of the testing process. Industrial test processes commonly undergo several subsequent steps, identified with differing terms, from unit to subsystem, and to system testing. Operational testing can only start when the software configuration and behavior are fairly stable, and is applied to the whole system, or to big-size portions of it. For the testing of single modules, or of small subsystems,

identifying an operational profile is quite difficult and expensive, and perhaps not sensible at all. Besides, there are not reasons why commonly used test methods at this time, e.g., branch coverage, should consistently exhibit a regular growth in reliability.

These are the underlying motivations for the work reported here. We introduce a dynamic model, called the Bemar model, that can be applied to predict the expected number of remaining failures in early test phases. The Bemar model is quite general and makes no assumption on as to how test are selected. The most attractive feature is indeed the simplicity of the model. It only requires to collect the intervals of time between subsequent failures. No estimation of parameters of the product or of the development process is needed.

In the next section the underlying intuitive model is described; the mathematical formulation is provided in Section 3. The model has been applied to some real world data; the results are presented in Section 4. Although the data available are too poor to validate the model, these first results look promising. This work is still in a preliminary phase; we briefly outline future directions in the Conclusions.

2. Model Rationale

In measurement, one tries to map observations of the empirical world to mathematical entities that can be formally manipulated. Models are defined trying to capture one's intuition and understanding of the real world; indeed, "intuition is the starting point for all measurement" [Fenton and Pfleeger,1997]. In this section we present the intuition underlying the Bemar model.

The stimulus for this work came from the analysis of the test results collected over several projects by a software producer, namely Ericsson Telecomunicazioni S.p.A. in Rome. This producer routinely logs for each product the failures observed since early test phases until beta testing, and is interested in finding more effective ways to use these data for project management and product control. So far, these data are used to derive measures of fault density, that is the ratio between the cumulative number of failures observed in a given time period and the product size, expressed in lines of code.

With regard to the results from beta testing, which is operational, standard approaches for reliability estimates and predictions can be applied. In [Bertolino et al., 1998], we describe a first case study conducted at the same producer, aimed at experiencing the use of software reliability engineering techniques. But, reliability growth models could not be applied to the early test phases, for the reasons we explained in the introduction.

It must be made clear beforehand that it is not the case that this producer is looking for new testing methods to be applied that would facilitate failure predictions (as could be for instance the case if fault seeding approaches were applied). On the contrary, this producer has a well established and formalized test process, and is looking for efficient metrics that can be applied to the data collected. It is plausible to assume that to a certain extent this proviso would be the same for many other producers.

We surveyed the literature in search for a dynamic model that could be applied to the test outputs from the early test phases; reliability growth models could not be used, as earlier explained. An

interesting finding of this survey was [Cai, 98]. Cai has proposed a model to predict the remaining number of defects in the code based on the failures that are observed in testing, which is in a sense a hybrid approach between static and dynamic models. Since the assumptions underlying Cai's model reasonably held for the projects of this producer, the model was applied to the data available, in order to see if the estimation of defects provided by the model was conclusive for our situation, but with negative results.

We investigated on the reasons why Cai's method, which reportedly worked well on his data, did not function on our data. One of the findings was that Cai's model does not consider the time occurrence of failures. Intuitively, Cai's model is similar to fault seeding methods, but instead of considering the proportion between seeded faults and unknown faults, Cai divides the software under test into two parts, and uses the relative occurrence of (real) faults in either parts. The model is thus only concerned with the number of faults and possibly with how these are distributed among the modules of a system, but not with the time of their detection.

In our opinion, the rate of failure discovery is a fundamental parameter, and should be included in the model. In simple words, the scenario we have in mind is that n failures are detected after d days of testing, and that we want to estimate how many more failures we expect to find in the next d' days, if we continue to test in the same way. We reasonably think that the prediction should be different if the failures are uniformly distributed over the d days, or if instead all the failures are, say, discovered in the first day of testing, and then the remaining (d - 1) days exhibit no failures.

We have consequently defined a new dynamic model taking into account the time distribution of failure discoveries. The intuition behind this new model is very simple: assuming that we can know a priori, or somehow estimate, the rate of failure findings over the sequence of executed tests, say t, then if by n we denote the total number of tests to be executed, quite obviously the expected number of failures f would be estimated by:

(1) $f = n * t$

Of course this formula is rather naive and cannot be used in practice in this simplistic form, because the rate of failure detection in testing can never be established with certainty; it is rather a random variable, for which a distribution should be identified. For each new product under test, the empirical distribution of the failure detection rate can only be precisely drawn only after the testing is completed. However, if we could assume that, after having observed the test results for some time it stabilizes (i.e., it can be used as an approximation of the real, so far unknown distribution, to predict future behavior), then a formula generalizing Eq. (1) could be used. This is the underlying intuition of the Bemar model. To derive the distribution of the failure detection rate from the observation of test results, Bemar uses a Bayesian approach. This is described in the next section.

According to its justification, we expect that Bemar performs better when the rate of detection of failures in testing remains more or less stable. This is in contrast with the assumption underlying reliability growth models. In fact, the Bemar model should be applied to early test phases, and in general to all those situations in which failures are found with some regularity, and remains valid

only for limited periods, i.e., till the point in which the rate of occurrence of failures starts to decrease as an effect of having removed a high number of faults.

In other words, the Bemar model performs well as long as reliability growth models cannot yet be applied. It is foreseeable that the Bemar model and a reliability growth model can be used in complementary way. How these could be combined will be object of future investigation.

3. Description of the Bemar Model

Before presenting the definitions and formulas adopted in the model, the typology of data available is described.

The software producer provided us with sets of failure data collected over several projects during the phase of subsystem testing. The test cases are deterministically chosen by examining the functional specifications and altogether before test execution starts (which means that the number of tests to be executed is decided in advance). The tests are not executed continuously, but only during the working days (i.e., five days in a week) and 8 hours per day. For each project, the information registered consists of the start and end dates of the test phase, and of the calendar day (but not the day time) of discovery of each failure. Test execution (CPU) times were not recorded.

Based on the coarse granularity of available data, we decided to group failure data into test intervals (TIs). A TI could be as long as a day, a week, or any other length (for instance measured in seconds), depending on the global duration of the testing, the precision of the data available and the amount of observed failures.

A TI in which at least a failure is observed is called a *failed test interval* (FTI), otherwise it is said a successful TI. Note that, anyhow small a test interval is chosen, until this remains larger than a single test there will always be a chance to observe more than one failure within a failed test interval. Hence, we predict the expected number of failures in two steps: first we predict N_{FTI}, i.e., the number of FTIs is estimated. Then, from this number, we derive the number of failures N_F.

In the first step, to estimate N_{FTI}, we define the distance between two subsequent FTIs as a random variable T, that can take discrete values within an interval [1, M] (where M is a maximum fixed value). Precisely, for each i within [1,M], the associated probability mass function (pmf), $p_T(i) = P(T=i)$, gives the probability that the next failure will be observed after i TIs (i.e., i -1 successful TIs are observed and then the ith TI is a FTI).

Denoting by NTI the total number of test intervals to be performed, and with $E[T] = \sum_{i=1}^{M} p_T(i) \cdot i$, it can be shown that the following formula holds[1]:

$$(2)\quad NTI = N_{FTI} \cdot E[T]$$

[1] Actually, this formula holds precisely if it can be assumed that the last test interval is a failed one. Otherwise, the left-hand side should be decreased by the number of test intervals occurring between the last FTI and the last test interval. This adjustment will be neglected in the paper.

Since for each project the number of test intervals can be easily derived (remember that the functional test cases are preselected in advance), Equation (2) above can be solved for N_{FTI}, yielding:

$$N_{FTI} = \frac{NTI}{E[T]} \quad (3)$$

We need now a procedure to derive E[T]. Looking at the data available, we see that the failures are variously distributed over the whole test period and it is not generally the case that towards the end of the functional test period less failures are observed than at the beginning (as it is expected in operational testing). In particular, the data do not show any consistent reliability increasing trend, appearance which was confirmed by the Laplace test [Kanoun et al., 1997] conducted over all the sets of data. In Figure 1, we show for instance the failure data relative to one of the products analyzed.

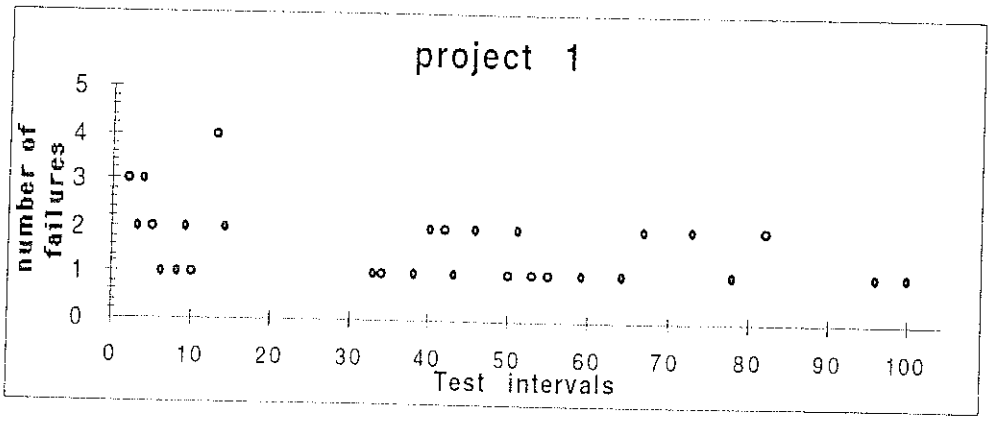


Figure 1: Failure data for a product

To develop a prediction procedure which is sensible, and correlated with the behavior of a given product under test, it is desirable to use the test results collected as functional test proceeds to adjust an initial estimate of the pmf. Hence, we chose to adopt a Bayesian approach.

In the Bayesian framework, probabilities are meant to describe an observer subjective knowledge of yet-unknown events. This knowledge evolves as events are observed. In this context, the pmf of T $p_T(i)$ is taken as the *prior* knowledge about the behavior of a product under test. I.e., $p_T(i)$ is taken to model a tester's subjective belief about the rate of failure detection *before* some evidence (the test results) about the product under test is observed. During the performance of the functional testing, the realization of a sequence of test intervals with and without failures is observed. Thanks to this evidence, the tester's knowledge about this product evolves and a new distribution for the pmf of T, called the *posterior* distribution, can be derived. Denoting by F_k the sequence of observed outcomes (failed/successful) for the first k TIs, the posterior distribution $p'_{T,k}(i)$ then gives $P(T=i \mid F_k)$, i.e., it is the update of $p_T(i)$ after having observed the sequence F_k. Applying Bayes' formula we have:

$$(4)\ p'_{T,k}(i) = (P(T=i \mid F_k) = \frac{P_{prior}(T=i)P(F_k \mid T=i)}{\sum_{j=1}^{M} P(F_k \mid T=j)P_{prior}(T=j)}$$

in which the term $P(F_k|T=i)$ is usually called a *likelihood function*. To derive it, we can consider that, if $T=i$, then the probability of observing a failure in the next test interval is $1/i$, i.e.:

$$(5)\ P(F_1|T=i) = \begin{cases} \frac{1}{i} & \text{if } F_1 \text{ is failed} \\ (1-\frac{1}{i}) & \text{if } F_1 \text{ is successful} \end{cases}$$

Substituting this in formula 4, and iterating the same reasoning also to the subsequent test intervals, we finally obtain[2]:

$$(6)\ p'_{T,k}(i) = \frac{p_T(i)\cdot\left(\frac{1}{i}\right)^f\left(1-\frac{1}{i}\right)^{k-f}}{\sum_{j=1}^{M} P_T(j)\cdot\left(\frac{1}{j}\right)^f\left(1-\frac{1}{j}\right)^{k-f}}$$

which gives the posterior pmf for the random variable T, after observing k test intervals, out of which f were failed.

In general, deriving a prior distribution for the probability of interest is a difficult task, which also generates some perplexity towards the usefulness of Bayesian inference methods. In our case, the form of $p_T(i)$ can be derived on the basis of data available from similar products. In general, some suitable representation of ignorance is often adopted, like for instance the uniform distribution, though actually absolute ignorance can never be assumed.

By using the posterior pmf provided by formula (6) to derive E[T], by (3) we can then derive $N_{FTI,k}$, i.e., the number of FTIs expected after NTI test intervals, using the test information collected during the first k test intervals.

From $N_{FTI,k}$ the total number of failures N_F needs now to be estimated. This clearly depends on how many failures on average are observed within a FTI. We can again define a random variable F to represents the number of failures observed within a FTI, and then derive N_F from N_{FTI}, with $N_F = N_{FTI} \cdot E[F]$.

We derive an empirical pmf for F by considering the results from the first k TIs. In particular, by analyzing the sets of failure data available, a maximum number of failures per FTI, MF, can be fixed. From the distribution of the number of failures within a failed test interval, we are able to calculate the expectation $E_k[F] = \sum_{i=1}^{MF} P_k(F=i)\cdot i$.

---

[2]In the generalization of this formula from the case k=1 to larger values of k, we have in reality used some relaxed assumptions, which could raise some objection to its validity from a purely theoretical viewpoint. In future work we will fix these problems. However, on the set of data available, this formula performed better than other theoretically stronger models.

Therefore, after having observed k TIs, the number of failures that a product will show at the end of the functional test is:

(7) $N_{F,k} = N_{FTI,k} \cdot E_k[F]$

The formulas (3) and (7) are to be used incrementally during functional test, i.e., considering each time a greater value for k, and adjusting the pmfs involved correspondingly. In this way, the prediction about the total number of failures for a product as testing proceeds will be more and more precise.

4. Application

The Bemar method has been experimented on the failure data relative to the functional test phase of several products; we have also tried it on some operational test results (for which we expect the model is not working as well as for functional testing). We briefly present the results in sections 4.1 and 4.2, respectively.

4.1 Functional testing

Before applying the Bemar model to the data relative to functional testing, we investigated ways to derive a suitable prior distribution for T.

About these data we knew that the products performed similar functionalities, they had been tested by the same producer and with the same methodology. It was plausible to expect that the test results could exhibit a similar behavior, which would be a useful fact to derive a prior pmf for T.

More in general, it is probable that a software producer has collected similar information about the functional tests developed in the past. In the case that the products exhibit a similar behavior, the information collected (in particular the mean and the variance) can be useful to establish a proper prior pmf of T for the next product that the producer will test.

First of all, analyzing the failure data we noticed that the distance between subsequent FTIs was not greater than 20 and that the maximum number of failures per FTI was 6. Therefore we considered that the variable T could take discrete values within the interval [1,20] and we put MF=6.

Then for each project we derived a histogram of the time distance (measured in elapsed test intervals) occurring between two subsequent FTIs. In Figure 2 we report the histogram corresponding to the product shown in figure 1. Analyzing the histograms for this and all the other sets of data available, a certain regularity in the failure behavior under functional testing was in fact noted. This observation would sustain the hypothesis that a general distribution for the distance between two successive FTIs for the functional test process of this producer can be identified.

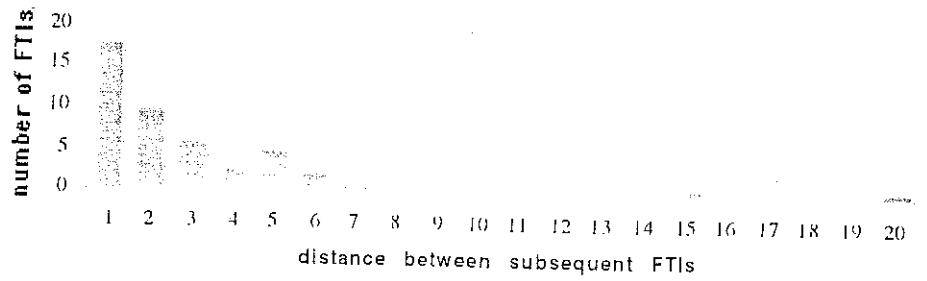


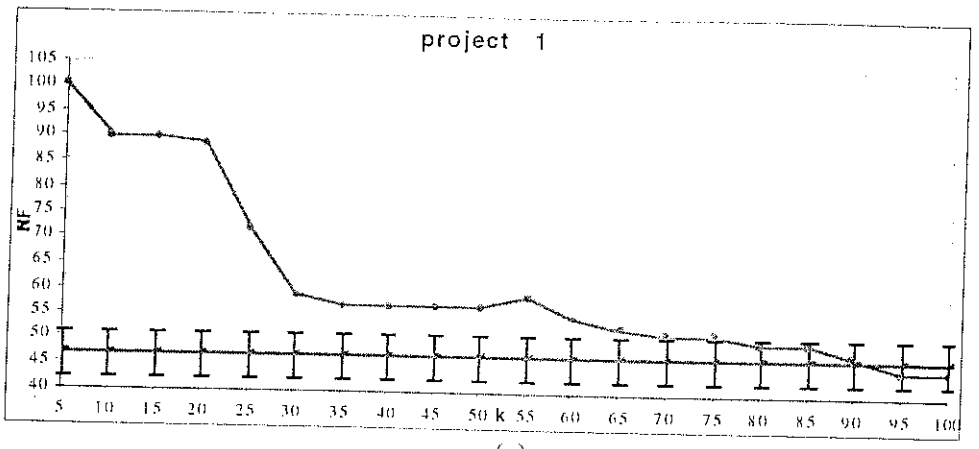
Figure 2: Histogram for the random variable T

In particular, after some analysis, we decided to approximate the prior pmf of T with a *normal truncated distribution*. We derived the normal curve with mean and variance equal to the sample mean and variance, and truncated it between 1 and 20. Since the data we have are grouped within intervals, we then approximated this continuous distribution with a discrete one.

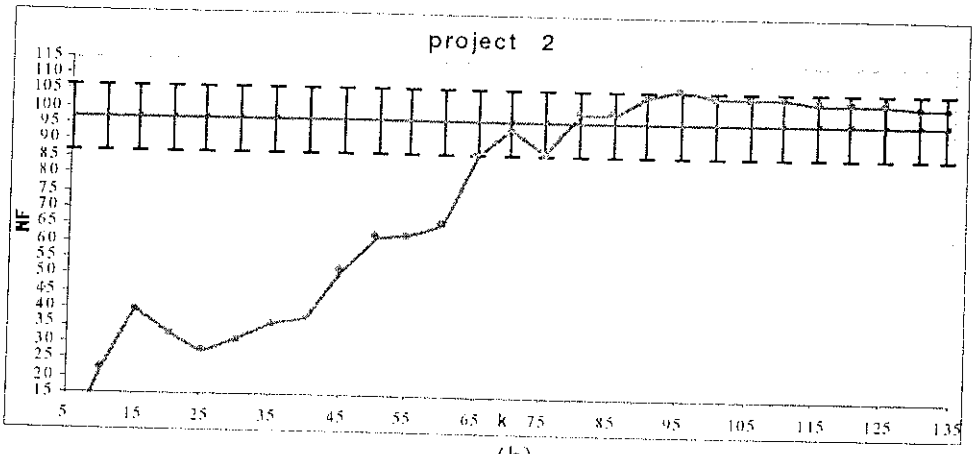
The approach we followed to verify the model was the following. Considering the whole series of test outputs of a product, an intermediate test interval TI_k is taken as the current point. From this point, the cumulative number of failures that will be observed for the whole testing period is estimated applying the Bemar model. For the prediction, therefore, we use the failure data collected from the beginning of the functional test up to the selected point TI_k.

This computation is repeated for progressively longer test intervals (i.e., for greater values of k), for instance after the first 5 TIs, after the first 10 TIs, 15, and so on. In fact, since a Bayesian inference procedure is used, the prediction is progressively updated considering each time a greater amount of collected data.

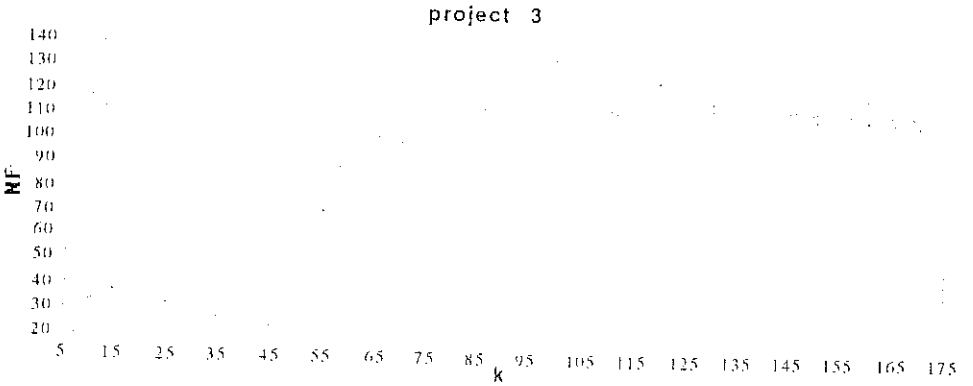
In Figure 3 the results obtained applying the Bemar method to some of the sets of data available are shown. In these figures on the horizontal axis we put the number of test intervals, k, considered to make the prediction, and on the vertical axis the cumulative number of failures, NF, predicted at the end of functional test. The dotted line represents the cumulative number of failures predicted at the end of the functional test using as prior pmf of T the normal truncated distribution. The effect of improvement of the prediction as more test outputs are observed is clearly visible. To compare the results predicted with the real ones, in the figures we drew the actual number of failures counted at the end of the testing (the horizontal line). The strip around the horizontal line marked with vertical segments signs the zone where the relative error of the estimation is below 10%.



(a)



(b)



(c)

Figure 3: Predictions with the Bemar model

In general, for all the case studies considered, we could observe that the model starts with very high errors, but after about a half of the test period, the prediction becomes quite good. We are currently studying other ways to derive a prior pmf for a specific producer from the test result observed in earlier projects. We expect that a prior pmf which fits better to the test process under investigation should converge more quickly to a valid prediction.

4.2 Application to operational test data

We are interested in discovering if and how the Bemar model can be applied as a complementary approach to reliability growth models or in those situations in which the failure data relative to operational testing do not show a reliability increasing trend. For this reason, we also tried our model on some operational test results collected by the same producer during beta testing.

The problem in applying the Bemar model to this kind of data was that operational test results collected previously on similar projects were not available. Therefore we could not apply the criteria described in the previous section for the selection of a prior pmf of T. We hence decided to adopt a uniform prior distribution.

For the rest, the approach to apply the Bemar model to the data collected during the operational phase is the same of that described in Section 4.1: we took an intermediate test interval k as the current point of the operational test, and from this point we predicted the expected final number of failures. This computation has been repeated taking progressively longer periods. We report the results in the figure below.

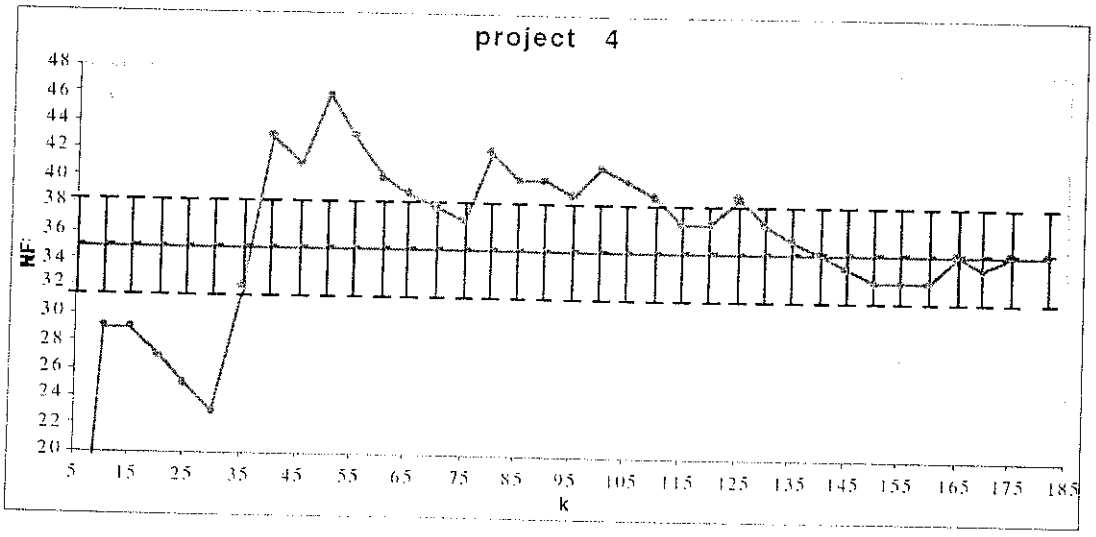


Figure 4: Prediction of the Bemar model for beta testing

The performance of the model becomes acceptable after 110 TIs, over a complessive period of 180 TIs. We must add that attempts to apply standard reliability growth models to these same data were not successful; the problem was that the reliability did not regularly increase, as required by those models.

On the contrary, we expect a worse performance of Bemar over data that exhibit consistent reliability growth. We have tried the model on a set of data taken from the literature (Abdel-Ghaly, 1986). These data are reported as execution times in seconds between successive failures. To apply our model, we have grouped the failure data into test intervals of 600 seconds.

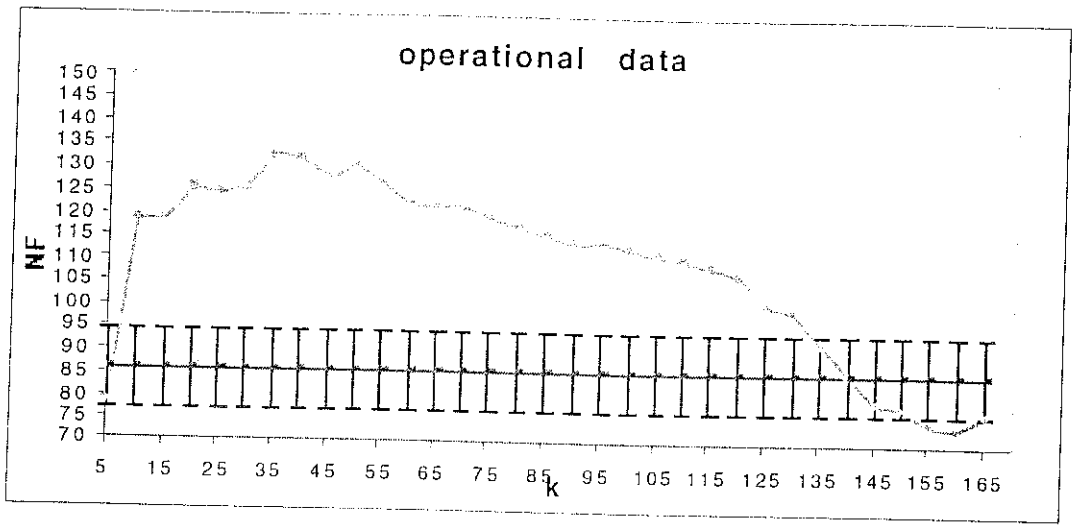


Figure 5: Prediction of the Bemar model for operational testing

5. Conclusions

This work is still in a preliminary stage. We are investigating dynamic models for monitoring and controlling the test process based on observed test results. In this paper we have briefly presented the motivations, the formulation and a few applications of a new model that can be applied to failure data to predict the expected number of failures in future tests. The model is still incomplete, and needs further validation on more data. In particular, the formula used to make the prediction needs to be augmented with some method to estimate in advance the error bound. For the time being, we have evaluated the relative error against known results, and the model performance looks encouraging.

This model assumes that the detected failures are distributed over the whole test period, and that reliability does not exhibit a regular trend. This could be the case for the early test phases, when many failures still remain, and standard reliability growth models cannot yet be applied. In this sense, we believe that this model works in complementary way with reliability growth models, and in fact we intend to investigate an approach to use both models in combination.

Acknowledgements

We thank Emilia Peciola and Gaetano Lombardi of the Research&Development Division of Ericsson Telecomunicazioni S.p.A. in Rome, for providing us with valuable data and information, as well as for useful discussions and interest during the development of this work.

References

A. Abdel-Ghaly, P. Y. Chan, and B. Littlewood, Evaluation of Competing Software Reliability Predictions, *IEEE Tr. On Software Eng.*, Vol. SE-12, No. 9, Sept. 1996, pp. 950-967.

A. Bertolino, G. Lombardi, E. Marchetti, E. Peciola, "Introducing a Reliability Measurement Program into an Industrial Context", *Proc. of ESCOM-ENCRESS 98*, Rome, May 27-29 1998, pp. 277-286.

K. Y. Cai "On Estimating the Number of Defects Remaining in the Software", *J. System Software*, Vol. 40, No. 2, pp. 93-114, February 1998.

S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*, The Benjamin/Cummings Publishing Co., Menlo Park, Ca, 1986.

N. E. Fenton and S. L. Pfleeger, *Software Metrics A Rigorous and Practical Approach*, 2nd Ed., Int. Thomson Comp. Press, 1997.

K. Kanoun, M. Kaaniche, and J. P. Laprie, "Qualitative and Quantitative Reliability Assessment", *IEEE Software*, Vol. 14, No. 2, Mach 1997.

M. R. Lyu (Ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill, 1996.