# Optimal Resource Allocation for the Quality Control Process

Pankaj Jalote
Department of Computer Sc. & Engg.
Indian Institute of Technology Kanpur
Kanpur, INDIA - 208016
jalote@cse.iitk.ac.in

Bijendra Vishal
Department of Computer Sc. & Engg.
Indian Institute of Technology Kanpur
Kanpur, INDIA - 208016

## Abstract

*Software development project employs some Quality Control (QC) process to detect and remove defects. The final quality of the delivered software depends on the effort spent on all the QC stages. Given a quality goal, different combinations of efforts for the different QC stages may lead to the same goal. In this paper we address the problem of allocating resources to the different QC stages, such that the optimal quality is obtained. We propose a model for the cost of QC process and then view the resource allocation among different QC stages as an optimization problem. We solve this optimization problem using non-linear optimization technique of Sequential Quadratic Programming. We also give examples to show how a sub-optimal resource allocation may either increase the resource requirement significantly or lower the quality of the final software.*

## 1 Introduction

Software development is typically a multistage process, each stage performing some defined transformations. Defects get injected in different stages of development. As the cost of removing defects increases with latency [3], in an attempt to detect and remove defects close to their injection, quality control (QC) stages frequently follow the development stages, in addition to the QC performed towards the end before delivering the final product. Large amounts of effort goes into these QC stages of a process for attaining the desired quality of software at the end.

Large amount of effort is spent in a project on the QC process, and this QC effort is distributed among the different QC stages. For example the project effort data discussed in [8] shows that on an average $12.5\%$ of QC effort was spent in Code Review, $16\%$ in Unit testing and $28\%$ in System Testing (remainder in other QC tasks). Most traditional software reliability models focus on the last stages of testing. Hence these models only capture the effort of the last QC stage[12]. In this paper we discuss the issue of allocating effort to the different tasks in a QC process for achieving some quality goal.

For the quality of the final software we will use the commonly used measure of delivered defect density - the number of defects present in the final product normalized by the size of the product. One of the main objectives of a project is to achieve the desired quality goal with least amount of resources.

Using defects as the defining metric for quality, we can view the process of a project as comprising of defect injection and removal stages. There are some stages like the requirements, design and coding, in which defects are injected. These defects are removed in various QC stages. A quality objective can be met by reducing the defect injection by defect prevention [8]. A complimentary approach to achieve a quality goal is to remove the injected defects to the desired level through different QC stages. In this paper we focus on the latter approach.

A QC stage can be characterized by the defect removal rate of that stage. There can be many possible combinations of defect removal rates for the different QC stages that can achieve the same overall quality goal. The different combinations will have different implications on the total QC effort. Clearly, for a process designer or a project manager, a key problem is to select the amount of effort to be spent in each QC stage such that the desired quality goal is met with the minimum cost. In this paper we address this problem of allocating effort to different QC stages for a given total cost such that the overall quality is optimal.

We propose a model for the cost of QC process. For cost, we make the standard assumption that the main factor determining cost is the effort requirement. That is, cost and effort are same and we will use them interchangeably. The average effort for detecting a defect in a QC stage increases as the total number of defects remaining in the software decreases. The average effort of removing a defect also increases if there is a high latency between the injection and removal of that defect. We use non-linear increasing func-

tions to represent these efforts, and use them to derive expressions for the effort estimate of each QC stage, once the defect injection rates are known. The problem of allocating resources then reduces to a optimal resource allocation problem, which we solve using sequential quadratic programming.

We give examples in this paper to show how optimal allocation of effort to each QC stage can be done to achieve a goal with minimum total effort. We also discuss how the model parameters can be obtained from process performance data that is often collected by organizations. We have also built a software that, given the project parameters, gives the optimal resource allocation schedule for any given overall quality goal.

## 2 Process Model

A software development process consists of a series of development activities, executed in some order to build the desired software. Each development activity performs some transformations on the inputs passed to it and generates some outputs. During these transformations, some defects get injected. Hence these development activities form the defect injection points of the software development process.

As defects get injected into the system, the quality of software being developed goes down. To deliver software of good quality, these defects are removed by the different QC activities before the software is delivered. The QC activities include requirements review, design review, code review, unit testing, integration testing, system testing, and acceptance testing. This process of defect injection and removal is shown in Figure 1 [8]. The injected defects are either removed in the next QC stage or passed on to the subsequent stages. We assume that the number of defects injected in the QC stages can be ignored. A similar model for injection and removal is used in [15].

In general, let there be $m$ stages in the software process. Out of the total $m$ stages, let there be $n$ QC stages, numbered from 1 to $n$ according to the order in which they occur in the process. Let the defect injection rate at an injection stage $i$ be $I_i$ defects per unit size and defect removal efficiency at some removal stage $j$ be $r_j$, $i = 1, 2, \ldots (m - n)$ and $j = 1, 2, \ldots n$. The defect removal efficiency (DRE) of a QC stage is the fraction of defects present that are removed by the QC stage [10]. Let the estimated size of work product at stage $i$ be $S_i$ units, measured in number of pages, KLOC, or some other metric.

It is known that the effort for removing defects increases with latency [3]. That is, the average effort for removing an injected defect increases with the time it remains in the system. So, for example, requirement defects (defects injected during the requirement stage) may require an hour or two each to remove if caught in the requirement review, but take huge effort if they are detected during the final testing stages.

This increase in cost due to increasing latency, clearly suggests that it be may require lesser effort to have a QC stage detect all (or almost all) the defects that exists at that time. In other words, it suggests that the DRE of a QC stage should be as close as possible to $100\%$.

However, we also know that detecting first few defects is much easier than detecting defects when very few are left. In general, the effort required for detecting defects in a QC stage increases as the DRE increases, and may reach a very high level when the DRE reaches $100\%$. This is the basis of many software reliability models [6][7][14] – that cost of detecting a defect increases as the number of defects present in the system decreases. The model relating coverage and number of test cases in [15] also suggests this non-linear growth of effort as coverage and defects detected increase. We capture these two effort drivers in the following cost functions.

1. The average effort for removing a defect increases with the defect removal efficiency. We refer to this cost as $C_1(i, r_i)$ – the average effort for removing a single defect in stage $i$ at DRE of $r_i$.

2. The average effort for removing a defect also increases with its latency i.e. the delay between the injection and the removal of the defect. For a defect injected in stage $i$ and removed in stage $j$, the average effort for removal increases by a factor of $C_2(i, j)$.

Note that by having $C_1$ as a function of just the stage identity and the DRE, we are assuming that the average effort for removing a defect in a QC stage is a function of its DRE and changes as more defects are identified and removed. In other words, this form of $C_1$ implies that the average effort for identifying and removing a defect when few defect has been removed (i.e. the DRE is low) is different from the average effort for identifying and removing a defect when most defects have been identified by the QC activity and there are few defects left in the system (i.e. the DRE is high.) It should be clear that $C_1$ will increase with DRE. It is worth noticing that most reliability growth models make this assumption for the system testing stage when they are applicable, though they do not use the concept of DRE. The reliability growth models assume that the effort to identify and remove a defect increases as the number of defects that have been identified and removed increases. (For a survey of various reliability models, kindly refer to [11][17])

Note that these forms of $C_1$ and $C_2$ also assume that the average effort for removal of a defect, or the increase in removal cost due to latency, is same for different types of defects. The model can be easily generalized to have different cost functions for different types of defects, but then the formulation and solution will get that much more complex.

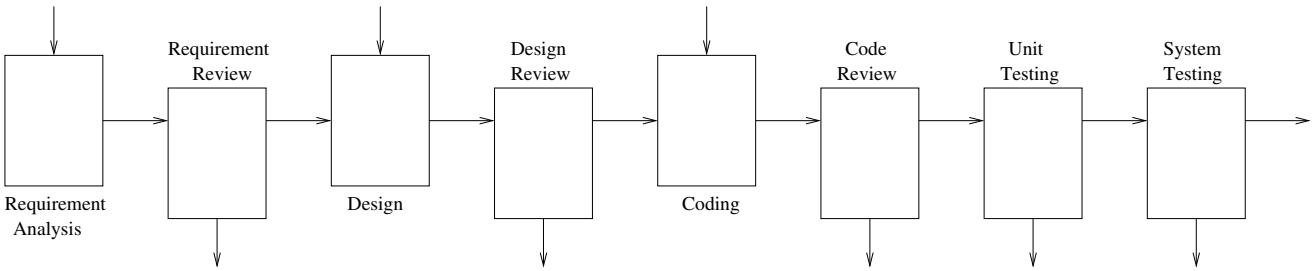


**Figure 1. Defect injection and removal**

Most reliability growth models also make this assumption of considering defects together.

The effort for removing defects in a QC stage clearly depends on the number of defects removed in that stage, which also depends upon the number of defects present at the beginning of the QC activity for that stage. The total number of defects present in the software at the start of the QC stage is the sum of the defects injected in the preceding defect injection stage, and the defects passed from earlier stages. Let $d_i$ be the number of defects injected in stage $i$, which is the product of injection rate and size, if the injection rate is specified in terms of per unit size. If the injection rate is specified in terms of per unit effort, as is sometimes done for practical reasons [8], then $d_i$ is the product of the injection rate and the estimated effort for this stage.

Let $D_i$ be the number of defects present at the beginning of the $i^{th}$ QC stage. The number of defects present in the first QC stage is $D_1 = d_1$. Out of $D_1$ defects $r_1 D_1$ gets removed at this stage and the remaining $D_1(1-r_1)$ defects pass onto the next QC stage. Thus, the number of defects in second QC stage is $D_2 = D_1(1-r_1) + d_2$. Similarly for the $3^{rd}$ stage, $D_3 = D_2(1-r_2) + d_3$ or $D_3 = d_1(1-r_1)(1-r_2) + d_2(1-r_2) + d_3$. By generalizing the formula for any QC stage $i$ in general, we get

$$D_i = d_i + \sum_{j=1}^{i-1} d_j(1-r_j)(1-r_{j+1})\cdots(1-r_{i-1})$$

Thus, given the defect injection estimates of the different injection stages and the defect removal efficiencies of the QC stages, we can compute the estimated number of defects in the final software, and hence the quality of the delivered software in terms of Defect Density. The number of defects present at the end of the last QC stage (stage $n$) is $D_n(1-r_n)$. If the size of the final software is $S$, the final defect density of the delivered software is $\frac{D_n(1-r_n)}{S}$.

It is worth pointing out that the defect injection rates and defect removal efficiencies of the different QC tasks can be determined for a process. Mature organization that collect metrics data often determine these parameters to characterize the capability of their process [9]. Examples of such characterization from a real organization can be found in [8]. In general for a particular project, the injection rates and removal efficiencies will be estimated based on process capability. Using past data to estimate for a project is commonly done in software organizations, and is a key requirement of CMM [16]. This approach rests on the foundation that for a stable process, the range of results that can be expected by using the process can be predicted from its past performance. (Formally this can be defined as the process being under statistical control [5][20]).

## 3 Allocating Resources to QC Tasks

Given the defect injection and removal model, let us now consider the problem of allocating resources, to the different defect removal stages. If the cost functions $C_1$ and $C_2$ for each of the QC stages are given, then the total effort required to achieve a $DRE$ in a stage can be determined. Note that, to determine the effort required for a QC stage, we not only need the function $C_1$, we also need the function $C_2$, so as to estimate the impact of defects left in the earlier stages on the total effort for defect removal in this stage. In other words, given some formulations for these cost functions we can construct a function that takes as input the $DRE$ of each of the QC stages and gives as output the effort required for each of those stages. Similarly, given the effort allocated for each of the QC stages, we can also compute the $DRE$ for those stages and hence the quality of the final software. Let us see how the the total effort can be determined using this.

### 3.1 Cost of the QC Process

One of the most commonly used function in cost estimation [3][1] is the polynomial function of the form $f(x) = ax^b$. Using this form for the cost function $C_1$, we propose

$$C_1(i, r_i) = a_i * r_i^{b_i}$$

It is a very general function that should be applicable in most situations. The function states that the average effort for removing a defect in a QC stage increases as the defect removal efficiency increases. The rate of increase depends on the constants $a_i$ and $b_i$ of the function. This formulation allows each QC stage to have a different function for capturing how the effort for removing defects increases with the defect removal efficiency. So, it is possible to say that for one QC stage the effort increases linearly with DRE (i.e. $b$ is 1.0) while in another QC stage the effort requirement quadruples when the DRE doubles (i.e. $b$ is 2.0). Note that though we are using a form of cost function used in project effort estimation models like the COCOMO [3], this function (and its constants) have no direct relationship with them.

The cost of defect removal also increases with latency. Figure 2[4], for example shows the relative cost of fixing a requirement defect at the various QC stages. This factor may be as large as 200 if detected an removed during the operation stage. One simple way of viewing this cost function is a constant scaling function i.e

$$C_2(i,j) = M_i^{j-i}$$

In other words, the relative effort required for removing a defect increases by a factor of $M - i$ for each stage it remains undetected. That is, the cost escalation per stage for detecting a defect injected in stage $i$ is fixed to be $M_i$. Hence, if a defect injected in stage $i$ is detected in stage $j$, the cost escalation will be by a factor given by $C_2$. Note again that the scaling factor can be different for different defect detection stages. We use $\vec{M}$ to represent the $M_i$s for different values of $i$. By having one scaling factor for each stage $i$, we are assuming that the average scaling factor for these defects is the same for each stage for the different stages $j$. Note that we can easily extend the function $C_2$ to have different scaling factors for different detection stages. But that will complicate the representation of $C_2$, and will add more parameters in the model.

The total effort of a QC stage is computed in the following way. In stage 1, total of $r_1 D_1$ are removed. The expected effort for removal the first defect is $a_1(1/D_1)^{b_1}$. The effort for the second defect is $a_1(2/D_1)^{b_1}$ and so on. Thus, the total effort expected to be incurred at stage 1, $K_1$ is $\sum_{j=1}^{r_1 D_1} a_1 \left(\frac{j}{D_1}\right)^{b_1}$. In the second QC stage defects left by the first QC stage as well as the new defects injected are present. Out of the total $D_2$ defects present at the beginning, $d_2$ are from stage 2, while $d_1(1-r_1)$ are the ones that remain from stage 1. The probability that a defect removed is from stage 1 is $\frac{d_1(1-r_1)}{D_2}$ and that it is injected in stage 2 is $\frac{d_1}{D_2}$. Thus the expected effort for removing one defect at a given $DRE$ is $\frac{M_1 d_1(1-r_1)+d_2}{D_2} a_2(DRE)^{b_2}$. So the total effort in the $2^{nd}$ QC stage, $K_2$ is, $\left(M_1 \times \frac{d_1(1-r_1)}{D_2} + \frac{d_2}{D_2}\right) \times$ $\sum_{j=1}^{r_2 D_2} a_2(j/D_2)^{b_2}$. For the $i^{th}$ QC stage in general the total effort $K_i$ is

$$K_i = \frac{1}{D_i} \left( \sum_{j=1}^{i} M_j^{(i-j)} d_j \prod_{k=j}^{i-1} (1 - r_k) \right) \left( \sum_{j=1}^{r_i D_i} a_i (j/D_i)^{b_i} \right)$$

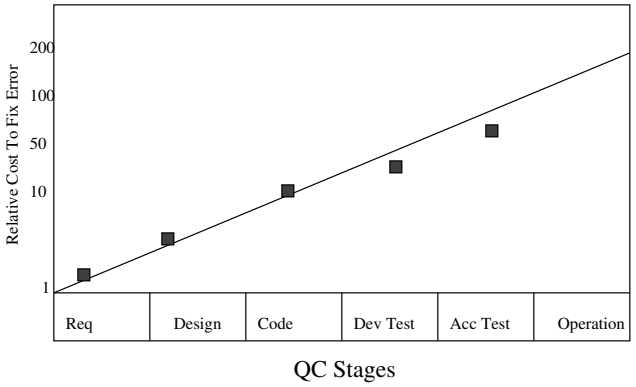


**Figure 2. Defect With Latency**

It is clear that the overall effort for removal of defects will depend on these two functions - $C_1$ and $C_2$. It should also be clear that there is a tradeoff involved - if we increase the DRE of a QC stage, we catch more defects and hence reduce the effort increase due to latency. However, we also increase the effort for this stage. Similarly, if we reduce the DRE of a stage the QC cost of this stage reduces but the cost due to latency increases. Clearly, by allocating resources judiciously, we can reduce our overall QC effort.

### 3.2 Optimally Allocating Resources

The resource allocation problem in its generic form has two variants. The first is to obtain best quality software for a given total effort. That is, the total effort to be spent on the entire QC process is given, and the goal is to allocate this effort among the different QC tasks in the process such that maximum number of defects are detected by the QC process. This situation occurs commonly when during project planning the overall resources for the project and the broad distribution of these resources to major activities or stages in the project are decided based on project properties and constraints.

The other formulation is to obtain the minimum effort required to meet the given quality goal. That is, given a quality goal (in terms of defect density delivered), what is the minimum effort required to achieve this goal, and how that effort should be distributed among the different QC tasks. It should be clear that these two problems are dual of each other. Here, we present the first variant of the problem.

Mathematically the first problem can be stated as, given the total effort for the QC process as $\bar{K}$, and the final size of software as $S$,

$$Minimize(\frac{D_n(1-r_n)}{S})$$

subject to the constraints

$$K_1 + K_2 + \cdots + K_n \leq \bar{K}$$

$$0 < r_i < 1.0 \text{ for } 1 \leq i \leq n$$

In other words given the total amount of resource we have ($\bar{K}$), we have to find the effort to be spent in the different stages (i.e. the $K_i$s) such that the overall quality is maximum (i.e. $D_n$ is minimum.) In this formulation, $K_1, K_2, \cdots, K_n$ are as defined above and can be determined for given $r_i$s, using $C_1$ and $C_2$ for the different stages. In other words, the goal of this formulation is to determine the $r_i$s, and then the $K_i$s for these $r_i$s. The final value of $K_i$s will give the optimum distribution of the total QC effort, which is fixed at $K$.

Observe that the cost functions $C_1$ and $C_2$ are nonlinear, so the overall cost of defect removal is a nonlinear function in terms of per stage defect removal efficiency, $DRE_i$. Therefore the optimization of this non-linear function cannot be treated as a linear programming problem. Thus the optimization problem we have to solve is a non-linear optimization with linear constraints.

To solve this formulation, the main inputs are the number of stages in the process, the cost functions $C_1$ and $C_2$ for each stage (i.e. the value of the different $a_i$, $b_i$, and $M_i$), the defect injection in different stages (or the defect injection rate and size estimate), the size $S$ of the final software, and the total QC cost K. In the alternate formulation, all input parameters are the same except that instead of total effort, the desired quality goal is specified.

## 4 An Example

To illustrate the approach, as an example, consider a process consisting of requirement specification, design specification and coding as the development sub-processes. The estimated mean values of defect injection rates, and sizes of the work products for these stages is shown in Table 1. Suppose that from the past data it is determined that the cost of removing requirement defects scale up by a factor of 1.6 for every phase that it remains latent. Similarly, the scaling factor for design and coding defects is known to be 2.9 and 2.8. Note that if the effort for defect removal is being logged and defects are being classified based on the origin, (as is frequently done), the scaling factors can be easily determined. The scaling factors for the defects injected in these stages are also shown in Table 1.

| Stage | Injection rate | Size | M |
|---|---|---|---|
| Requirements | 8 defects/page | 15 pages | 1.6 |
| Design | 12 defects/page | 30 pages | 2.9 |
| Coding | 40 defects/KLOC | 20 KLOC | 2.8 |

**Table 1. Defect injection phases**

Suppose that the QC process being used in the project consists of requirement review, design review, code review, unit testing, and system testing. Parameters $a$ and $b$ for these stages have to be estimated for defining the function $C_1$. Estimating these parameters from past data can be done using statistical techniques, if suitable data is available. These parameters can also be estimated if two vectors consisting of different $DREs$, etc are obtained from past data. In this approach, the past data is stratified into two or more groups using the final DRE obtained (or the final quality.) Then for each group, the DREs for the different QC stages (i.e. the vector of DREs) are obtained. Using these vectors, simultaneous equations can be constructed, which can then be used to determine the estimates for $a_i$s and $b_i$s.

These parameters can also be approximated if the project manager gives an estimate of average defect removal cost at two or more $DREs$ for each stage. In this approach, the total cost for a DRE is computed as given earlier. If the average cost (and hence the total cost) for different DREs is given for a QC stage, then for each DRE we can obtain an equation. To simplify the equation for a particular QC stage, we can assume that the DRE of the earlier stages is the average DRE for which the data is given. In this manner, we can get a set of simultaneous equations, one equation for each DRE for which the average cost is given. These simultaneous equations can then be solved to get the approximations for $a$ and $b$. Further details on these approaches is given in [19].

For the data given in Table 2, the average cost at different DRE is computed using the equations above in terms of a and b. This is then equated to the actual average cost values given in the table to get the simultaneous equations. Solving these equations gives the values of $(a, b)$ as $(23.8, 2.1)$, $(31.0, 2.7)$, $(12.2, 3.2)$, $(36.7, 3.8)$,$(53.0, 4.0)$, for the five QC stages listed in the table.

Now we have the complete specification of the functions and we are ready for determining an optimum allocation. In order to see the benefit of optimization, consider a project manager who sets the DRE for the different stages as given in Table 3. These DREs are quite reasonable and what a project manager might choose - fairly high in the earlier stages, but a heavy reliance on system testing for removing defects. For the injection rates given in Table 1, the effort for each QC stage can be computed as discussed in Section 3. This effort is also shown in Table 3. For this resource

| QC activity | 50% | 70% | 90% |
|---|---|---|---|
| Req. review | 1.76 | 3.61 | 6.00 |
| Design review | 1.42 | 3.52 | 7.00 |
| Code review | 0.57 | 1.69 | 3.78 |
| Unit testing | 2.74 | 9.83 | 25.56 |
| System testing | 9.22 | 35.42 | 96.78 |

**Table 2. Average Effort for Different DRE**

allocation, the overall quality of the final software is $0.21$ defects per KLOC and the overall effort required is $6740.7$ person-hours.

Now we consider optimizing the resource allocation for this project. We set our overall quality goal to be $0.21$ defects per KLOC (which is obtained by the resource allocation the project manager has done) and use the optimization technique described earlier to determine the resource allocation among different QC stages. The optimum resource allocation schedule turns out to be as given in Table 4. For this resource allocation schedule the overall effort required is just $4183.6$ person hours. In other words, the resource allocation selected by the project manager (as shown in Table 3) is almost $60\%$ higher than optimum, even though it results in the same quality! It is worth noting that the optimum allocation suggests that code review consume the highest amount of effort, while system testing is not allocated too much resources. This is in contrast to the original effort distribution which was heavily in favor of system testing and had far lesser effort on code reviews. Note also that the optimum distribution depends on the cost functions - clearly if the cost function for code review is such that cost of detecting defects is higher, the optimum allocation will show lesser effort for code review.

| QC activity | DRE in % | Effort (person-hours) |
|---|---|---|
| Req. review | 80.0 | 487.4 |
| Design review | 74.5 | 951.1 |
| Code review | 77.0 | 1160.0 |
| Unit testing | 80.0 | 1680.1 |
| System testing | 90.0 | 2462.0 |

**Table 3. Quality control activities**

Now suppose the project manager (who does not have this optimization technique available) is told that he has only 4183.6 person-hours available for the QC activities. To get the best results from this, the project manager studies the literature and finds that catching defects as early as possible is considered a "best practice" and is suggested by many consultants. He decides to follow this technique of catching defects early by "front loading" the process. Suppose that he redistributes this effort as follows : Requirement Review - 800 person hours, Design Review - 800 person hours, Code Review - 800 person hours, Unit Testing - 800 person hours and System Testing - 983.6 person hours (for a total of 4183.6 person hours). This distribution is again quite intuitive for someone who believes that good requirement and design review will give him maximum benefits. Using this effort distribution, the delivered defect density can be computed as discussed earlier. For this resource allocation, it turns out that the overall quality of the project falls to 1.7 defects per KLOC - which is 8.3 times the defect density achieved with an optimal resource allocation. This further shows that even recommended practices may not necessarily work - the effectiveness of practices depends on the nature of QC processes and how the cost varies with the different factors.

| QC activity | DRE in % | Effort (person-hours) |
|---|---|---|
| Req. review | 44.0 | 81.4 |
| Design review | 62.5 | 554.3 |
| Code review | 96.4 | 3321.3 |
| Unit testing | 70.5 | 155.8 |
| System testing | 58.15 | 70.8 |

**Table 4. Optimum distribution of effort**

This example clearly illustrates the substantial impact of a sub-optimal resource allocation. If the resources are not judiciously allocated, either the cost rises considerably for achieving the same quality, or the end quality suffers.

## 5 A Web-Service for Optimal Resource Allocation

It is clear that software support will be needed to implement the approach to determine the best resource allocation. We have built a software for this purpose. In this software, we have the provision of solving both the versions of optimal-resource allocation problem. That is, using this software, one can solve the problem of minimizing the cost given a quality goal, as well as the problem of maximizing the defects detected for a given QC cost.

The software supports two different functions for $C_1$ and $C_2$. Apart from the polynomial form for $C_1$ discussed above, the user can select $C_1$ to be an exponential function of the form $C_1(i, DRE) = a_i(e^{b_i DRE} - 1)$. For $C_2$, the second function is of the form $C_2(i, j) = M_i \times (j - i)$. Which form to select depends on which one the user believes represent his process the best, and also what type of data is available to estimate the parameters. We believe

that in the start, the polynomial form that we have discussed above is sufficient for experimentation.

To estimate the parameters of $C_1$, the user needs to provide the average cost of defect removal for different DREs for each QC stage. For $C_2$, the user has to provide the scaling factors for each stage. With these inputs, the functions can be determined by the software. To give the user a feel for the cost functions, the plot of the cost functions is shown to the user.

The defect injection data is provided in terms of the total number of defects that are estimated to be injected for each stage. By using the expected number of defects injected, we are essentially bypassing the need for size as an input.

For optimization, if the final quality is fixed and cost is to be minimized, then the user has to specify the required quality, $e$. When the quality is to be maximized for a given cost, then the total planned effort for the QC process has to be supplied as input. With these inputs, the software formulates resource allocation as a nonlinear optimization problem, solves it, and then returns the optimum allocation of effort to different QC stages.

Standard methods exists in the literature to solve such problems, like dynamic programming, multivariate search methods, multi-criterion optimization etc. We have used the Matlab[18] implementation of Sequential Quadratic Programming(SQP) to get the optimal solution. The input to the algorithm is the cost function and the constraint matrix in terms of variables: $(r_1, \ldots, r_n)$. The output is a vector $(r_1^*, r_2^*, \cdots, r_n^*)$ for which the the overall cost is minimum and all the constraints are satisfied. Having obtained the per stage defect removal efficiency - $DRE_i$, the cost of each QC stage can be computed using the equations derived in Section 2. Details about SQP algorithm can be found in [2]. An experimental version of the website is available through www.cse.iitk.ac.in/users/jalote/SDA.html.

## 6 Conclusion

Deciding the allocation of effort to different activities is one of the most important tasks during project planning. Though there are mature models available for estimating the overall effort for a project, few models are available for distributing this effort among different tasks. In this paper we have addressed the issue of building a model for distributing the total effort for the QC process among the different QC tasks.

We have used defect density as the metric for measuring the quality of software at any stage and have used the defect injection and removal stages as the basic model. We assume that the cost of defect removal increases with the latency between injection and removal and also with the defect removal efficiency. Using these two assumptions we have modeled the problem as an optimal resource allocation problem. We have also developed the software to determine the optimal resource allocation among different QC stage.

There are clearly many issues that need further exploring. The current model seems to be too complicated, requiring a lot of data for use. Clearly, one area that needs to be explored is to design cost functions that closely represent reality but require lesser information about the process. One possibility is to work with functions that deal with average cost at a DRE - parameters for such a function may be easier to determine with lesser data. In general, there is a need to explore different cost models.

Work also needs to be done to relate the different constants used in the model to the software development process, as has been done for many reliability models in [13]. We believe that this initial effort can lead to models for helping resource allocation among different stages in a process and to study the tradeoffs involved.

Finally, there is a considerable work needed to apply and validate the models, once simpler and more intuitive models become available. Validation of such models, however, is tricky as for any situation the correct optimal allocation against which the results of a model can be validated are not likely to be known. Hence, though empirical validation of the two cost functions is possible, any validation of the overall resource allocation model will probably involve some subjective evaluation, perhaps by experienced professionals and people in the field.

## References


[1] V. R. Basili. *Tutorial on models and metrics for software management and engineering*. IEEE Press, 1980.

[2] M. C. Biggs. Constrained minimization using recursive quadratic programming. In *Towards Global Optimization*, pages 341–349. North-Holland, 1975.

[3] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

[4] B. Boehm. *Software Risk Management*. IEEE Press, 1989.

[5] W. A. Florac and A. D. Carleton. *Measuring the Software Process: Statistical Process Control for Software Process Improvement*. Addison-Wesley, 1999.

[6] A. L. Goel. Software reliability models: Assumptions, limitations, and applicability. *IEEE Transactions on Software Engineering*, SE-11(12):1411–1423, Dec. 1985.

[7] D. Houston and J. B. Keats. Cost of software quality: A means of promoting software process improvement. *Quality Engineering*, 10(3):563–573, 1998.

[8] P. Jalote. *CMM in Practice: Processes for Executing Software Projects at Infosys*. Addison-Wesley, 2000.

[9] P. Jalote. Use of metrics in high maturity organizations. *Software Quality Professional*, 4(2), 2002. Also available at www.cse.iitk.ac.in/users/jalote.

[10] S. H. Kan. *Metrics and Models in Software Quality Engineering*. Addison-Wesley, 1995.

[11] M. R. Liu. *Handbook of Software Reliability Engineering*. McGraw Hill, 1996.

[12] M. R. Lyu, S. Rangarajan, and A. P. van Moorsel. Optimal allocation of test resources for software reliability growth model in software development. *IEEE Transactions on Reliability*, 51(2), June 2002.

[13] Y. K. Malaiya and J. Denton. What do the software reliability growth model parameters represent. In *Proceedings of the Eighth International Symposium on Software Reliability Engineering (ISSRE '97)*, pages 124–135, Albuquerque, NM, USA, Nov. 1997.

[14] J. D. Musa, A. Iannino, and K. Okumoto. *Software reliability—measurement, prediction, application*. McGraw Hill, 1987.

[15] P. Piwowarski, M. Ohba, and J. Caruso. Coverage measurement experience during function test. In *Proceedings of the 15th International Conference on Software Engineering (ICSE)*, Baltimore, MA, USA, May 1993.

[16] Software Engineering Institute. *The Capability Maturity Model for Software: Guidelines for Improving the Software Process*. Addison Wesley, 1995.

[17] T. Thayer and M. E. Lipow. *Software Reliability*. North Holland, 1978.

[18] The Mathworks Inc. http://www.mathworks.com/product/matlab.

[19] B. Vishal and P. Jalote. Allocating resources to quality control tasks in a software project. Technical report, Department of Computer Sc. and Engg., I. I. T. Kanpur, India, 2003.

[20] D. J. Wheeler and D. S. Chambers. *Understanding Statistical Process Control, 2nd edition*. SPC Press, Knoxville, TN, USA, 1992.