C. Wohlin and U. Körner, "Software Faults: Spreading, Detection and Costs",
Software Engineering Journal, Vol. 5, No. 1, pp. 33-42, 1990.

Software faults: spreading, detection and costs

by Claes Wohlin and Ulf Körner

The paper considers, through modelling, how software faults are spread throughout the entire life-cycle of a large software product and how fault detection and correction processes will affect the spreading mechanism. The study is further enlarged to incorporate models for cost estimation. The models can be described as being of a qualitative rather than a quantitative nature, in that they highlight the effects of different approaches relative to each other before giving 'exact' values for each approach. The study reveals that the behaviour and consequences of different ways of spreading and detection, as well as different cost mixtures, can be studied and thus understood.

1 Introduction

During the last two decades we have witnessed a dramatic and rapid change in large real-time systems (e.g. telecommunication systems), in that today they are often based on huge software systems. The rapid development on the hardware side, with totally new concepts as well as technologies for increased reliability and availability, has not so far been met by rapid development on the software side, although the latter does often dominate in terms of developing efforts. The demands are great for new techniques, methods and tools throughout all phases of the life-cycle of a system for design as well as for planning. System analysts need new and realistic models to capture a basic system behaviour. Models and metrics are needed for calculations and estimations. As software is becoming a greater and more important part of a system, it is necessary to incorporate metrics and models for software performance analysis [1, 2], together with 'classical system' performance measurements, to get more complete and comprehensive models of a total system. The needs for models and metrics for understanding software measures, such as reliability, availability and effectiveness, cannot be stressed enough [3, 4]. It is not possible to develop new cost-effective systems without taking software performance into account: this new dimension must be added to the developing process. Tools for analysing control situations, for dimensioning and scheduling etc. are still vital issues, but a system analyst can no longer overlook the software [5].

The examples of software products not fulfilling required quality constraints are numerous. These problems can be overcome by a thorough study of software and its performance. The solution to the problem can be divided into different steps:

- *Awakening*, i.e. realising that software is a critical part of system performance.
- *Understanding*, i.e. it is necessary to understand the process of developing software products and its consequences [6] if the intention is to develop a cost-effective system in some sense.
- *Modelling* of software performance.
- *Evaluation* of software performance.
- *Incorporation*, i.e. models and metrics for software performance have to be incorporated with other aspects of system performance to develop cost-effective systems in the future.

This paper will cover parts of the second step, understanding, assuming that most professionals within the software community have entered the first step, awakening. We will, by means of models, try to understand the process of the spreading, detection and costs of software faults.

During system development, which incorporates a number of phases, faults are introduced and removed. The costs caused by a fault are highly dependent on when the fault was introduced, detected and hopefully removed. A fault introduced during the system specification phase will probably have a large impact on a number of software units during the coding phase, i.e. the fault has spread. The

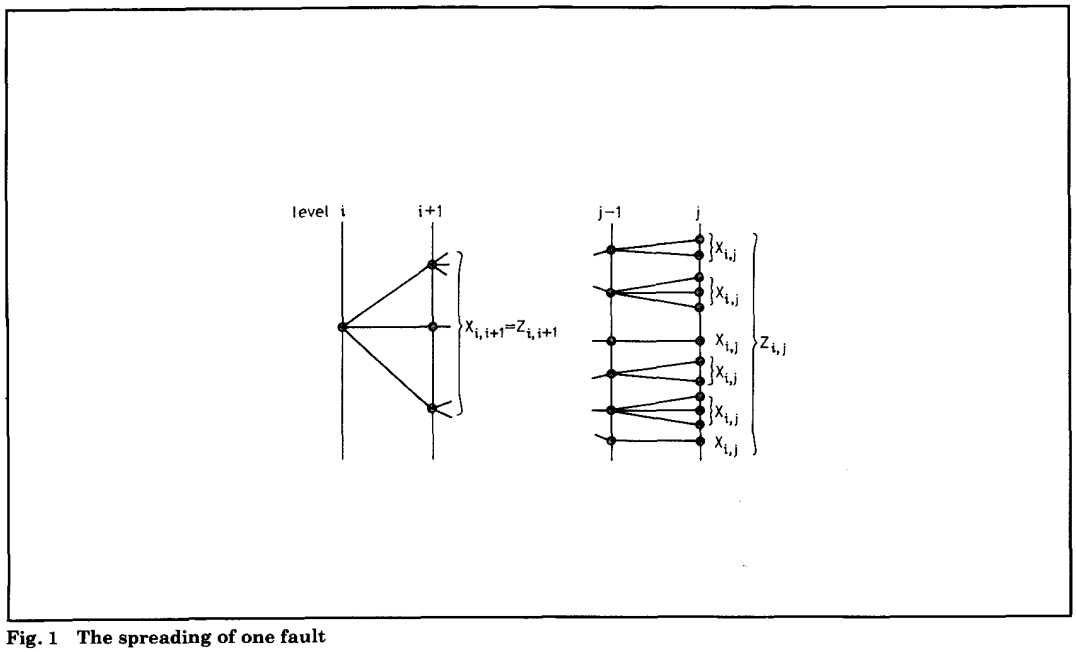

Fig. 1 The spreading of one fault

spreading of a fault depends on the number of phases between introduction and detection as well as to what extent a fault is spread when entering a new phase.

Some general information about faults may be appropriate at this point. An error is made by man and results in a fault in the product. The manifestation of a fault, which means a departure from what the software is supposed to do, is referred to as a failure. These are the suggested standards in IEEE Standard 982, where it is also suggested that a defect is a product anomaly. This may be interpreted as a defect being the result or consequence of a fault in a preceding phase. A fault may spread and cause a number of defects.

The consequences of the spreading and the ability to detect the fault affect the number of defects caused by that fault. This is a critical issue that has to be addressed when developing cost-effective software products. To understand the behaviour of the spreading mechanisms and the question of detection, a model is developed. The model is developed in two steps: first, a pure spreading model, and secondly, this model will be complemented by introducing detection. This model is further enlarged by taking costs into account for defect removals during test and operation, respectively. The last model can also be used to determine an optimal time for field entry.

The primary goal of the models is to understand the mechanism of the spreading of faults and its consequences. It shall be pointed out that the models are in an early state and consequently are very much simplifications of a complex reality. Despite this, they will provide a possibility to gain an understanding of the process studied. The models here will, at least, give qualitative rather than quantitative results.

The mathematics used to develop the models may be found in any basic book on probability theory [7]. The notation used is presented in Appendix 8, and some of the derivations for the spreading model are shown in Appendix 9.

The developed models will be used in a simple example to describe the influence of different parameters in the models. The example will help us understand the different processes (spreading, detection and costs) modelled and draw valuable conclusions, which can probably be extended to be valid in more realistic cases as well.

2 The spreading model

From the very first phases of a system development, i.e. specification, to the last phases before final testing, faults being introduced during one phase will spread to the underlying phases. A fault introduced during the design may spread, i.e. result in a number of defects during a coding phase.

In our models we use an independent assumption, i.e. faults are introduced and spread independently. This is a standard straightforward assumption, which of course can be argued. We know that in most cases it is necessary in order to derive a model which can be solved. In general, experiences from queueing theory modelling, for example, show that the assumption is not critical, i.e. results obtained with the independence assumption can often be verified with simulations. The spreading mechanism depends on the number of phases (or levels) the system may go through from the very beginning of a project to the last level before the testing phase. The number of levels before testing is denoted by l. During each of these levels faults may be introduced. It is assumed that a fault, introduced and detected during the same phase, was never made, i.e. it is not dealt with in our models. Let the r.v. $X_{i,j}$ denote the number of defects on level j caused by one defect on level $j-1$ and $Z_{i,j}$ denote the total number of defects on level j when a fault is introduced on level i. It should be noted that $X_{i,i+1}$ is the number of defects on level $i+1$ caused by one fault on level i and that $Z_{i,i}$ is equal to one, i.e. the fault itself.

From these definitions it is possible to derive a recursive z-transform for $Z_{i,j}$, which, in turn, makes it possible to derive the mean and variance of the total number of defects on level j when the fault was introduced on level i. The derivations are based on standard branch processing [8], and a similar approach has been used earlier for fault propagation for hardware [9]. The emphasis in this paper will be on cost control and product release for software products, whereas in Reference 9 they concentrated on test strategy for hardware. The derivations are shown in Appendix 9.

$$E[Z_{i,j}] = \prod_{k=i+1}^{j} E[X_{i,k}] \quad (1)$$

$$V[Z_{i,j}] = \sum_{k=i+1}^{j} \left(V[X_{i,k}] \prod_{n=i+1}^{k-1} E[X_{i,n}] \prod_{n=k+1}^{j} E[X_{i,n}]^2 \right) \quad (2)$$

where $E[Y]$ is the mean value of r.v. Y and $V[Y]$ is the variance of Y.

These formulas give the mean and variance of the number of defects on an arbitrary level caused by one fault on any preceding level. The total number of faults introduced on level i will be denoted by A_i. This definition and the formulas above make it possible, by taking the appropriate sums, to obtain

- the total number of defects, independent of the level, caused by one fault made on level i;
- the total number of defects caused by faults introduced on level i;
- the total number of defects which is the result of the errors made by the people developing the software product.

The model can, of course, be used with different probability distributions for each $X_{i,j}$, and, when studying the sums mentioned above, different distributions can also be used for each A_i. This means that the consequences of different distributions can be studied. It is important to try to identify realistic distributions in order to obtain good, i.e. realistic and reliable, results. We will, however, restrict ourselves to studying a simple example to describe and illustrate the use of the model.

Example 1

Assume $X_{i,j}$ follows a geometric distribution for $k \geq 1$, i.e.

$$\text{prob}(X_{i,j} = k) = pq^{k-1}$$

where $p + q = 1$. Here, the spreading of defects is not dependent on the current level. This gives us

$$E[X_{i,j}] = \frac{1}{p} \quad V[X_{i,j}] = \frac{q}{p^2} \quad (3)$$

p may be assigned any value between zero and one, because in all these cases we obtain an increase of the number of defects from one phase to another. A low value on p leads to an avalanche-like effect of the spreading increases, which we will see below.

From eqns. 1 and 2, $E[Z_{i,j}]$ and $V[Z_{i,j}]$ are obtained as

$$E[Z_{i,j}] = \left(\frac{1}{p}\right)^{j-i} \quad (4)$$

$$V[Z_{i,j}] = \sum_{k=i+1}^{j} \left(\left(\frac{q}{p^2}\right) \prod_{n=i+1}^{k-1} \frac{1}{p} \prod_{n=k+1}^{j} \frac{1}{p^2} \right) = \frac{1-p^{j-i}}{p^{2(j-i)}} \quad (5)$$

Let us assume that $i = 0$ and study $E[Z_{i,j}]$ and $V[Z_{i,j}]$ for different values of j and p. In this particular case it is possible to derive the distribution for $Z_{i,j}$, but in the general case we will have to adapt a distribution to the two moments derived or approximate the distribution for $Z_{i,j}$ with a normal distribution. The reason for doing this is that we would like to calculate confidence intervals for $Z_{i,j}$. The results are presented in Tables 1 and 2. In Table 1 it is shown how a 90% confidence interval will increase as the distance in levels between introduction and detection increases. In Table 2 it can be seen how the mean number of defects on level j increases for different values on p, i.e. $E[Z_{i,j}]$, when the fault was introduced on level i.

Although one may argue on our choice of the distribution for $X_{i,j}$, the avalanche-like effect of the mean values is a reality. For example, if the defect gives rise to four defects on the underlying level, we will end up with, on average, over 1 000 defects five levels below the level where fault was introduced (see Table 2). These figures show the importance of minimising the time between introduction and detection of faults.

Let us study the total number of defects in the software product. Assume A_i follows a geometric distribution for $k \geq 0$, i.e.

$$\text{prob}(A_i = k) = ab^k$$

where $a + b = 1$. This gives us

$$E[A_i] = \frac{b}{a} \quad V[A_i] = \frac{b}{a^2} \quad (6)$$

Table 1 The number of defects at level j

$p = 0.25$	$E[Z_{i,j}]$	$V[Z_{i,j}]$	90% confidence interval
$j = 1$	4	12	1.18–11.41
$j = 2$	16	240	1.79–47.42
$j = 3$	64	4032	4.26–191.2
$j = 4$	256	65280	14.11–766.4
$j = 5$	1024	1047552	53.50–3067

Table 2 $E[Z_{i,j}]$: the mean number of defects at level j for different values on p

	p				
j	1	0.5	0.25	0.1	0.05
1	1	2	4	10	20
2	1	4	16	100	400
3	1	8	64	1000	8000
4	1	16	256	10000	160000
5	1	32	1024	100000	3200000

Table 3 The total number of defects in the software product

	a			
p	$\frac{1}{3}$	$\frac{1}{6}$	0.1	0.05
1	42	105	189	399
0.5	240	600	1080	2280
0.25	3636	9090	16362	34542

The average of the total number of defects in the software product is presented in Table 3 for different values of a and p. We assume that $i = 0, \ldots 5$, i.e. the total number of defects is calculated after level five. The figures in Table 3 are obtained from

$$\sum_{i=0}^{5} \sum_{j=i}^{5} \frac{b}{a} E[Z_{i,j}] \tag{7}$$

In Table 3 it is seen how the total number of defects at level five vary as a function of a and p. For example, it can be seen that, if on average five faults are introduced at each level ($a = \frac{1}{6}$) and the average spreading factor is equal to four ($p = 0.25$), then we will obtain, on average, over 9000 defects in the product. It shall, however, be observed that these figures are obtained without taking detection into account. The model will be complemented with the possibility of detection in Section 3.

The conclusions from this example are quite obvious and perhaps not the most important result. The conclusions are discussed in Section 5. Perhaps the most important aspects of the example are the possibility of modelling the behaviour of fault spreading, and using different distributions and parameters in the models. Finally, it should be noted that the distributions and their parameters may depend on the levels. In this case we might, for example, let p depend on both the level of introduction (i) and the level of study (j), and the parameter a can be dependent on the level of introduction (i).

3 Spreading and detection

The detection process is modelled by constant probabilities, which are functions of the current level and the level of fault introduction. The probabilities are, of course, assigned values according to the effort that is put into the fault detection mechanism on each level. This gives us the chance to study the impact of different parameters on the spreading and detection process. The model can be used for arbitrary distributions regarding the number of introduced faults on any level, as well as the distribution of $X_{i,j}$. The model gives us the number of undetected faults when entering the testing phase. It makes it possible to study and understand the mechanism of fault spreading and detection. This is a vital issue when trying to develop cost-effective systems.

It shall be observed that the stochastic behaviour of the spreading model is used in the sense that different confidence intervals can be calculated, and that these values can be used to study different cases or outcomes from the spreading model. Let $Z_{i,j}$ denote the outcome from the spreading model, then $Z_{i,j}$ can be set to the values obtained when calculating the 90% confidence interval or the mean value for example. In the formulas below, $Z_{i,j}$ will be used for denoting the outcome and $Z_{i,j}$ will be substituted with the mean value, i.e. $E[Z_{i,j}]$, when the actual values are inserted and for reason of simplicity in the cost model present below. The detection model is in itself deterministic, but based on the results from the stochastic spreading model.

The detection probabilities are denoted by $p_{i,j}$ and defined as the probability to detect one defect on level j when the fault was introduced on level i. This means that, if the outcome is assumed to be $Z_{i,j}$, then $(1 - p_{i,j})^{Z_{i,j}}$ is the probability that none of the $Z_{i,j}$ defects on level j, caused by one fault on level i, will be detected. This is equivalent to the spreading of the fault continuing to level $j + 1$.

It shall be noted that, in reality, the detection process is binary; either we detect a defect or we do not. If a defect is found it is assumed that we track the fault and correct all defects caused by that specific fault (see Fig. 2).

In Fig. 2 the fault is tracked by following the errors from the detected defect to the fault which caused the malfunction. When the fault is corrected, we will go down the tree structure and correct all defects caused by the fault.

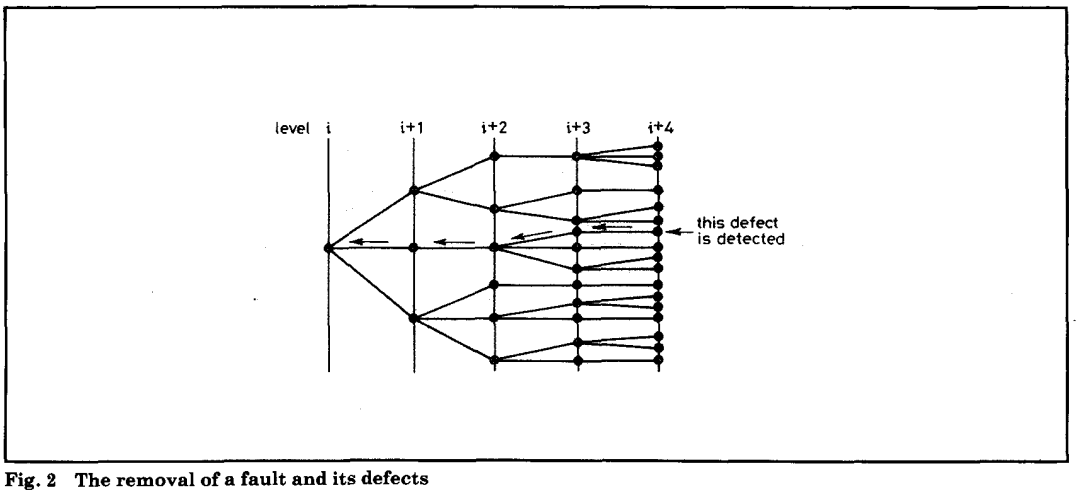

Fig. 2 The removal of a fault and its defects

The binary effect of detection is best seen by studying the spreading mechanism and the probabilities that none of the defects has been detected when we reach level j. In particular, if we assume that the outcome is equal to $E[Z_{i,j}]$, we will obtain the mean number of defects on level j caused by one fault from level i, when we have introduced the possibility of detection in the model.

We will study the probabilities that none of the defects has been detected when we reach level j and the mean number of defects remaining on level j (see example 2). It is also possible to study all cases from the best to the worst, where the worst, of course, is the result obtained in Section 2, i.e. no detection at all.

So far in this Section we have been concerned with the detection of defects caused by one fault. This may be extended to a study of the total number of defects in the same way as discussed above.

Example 2

We assume that $X_{i,j}$ and A_i belong to the same distributions as in example 1 and to this we add the assumption that $p_{i,j} = c$, where $0 \leqslant c \leqslant 1$. The latter assumption is made here for reason of simplicity. Of course, the probabilities for detection may depend on the levels and can be set to realistic values according to the effort and techniques used during different phases in the software life-cycle.

First, we will study the probability that none of the defects will be detected for various different cases (see Table 4). The values in the lower right part of the table (indicated by ≈ 0) are all less than 10^{-10} and are thus very small, due to the fact that the chosen spreading factor is set to 4 for all levels. In this example it is assumed that the number of defects on a specific level, caused by one fault on a higher level, is set to the mean. This can be compared with the worst case, which was presented in example 1, i.e. no detection at all.

This means that the probability that none of the defects is detected on level j when the error was made on level i, can be written as

$$P_{i,j} = (1 - p_{i,j})^{Z_{i,j}} \tag{8}$$

where $p_{i,j} = c$ and the outcome $(Z_{i,j})$ is assumed to be equal to $E[Z_{i,j}]$, which is obtained from example 1.

To calculate the number of remaining defects, it is necessary to make the study one level at the time, because the probability of detection will have an impact on the number of defects remaining at the next level. Let us define $\hat{Z}_{i,j}$ as the number of defects remaining to be spread to the next level. The difference between this new definition and that for $Z_{i,j}$ shall be observed. The latter shall still be interpreted as the number of defects on level j caused by a fault from level i. Eqns. 9 and 10 may clarify the relationships

$$\hat{Z}_{i,j} = P_{i,j} Z_{i,j} \tag{9}$$

$$Z_{i,j} = X_{i,j} \hat{Z}_{i,j-1} \tag{10}$$

The definition of $P_{i,j}$ is found above. The formulas can be used to calculate the number of defects remaining at each level for different cases. In particular, we can study the mean values, assuming independence between the random variables involved, for which an example is shown in Table 5.

The total number of defects is calculated in a similar manner as in example 1, i.e. we use eqn. 8 and the results from Table 5. The results presented in Table 6 can be compared with those in Table 3. In Table 6 the average number of defects in the software product when detection has been introduced is presented, whereas Table 3 contained the average number of defects in the software product without having introduced the possibility of detection. We see that the total number of defects has been lowered to about a third in some cases due to the detection process; see, for example, how the 9090 has been lowered to 3180. The results also indicate how sensitive the total number of defects are to the detection probabilities, especially when there are many defects. Otherwise, they are very hard to detect, and we have to have a higher probability to detect a specific defect in order to lower the total number of defects significantly.

The gain with early detection can easily be seen from the

Table 4 The probability that none of the defects is detected at level j

	j				
c	1	2	3	4	5
0.01	0.961	0.851	0.526	0.076	3×10^{-5}
0.05	0.815	0.440	0.038	2×10^{-6}	≈ 0
0.1	0.656	0.185	0.001	≈ 0	≈ 0
0.3	0.240	0.003	≈ 0	≈ 0	≈ 0
0.5	0.062	2×10^{-5}	≈ 0	≈ 0	≈ 0

$p = 0.25$

Table 5 The mean number of defects at level j

j	$E[Z_{i,j}]$	$P_{i,j}$	$\hat{E}[Z_{i,j}]$
1	4	0.961	3.84
2	15.36	0.857	13.16
3	52.64	0.589	31.00
4	124.00	0.288	35.71
5	142.84	0.238	34.00

$p = 0.25$ and $c = 0.01$

Table 6 The average number of defects in the software product

	a			
p	$\frac{1}{3}$	$\frac{1}{6}$	0.1	0.05
1	41.6	104	187	395
0.5	213	533	959	2025
0.25	1272	3180	5724	12084

$c = 0.01$

values in the tables, i.e. the higher probability for detection the better. Once again we have seen that it would be possible to model an important aspect of faults and their behaviour. We can now study the consequences of both spreading and detection. This means that we have obtained a way to study the effects of different sets of parameters, and if we could obtain realistic distributions or at least mean values, this model will be a valuable tool in the planning and control of a software project. These aspects are, however, not sufficient to see the consequences of faults on the total cost. Therefore we will develop a cost model in Section 4.

4 The cost model

Now we extend our basic model to grasp cost-effectiveness and we assume that

- the costs from the phases before testing depend only on the spreading and detection of faults;
- the time spent in phases before testing is independent of the number of defects present and the detection;
- no new faults are introduced during testing and the cost during the testing phase depends on two factors: the time spent in the phase and the number of removed faults and how they have spread during earlier phases;
- the cost for fault removal during operation depends only on the number of faults remaining when the product is released and in which phase they were introduced;
- costs depending on the market and customer behaviour are neglected. (This assumption is further discussed below).

These assumptions lead to a total cost, which can be seen as the sum of three parts, i.e. $C = C_1 + C_2 + C_3$; where C_1 is the cost for fault removal before the testing phase; C_2 that during the testing phase; and C_3 that when the system is in operation. C_2 and C_3 are time-dependent and they do especially depend on the time when the product is released.

The cost model will provide a possibility to study the costs based on spreading and detection of faults before the testing phase, time-dependent detection during testing and operation, and the fault removal costs during test and operation taking the behaviour before the testing phase into account, i.e. the spreading and detection of faults.

The model will give us the chance to study the consequences and impact of different parameters on the cost. The model shall be seen as a tool for understanding, and as we learn more and more about the actual behaviour, the model can be adapted or changed according to the lessons learned.

We will begin by deriving the cost before the testing phase, i.e. C_1. Here the costs will be derived in terms of mean values, but it should be emphasised that it is possible to study different cases through variation of the parameters in both the spreading and detection model and variation of the various costs defined in the model in this section too. Let $K_{i,j}$ be the cost to correct a defect on level j caused by an error made on level i, and C_i the total cost, before the testing phase, to correct a fault introduced on level i and the defects which have occurred due to the error made. $K_{i,i}$ is the cost for the correction of the fault itself. These definitions give

$$C_i = \sum_{j=i}^{l} E[Z_{i,j}]K_{i,j} \tag{11}$$

where, as before, l is the number of levels and $E[Z_{i,j}]$ is the number of defects on level j caused by a fault on level i. It is assumed that all faults will eventually be detected and removed. It should be noted that $E[Z_{i,j}]$ is the mean number of defects remaining at level j and it can be calculated according to different detection probabilities, as discussed in preceding Sections in the spreading and detection model.

The total cost for fault removal before the testing phase is obtained as

$$C_1 = \sum_{i=0}^{l} E[A_i] \sum_{j=i}^{l} E[Z_{i,j}]K_{i,j} \tag{12}$$

where $E[A_i]$ is the mean number of errors made on level i.

We assume that C_1 covers all costs for fault and defect removal before the testing phase; even if the fault is detected during testing, C_1 is the cost for correcting the documents produced before the testing phase. We will, however, add an extra cost which depends on the spreading of faults during the development phases when we calculate the cost for fault removal during testing and operation, respectively. This cost will be discussed below. The next step will be to calculate these costs, which will be denoted by C_2 and C_3.

The mean number of remaining faults has to be calculated. The reason for needing the number of remaining faults, and not just the number of defects, is that when a failure occurs it is assumed that the fault is located and all defects corrected. This means that the number of malfunctions during testing and operation is equal to the number of faults or at least those of them that are detected.

Let the remaining fault vector be denoted by R, where an element r_i is the mean number of faults introduced on level i but not detected before the testing phase. It shall once again be noted that r_i could have been defined for studying different outcomes instead of just the mean value as here, but for simplicity we have chosen to present the formulas for the cost model with the mean values. The formulas can, however, easily be transformed to incorporate the study of different outcomes. This means that the stochastic modelling of the spreading model is used, since it is possible to study different outcomes based on the confidence intervals in the spreading model. The cost model itself is, however, deterministic, but it is based on the outcomes from the stochastic spreading model. r_i can be written as

$$r_i = E[A_i] \prod_{j=i+1}^{l} (1 - p_{i,j})^{E[Z_{i,j}]} \tag{13}$$

where the product models the probability that none of the defects is detected from level $i + 1$ to level l. The number of defects originating from one fault introduced on level i in the testing phase is denoted by $E[Z_{i,l+1}]$, where level $l + 1$ is the testing phase.

Let the total number of remaining faults at the test start be denoted by $M_{tot}(0)$, i.e.

$$M_{tot}(0) = \sum_{i=0}^{l} r_i \tag{14}$$

The remaining number of faults after testing a time t is denoted by $M_r(t)$, and it is assumed that the function, according to which the number of faults decreases during testing, is called $f_{l+1}(t)$. This function may, for example, depend on the detection rate and the elapsed time. This is

further discussed in References 10 and 11. These definitions lead to

$$M_r(t) = M_{tot}(0) f_{l+1}(t) \tag{15}$$

Consequently, the number of detected faults at t can be written as

$$M_d(t) = M_{tot}(0) - M_r(t) = M_{tot}(0)(1 - f_{l+1}(t)) \tag{16}$$

To calculate the cost for testing, we define C_f as the cost to test one time unit independent of whether faults are detected or not; C_{l+1} as the cost for removing a fault during testing; and $C_{i,l+1}$ as the extra cost for not finding a fault introduced on level i until level $l+1$, i.e. the testing phase. C_{l+1} may include costs for writing fault reports, administration and other extra costs during testing because the fault was detected. The cost is independent of the level of introduction. On the contrary, $C_{i,l+1}$ depends on the level of introduction and it should not be mixed up with C_1. $C_{i,l+1}$ is used to take into account all extra costs due to the late detection, whereas the ordinary cost for correcting the documents before the testing phase has been taken care of by C_1. The extra cost shall, for example, cover the effect of moving development staff back into the project for fault correction and restarting several development activities, say, block design and coding. This may lead to another project being delayed, and the cost for this ought to be charged to the project where the faults have been found. It may be difficult to assign a realistic value on the extra cost, but it ought to be possible to estimate if data from a couple of projects have been collected.

The probability that a detected fault was introduced on level i will be approximated by

$$\frac{\hat{E}[Z_{i,l}]}{\sum_{k=0}^{l} \hat{E}[Z_{k,l}]} \tag{17}$$

The assumption that this factor is constant over time is reasonable, since all defects are considered to be equal in size, i.e. they are detected with the same probability. This means that even after some time the above approximation will be suitable, since faults already detected will, on average, follow the proportion given by the equation.

This probability will be used to calculate the mean extra cost due to the level of introduction. Thus the cost for fault removal during testing is given by

$$C_2(t) = C_f t + M_d(t)\left(C_{l+1} + \sum_{i=0}^{l} C_{i,l+1} \frac{\hat{E}[Z_{i,l}]}{\sum_{k=0}^{l} \hat{E}[Z_{k,l}]}\right) \tag{18}$$

The cost for fault removal during operation is derived in a similar manner as $C_2(t)$. We still assume that we are interested in updating our documents because of faults found during operation. This is a reasonable assumption, otherwise the product will soon be unmaintainable and consequently obsolute in the near future.

The same assumption will be used for the probability that the fault was introduced on level i. Let t_r denote the time when the product is released, and let $f_{l+2}(t)$ be the function for how the number of faults decreases during operation. $f_{l+2}(t)$ is only defined for values of t larger than t_r. Let $M_{tot}(t_r)$ be the number of remaining faults when the operational phase starts. This means that

$$M_r(t) = \begin{cases} M_{tot}(0) f_{l+1}(t) & t \leqslant t_r \\ M_{tot}(t_r) f_{l+2}(t) & t \geqslant t_r \end{cases} \tag{19}$$

where $M_{tot}(t_r) = M_{tot}(0) f_{l+1}(t_r)$ and $M_d(t)$ for $t \geqslant t_r$ is of course equal to $M_{tot}(t_r) - M_r(t)$. From these formulas it is possible to determine the cost during operation as

$$C_3(t) = M_d(t)\left(C_{l+2} + \sum_{i=0}^{l} C_{i,l+2} \frac{\hat{E}[Z_{i,l}]}{\sum_{k=0}^{l} \hat{E}[Z_{k,l}]}\right) \tag{20}$$

$f_{l+2}(t)$ only fills a function if the product is taken out of operation before all faults have been removed. If we assume that $t \to \infty$, i.e. all faults will be found and removed, we can conclude that the three costs can be calculated from

$$C_1 = \sum_{i=0}^{l} E[A_i] \sum_{j=i}^{l} E[Z_{i,j}] K_{i,j} \tag{21}$$

$$C_2(t_r) = C_f t_r + M_d(t_r)\left(C_{l+1} + \sum_{i=0}^{l} C_{i,l+1} \frac{\hat{E}[Z_{i,l}]}{\sum_{k=0}^{l} \hat{E}[Z_{k,l}]}\right) \tag{22}$$

$$C_3(t_r) = M_{tot}(t_r)\left(C_{l+2} + \sum_{i=0}^{l} C_{i,l+2} \frac{\hat{E}[Z_{i,l}]}{\sum_{k=0}^{l} \hat{E}[Z_{k,l}]}\right) \tag{23}$$

The latter two depend on the time for release of the product. This fact can be used to determine an optimal time for field entry by minimising the total cost for fault removal, i.e. to find the value of t_r which minimises the cost. (This will be discussed in more detail in example 3.)

Before anyone objects that an important cost is missing, we will discuss this cost and try to explain why we have chosen to ignore it in the model. The cost which has been ignored so far is the market cost C_m, which depends mainly on two things; the release time t_r and the number of remaining faults at the release time $M_{tot}(t_r)$.

The cost depends on several market aspects; for example, goodwill, policy and market shares. The two parts, i.e. t_r and $M_{tot}(t_r)$, which constitute the market costs can be described partly as

- the part which depends on the release time of the product. The cost (or perhaps gain) comes from the departure from the predetermined target date. An overdraft will cause a cost because of missed shares of the market or the market may be annoyed if the product is delayed. If the product is released before the target date, we may win market shares and general goodwill for the company, which will be a gain in the long run.
- the part which depends on the number of remaining faults at the release time. This cost comes mainly from the quality of the product released. The quality of the product should be related to similar products on the market.

The market costs are very complex to determine, and therefore we have chosen to neglect them. These effects should be taken into account and incorporated in the entire model when the model has been in use and calibrated, but they are hard to grasp at this early stage. It will not be fruitful to thoroughly investigate the market aspects of the costs for faults until the model has been evaluated.

It should also be noted that we have not taken into consideration the time spent in the phases before the testing phase. Let us assume that the search for faults can be done in parallel with development and that the search for faults will not prolong the different phases in any significant way. Another solution would be to study the transition between two consecutive phases in the same way as has been done

with the testing and operational phases, although this would probably cost more effort than we would gain from this approach.

It has, however, been shown here that it is possible to estimate costs and probably optimise the release time in order to minimise the total cost due to faults. The cost model incorporates the aspects from the spreading and detection model and ought to be a valuable tool when planning and controlling software projects.

The estimation of the total number of faults remaining when the test phase starts, $M_{tot}(0)$, can be used in software reliability models, for example [3, 12, 13]. This can be done, since in many reliability models the initial number of faults is a parameter. The reliability models are probably the ones to be used for the functions $f_{l+1}(t)$ and $f_{l+2}(t)$. The problem of determining which model to use will not be discussed any further here. Criteria for choosing models are discussed in References 14 and 15.

Example 3

This example will illustrate how the cost model can be used for determining a time for optimal field entry. The problem of optimal field entry is also discussed in References 16 and 17, although the effects of fault spreading and detection are not taken into account. To determine an optimal time to release the product, the total cost for fault removal will be minimised.

In this example we will assume that $p = 0.25$, $c = 0.01$ and $a = 0.05$. We also assume that the testing phase is level number 6, i.e. $l = 5$. The costs $K_{i,j}$ shall be estimated or, possibly better, be found in the data library of the company.

It is now possible to calculate C_1, which is given by

$$C_1 = \sum_{i=0}^{5} E[A_i] \sum_{j=i}^{5} E[Z_{i,j}]K_{i,j} \tag{24}$$

where $E[Z_{i,j}]$ is obtained from example 2 and $E[A_i] = b_i/a_i = b/a = 19$.

This cost will, however, not be of interest when studying the time for optimal field entry, since it is independent of the release time t_r.

We need to determine the parameters in C_2 and C_3 in order to minimise the total cost. Some of them are obtained from example 2, i.e. the vector R and $\hat{E}[Z_{i,l}]$ for $i = 0, \ldots, 5$. R is given by

$$R = (r_0, r_1, r_2, r_3, r_4, r_5)$$
$$= (0.63, 2.66, 9.22, 15.65, 18.26, 19.00) \tag{25}$$

The sum of the elements in R should be compared with the total number of faults introduced. The sum of R, which is equal to $M_{tot}(0)$, is 65.42, whereas the total number of faults introduced is $\sum_{i=0}^{5} E[A_i] = 6 \times 19 = 114$. The values of $\hat{E}[Z_{i,j}]$ can also be written as a vector

$$\hat{E}[Z_l] = (\hat{E}[Z_{0,l}], \hat{E}[Z_{1,l}], \hat{E}[Z_{2,l}], \hat{E}[Z_{3,l}], \hat{E}[Z_{4,l}], \hat{E}[Z_{5,l}])$$
$$= (33.95, 35.72, 31.00, 13.16, 3.84, 1) \tag{26}$$

The costs are assumed to be found in the project parameters database of the company, i.e. we have compared the conditions for the project with earlier projects and obtained the following estimates:

$$C_f = 50, \quad C_{l+1} = 10 \quad \text{and} \quad C_{l+2} = 100 \tag{27}$$

$C_{i,l+1}$ and $C_{i,l+2}$ have also been found through earlier experiences; let us assume that we have obtained them as shown in Table 7.

Table 7 The extra costs due to late fault detection

Extra costs	i = 0	1	2	3	4	5
$C_{i,l+1}$	6	5	4	3	2	1
$C_{i,l+2}$	10	9	8	7	6	5

For simplicity we assume here that $f_{l+1}(t) = f_{l+2}(t)$ and that the number of faults decreases according to an exponential distribution, i.e.

$$f_{l+1}(t) = f_{l+2}(t) = \exp^{-\phi t} \tag{28}$$

where ϕ is a constant. This function is equal to how the mean number of faults decreases according to two well known software reliability models [12, 13]. although they assume different distributions. If we determine ϕ, it would be possible to minimise the total cost and determine the optimal time for field entry. Let us assume that we have obtained ϕ from our database and that it has been assigned the value 0.1.

We should minimise

$$C_{tot}(t_r) = C_1 + C_2(t_r) + C_3(t_r) \tag{29}$$

where C_1, $C_2(t)$ and $C_3(t)$ are obtained from eqns. 21, 22 and 23, respectively.

If we let

$$G_n = C_n + \sum_{i=0}^{l} C_{i,n} \frac{\hat{E}[Z_{i,l}]}{\sum_{k=0}^{l} \hat{E}[Z_{k,l}]} \tag{30}$$

then we should minimise

$$C_{tot}(t_r) = C_1 + C_f t_r + M_{tot}(0)(1 - \exp^{-\phi t_r}) \times G_{l+1} + M_{tot}(0) \exp^{-\phi t_r} G_{l+2} \tag{31}$$

The minimum value for this function is obtained for

$$t_r = -\frac{1}{\phi} \ln \left(\frac{C_f}{M_{tot}(0)\phi(G_{l+2} - G_{l+1})} \right) \tag{32}$$

We have $\phi = 0.1$, $C_f = 50$, $M_{tot}(0) = \sum_{i=0}^{5} r_i = 65.42$, $G_{l+1} = 14.67$ and $G_{l+2} = 108.67$. From these figures we observe that the minimum value of the total cost is obtained for $t_r = 25.1$. This value means that we should test 25.1 time units and that the minimum value of the cost is obtained when only 5.3 faults remain when the product is released. The initial number of faults when the test phase started was 65.42. Fig. 3 shows $C_{tot}(t) - C_1$ for different probabilities of detection, c.

It can be seen in the Figure how the cost and optimal time for field entry decrease, with higher detection probabilities during the development phases. It shall be noted that the total number of faults when the test phase starts is lower for a higher c, but the remaining number of faults at the optimal time for release is about the same for the three cases. This is because the same costs have been assumed.

The actual values in the example are of little interest. The important thing to point out is the necessity and importance of adopting or adapting such a model to collect data from projects and products and store them in a database and the

possibilities this will give us to plan and control software projects.

In the example we have seen how it would be possible to estimate a time for optimal field entry based on the spreading and detection during phases before the test phase and time-dependent fault removal during testing and operation. Proper distributions and parameters for a model like this would make it an invaluable tool for any software project manager.

5 Conclusions

Three important issues have been discussed: spreading, detection and costs. Each has been implemented in models, which has led to a greater understanding of the consequences of software faults. The model of spreading and detection leads to a greater insight into the problems of software faults before the testing phase. The cost model shows the impact of the spreading and detection of faults on total cost for fault removal, and it also shows how the fault removal rate during testing and operation, respectively, affects the total cost. The time for optimal field entry is an important spin-off from the cost model.

The general conclusion from the study is that it is possible to model the behaviour of software faults in order to understand the behaviour and consequences of it. The understanding will make it easier to develop cost-effective systems in the future.

It can be stated that

- the spreading of faults is a costly fact;
- the total cost is very sensitive to small variations in some of the parameters; for example, the spreading factor and the probabilities for detection;
- the time between introduction and detection of faults is an important factor when calculating the total cost;
- it is possible to estimate a time for optimal field entry based on spreading, detection and costs.

These conclusions from the proposed models, which are supported empirically elsewhere [18], lead to a number of necessary actions that have to be taken in order to cope with the problems:

- better methods are needed for early fault removal to make the time between fault introduction and detection shorter;
- models and metrics, for keeping track of the number of remaining faults, are invaluable tools for staying in control of the failure process; for example, software structure metrics and software reliability models. Fault content estimations are further discussed in Reference 19;
- better description techniques and methods for transformation of documents from one phase to another are needed;
- resource allocation during testing ought to be based on the time for optimal field entry and the predetermined target date.

Finally these studies highlight the necessity of modelling the software as well. Models of the software and its behaviour will make it possible to stay in control of software products, and they are also invaluable tools in the planning process. The use of models for software products is a large step towards complete cost-effective systems where all aspects have been considered.

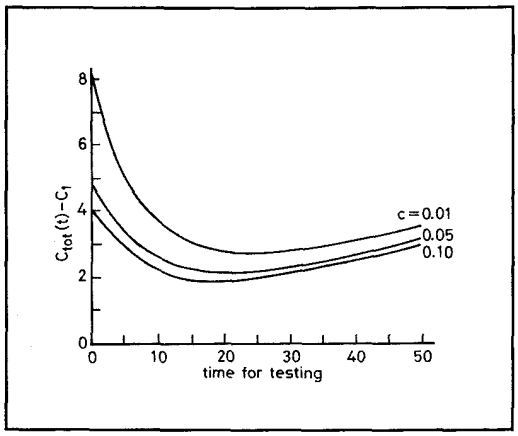

Fig. 3 Cost as a function of time for testing

6 Acknowledgment

Many thanks to Bo Lennselius of the Department of Communication Systems, Lund Institute of Technology, Lund, Sweden for enlightening discussions and valuable comments throughout the work on this paper.

7 References

1 BOEHM, B.: 'Software engineering economics' (Prentice Hall Inc., Englewood Cliffs, USA, 1981)
2 CONTE, S.D., DUNSMORE, H.E., and SHEN, V.Y.: 'Software engineering metrics and models' (The Benjamin/Cummings Publishing Company Inc., Menlo Park, California, USA, 1986)
3 GOEL, A.L.: 'Software reliability models: assumptions, limitations and applicability', *IEEE Trans.*, 1985, **SE-11,** (12), pp. 1411–1423
4 WOHLIN, C., and VRANA, C.: 'A quality constraint model to be used during the test phase of the software life cycle'. Proc. 6th Int. Conf. of Software Engineering for Telecommunication Switching Systems, 1986, pp. 136–141
5 KITCHENHAM, B.A., and WALKER, J.G.: 'The meaning of quality' *in* BARNES, D., and BROWN P., (Eds.): 'Software engineering '86' (Peter Peregrinus Ltd., Stevenage, UK, 1986) Chap. 26, pp. 393–406
6 WOHLIN, C.: 'Software testing and reliability for telecommunication systems' *ibid.*, Chap. 4, pp. 27–42
7 FELLER, W.: 'An introducton to probability theory and its applications' (John Wiley & Sons, New York, USA, 1957)
8 COX, D.R., and MILLER, H.D.: 'The theory of stochastic processes' (Chapman and Hall, London, UK, 1965)
9 SMALL, and FAULKNER, in Proc. 13th Int. Symposium on Fault-tolerant Computing, 1983
10 ADAMS, E.N.: 'Optimizing preventive service of software products', *IBM J. Res. Dev.*, 1984, **28,** (1), pp. 2–14
11 LITTLEWOOD, B.: 'Stochastic reliability-growth: a model for fault-removal in computer-programs and hardware-designs', *IEEE Trans.*, 1981, **R-30,** (4), pp. 313–320
12 JELINSKI, Z., and MORANDA, P.: 'Software reliability research' Statistical Computer Performance Evaluation, 1972, pp. 465–484
13 GOEL, A.L., and OKUMOTO, K.: 'Time-dependent error-detection rate model for software reliability and other performance measures', *IEEE Trans.*, 1979, **R-28,** (3), pp. 206–211
14 IANNINO, A., MUSA, J.D., OKUMOTO, K., and LITTLE-

WOOD, B.: 'Critieria for software reliability model comparisons', *IEEE Trans.*, 1984, **SE-10,** (6), pp. 687–691
15 WOHLIN, C.: 'Some new aspects on software reliability model comparisons'. *In* Proc. Software Engineering '88, 1988, pp. 38–42
16 KŘTĚN, O.J., and LEVY, D.: 'Software modelling for optimal field entry'. *In* Proc. Annual Reliability and Maintainability Symposium, 1980, pp. 410–414
17 ROSS, S.M.: 'Software reliability: the stopping rule problem', *IEEE Trans.*, 1985, **SE-11,** (12), pp. 1472–1476
18 FAGAN, M.E. 'Design and code inspection to reduce errors in program development', *IBM Syst. J.*, 1976, **15,** (3), pp. 182–211
19 LENNSELIUS, B., WOHLIN, C., and VRANA, C.: 'Software metrics: fault content estimation and software process control', *Microprocess. Microsys.*, 1987, **11,** (7), pp. 365–375

8 Appendix

Notation

l = number of levels
i = level of fault introduction
j = level of study
$X_{i,j}$ = the number of defects on level j caused by one defect on level $j-1$ when the fault was introduced on level i. In particular, $X_{i,i+1}$ is the number of defects on level $i+1$ caused by the fault on level i
$Z_{i,j}$ = the total number of defects on level j when the fault was made on level i. In particular, $Z_{i,i}$ is equal to one, i.e. the fault itself
A_i = the total number of faults introduced on level i
p and q = probabilities in the distribution for $X_{i,j}$, i.e. $\text{prob}(X_{i,j} = k) = pq^{k-1}$
a and b = probabilities in the distribution of A_i, i.e. $\text{prob}(A_i = k) = ab^k$
$p_{i,j}$ = detection probabilities. $p_{i,j}$ is the probability to detect a fault introduced on level i on level j
c = the detection probabilities are assigned this value in the presented examples
$P_{i,j}$ = the probability that none of the defects belonging to a fault introduced on level i is detected on level j
$\hat{Z}_{i,j}$ = the number of defects remaining to be spread to the next level
C_1 = the cost for fault removal before the testing phase
C_2 = the cost for fault removal during testing; this cost is time-dependent
C_3 = the cost for fault removal during operation; this cost is time-dependent
$K_{i,j}$ = the cost to correct a defect on level j caused by an error made on level i. In particular, $K_{i,i}$ is the cost for the correction of the fault itself
C_i = the total cost, before the testing phase, to correct a fault introduced on level i and the defects which have occurred due to the error made
R = the remaining fault vector when the test phase is to start. Its elements are denoted by r_i
$l+1$ = level $l+1$ is the testing phase
$l+2$ = level $l+2$ is the operational phase
$M_{tot}(0)$ = the total number of remaining faults at the start of the testing phase
$M_r(t)$ = the remaining number of faults at time t
$M_d(t)$ = the number of detected faults at time t
$f_{l+1}(t)$ = the function according to which the number of faults decreases during testing
C_f = the cost to test one time unit independent of whether faults are detected or not
C_{l+1} and C_{l+2} = the cost for removing a fault during testing and operation, respectively
$C_{i,l+1}$ and $C_{i,l+2}$ = the extra costs for not finding a fault introduced on level i until the testing and operational phase, respectively
t_r = the time when the product is released
$f_{l+2}(t)$ = the function according to which the number of faults decreases during operation
$M_{tot}(t_r)$ = the total number of remaining faults at the start of the operational phase
ϕ = constant used in $f_{l+1}(t)$ and $f_{l+2}(t)$
G_n = variable introduced to simplify the notation (see eqn. 30)

9 Appendix

Derivation of formulas for the spreading

The definitions in Appendix 8 lead to

$$Z_{i,j} = Z_{i,j-1} X_{i,j} \tag{33}$$

By introducing the generating functions

$$P_{Z_{i,j}}(z) = \sum_{k=0}^{\infty} P(Z_{i,j} = k) z^k \tag{34}$$

and

$$P_{X_{i,j}}(z) = \sum_{k=0}^{\infty} P(X_{i,j} = k) z^k \tag{35}$$

we obtain

$$P_{Z_{i,j}}(z \mid Z_{i,j-1} = y) = (P_{X_{i,j}}(z))^y \tag{36}$$

$$\Rightarrow$$

$$P_{Z_{i,j}}(z) = \sum_{y=0}^{\infty} (P_{X_{i,j}}(z))^y P(Z_{i,j-1} = y) \tag{37}$$

$$\Rightarrow$$

$$P_{Z_{i,j}}(z) = P_{Z_{i,j-1}}(P_{X_{i,j}}(z)) \quad j \geq i+2 \tag{38}$$

i.e.

$$P_{Z_{i,j}}(z) = \begin{cases} z & j = i \\ P_{X_{i,j}}(z) & j = i+1 \\ P_{Z_{i,j-1}}(P_{X_{i,j}}(z)) & j \geq i+2 \end{cases} \tag{39}$$

By taking the first derivatives of $P_{Z_{i,j}}(z)$ and letting $z \to 1$, we obtain

$$E[Z_{i,j}] = \prod_{k=i+1}^{j} E[X_{i,k}] \tag{40}$$

$$V[Z_{i,j}] = \sum_{k=i+1}^{j} \left(V[X_{i,k}] \prod_{n=i+1}^{k-1} E[X_{i,n}] \times \prod_{n=k+1}^{j} E[X_{i,n}]^2 \right) \tag{41}$$

The paper was first received on 3rd May 1988 and in revised form on 25th September 1989.

The authors are with the Department of Communication Systems, Lund Institute of Technology, Box 118, S-221 00 Lund, Sweden.