Chapter 6

Decision Making Framework For Project Management Using A Value Based Software Engineering Approach

6.1 Introduction

Our discussion in the previous two chapters has been aimed at using statistical and analytical techniques for risk management in software engineering. In this chapter we focus our attention on the modeling aspect of software development process, with a special emphasis on estimating schedule, total cost, and total manpower or labor requirement. We analyze the impacts of corrective actions and decision making on the quality of the software product. Such actions or decisions can be aimed at changing the "values, productivity, and learning ability" of the software development team or the effectiveness of CASE tools deployed during the development process.

The estimation models to study the impacts on the quality of the software product should be as realistic as possible to real life situations. Productivity, although defined differently by software engineers, is considered an important attribute in a software development project. Norden [NOR60] defines a development project as a finite sequence of purposeful, temporally ordered activities, operating on a homogenous set of problem elements, to meet a specified set of objectives representing an increment of a technological advance." If we consider each new software system as a technological advance, then the people or team members play a major role in this advancement.

In a cost model proposed in [MIZ93b], cost is directly proportional to productivity and project engineering software system. Productivity can be based on a number of factors such as the effectiveness of software development CASE tools, change in the

stability of specification and the frequency of such change, design configuration stability, and quality requirements. Productivity, as defined by Mizuno in [MIZ93b], can be improved by the software management team who can select factors to improve productivity. Productivity as defined in [HAM86] can be viewed as a function of future concerns such as remaining tasks to be captured or the and remaining project time.

In this chapter we propose two models that combine the flexibility of model proposed by Mizuno in [MIZ93b] and a more realistic time varying productivity phenomena. In our models we consider the following cases of learning or productivity in a software development project:

- a rapid learning/productivity environment
- an average learning/productivity environment

The objective of these new models is not only to accurately analyze the man power and cost entities but also help us in improving the decision making process by providing corrective alternatives. In order to facilitate this process we need to develop a powerful estimation mechanism that is as realistic as possible for the life cycle of a project, accurately predicts the maximum manpower required, and the time at which the maximum manpower requirement occurs. By proposing such a technique, we can find answers to some critical management issues such as:

- How to meet the completion deadline by choosing appropriate team members.
- Whether or not the current team satisfies the desired time requirements or not at any time in the project life.
- How to make corrective actions at any point of the project due to changes in time requirements or quality of the project.

- How to predict the cost and quality based on the present and past data.

Corrective actions may imply replacing old team members by more advanced/skilled personnel, or bringing few expert software developers into the process. The proposed models are not only appreciable to the productivity aspects of the development team, they can also be used to study the effect of evolving CASE technology on the development process of the software project. They can be used for changing cost assignments to control the quality of the project.

We use our analysis for three major software metrics, namely, design stability, fault profiles, and requirement stability data by treating them as quality indicators.

6.2 Background

In a study about predicting software productivity [HUM91], it is indicated that companies measure software development progress by the lines of code developed by programmers. The study suggests that coding is only a part of many activities that a software development engineer has to perform such as attending meetings, taking courses, and calling out detailed planning. An experimental approach is instead proposed. Based on pre-existing productivity data recorded about a specific programmer or team, predictions can be made about the same programmer or team. It has been noted in [LON87] that during the life cycle of a project, the productivity of team members increases progressively. Such productivity can be the result of formal experience or increased learning capability. Human productivity as a result is the increasing capability to solve invisibility, conformity and complexity problems. All these definitions of productivity can be transformed into the notion of quality, since all such factors effect the ultimate quality of the product.

There have been different practical and theoretical paradigms to developing software systems. To minimize delivery time, some companies use software analyzers during the coding phase. The purpose is to detect interface errors which contribute to about 75% of all errors in a system. These errors are found in the testing phase of traditional systems. Furthermore, in some software environments additional 10% errors are detected by test cases which are automatically generated by special software testing tools. By automatically detecting up to 85% of all the errors in a software system, delivery time can be reduced. In this regard, the integrated framework proposed in Chapter 5 with many CASE tools can help in increasing productivity and reducing errors as well and hence improving the quality.

As mentioned above the estimation models must be as realistic as possible to real life situation. An *estimation model* is defined as a scheme that predicts computer software data required by project planning steps, using empirically derived formulae. Due to limited number of samples that provide empirical data, no estimation model can be used for all types of software and development environments. Therefore, estimation models must be used cautiously. These models are categorized into two main categories: static and dynamic. Static models take a unique variable, such as size, as a starting point and use that to calculate other variables such as the cost. In dynamic models, on the other hand, multiple inter-dependent variables are used and notion of basic variable is not valid. Two well-known models each fall into one of these two categories: The COCOMO, abbreviated for the Constructive Cost Model, is a static model [BOE81a]. On the other hand, Putnam/Norden is a dynamic model [PUT76,PUT78].

Estimation models can be further divided into two groups: *single-variable* and *multivariable*. Single-variable methods make use of a single basic variable to estimate other desired values. A typical equation that formulates this type of models is:

$C = aS^b$

where C is the cost and S is the size of the code, and a and b are constants.

Static, multivariable models are generally based on the same principle, except that the parameters depend on more than one variable such as methods used, user participation, memory constraints, etc. Among the numerous models in the literature which fall into this category, COCOMO is the most widely accepted model, and is elaborated in the next section.

COCOMO is a hierarchy of software estimation models [BOE78,BOE81a]. This hierarchy consists of three sub-models, illustrated below:

Basic Sub-Model: This is a static single-valued model that relates the effort (cost) of the software development to the size of the code.

Intermediate Sub-Model: This sub-model uses a set of ``cost drivers'' in addition to program size to compute the cost of the projects.

Advanced Sub-Model: This sub-model takes the effect of cost drivers on each development step of the development process into account.

The basic COCOMO model is used for a quick and efficient cost estimation in most of the small- to medium-sized software projects. Three modes of software development are considered in this model, namely: the organic mode, the embedded mode, and the semi-detached mode. The *organic mode* deals with simple projects in which a team of experienced programmers work in small group, called the *embedded*

mode. In embedded mode, a project is assumed to have tight hardware, software, and operational constraints. The *semi-detached mode* is an intermediate mode between the organic mode and embedded mode. The mathematical models for determining the manpower, cost, and the development time for the organic, semi-detached and embedded modes, respectively, are as follows:

$C_o = 2.4\ S^{1.05}, \quad t_d = 2.5\ C_o^{0.38}$

$C_s = 3.0\ S^{1.12}, \quad t_d = 2.5\ C_s^{0.35}$

$C_e = 3.6\ S^{1.20}, \quad t_d = 2.5\ C_e^{0.32}$

These formula provide an estimate for the approximate development time in terms of the cost and the projected program size. We use the relationship between the completion time and cost to determine the performance profile of projects for different cost levels. The same is done for varying manpower levels (team sizes), using a dynamic estimation model, which is discussed next.

6.3 The Putnam/Norden Model

The objective of this model is to progressively reduce the number of problems (tasks) or faults in a project at a constant rate and hence model productivity as a linear learning curve [PUT76,PUT78]. The following assumptions are made in this model:

1. The number of problems to be solved is unknown a prior but finite.
2. The problem-solving effort does make an impact on the unsolved problem set.

3. A decision removes one unsolved problem from the set.

4. The staff size is proportional to the number of problems seeking solution.

The above assumptions lead to a Rayleigh distribution of manpower over the course of project development. This is due to the assumption that the number of problems to be solved is unknown and finite, and the total manpower effort, K, is an indicator of the number of problems. The rate of variation for the cumulative manpower cost, $\frac{dC}{dt}$, signifies the number of people involved in the project, $m(t)$. The cumulative cost $C(t)$ can then be expressed as:

$$C(t) = \int_0^t m(\tau)d\tau \qquad (6.1)$$

According to the fourth assumption given above, the number of people involved is proportional to the effort remaining to be employed. Therefore,

$$\frac{dC(t)}{dt} = p(t)[K - C(t)]$$

where K is the total manpower effort and $p(t)$ is the proportionality factor, which is a function of time, and takes the effect of learning/productivity into account. By integrating this equation, one obtains:

$$C(t) = K[1 - \exp(-\int_0^t p(\tau)d\tau)] \qquad (6.2)$$

Another assumption that is made in this model is that the most representative learning curve is linear. This linearity is expressed as:

$$p(t) = 2at$$

where a is a positive constant. By carrying out the integration in Equation 6.2, one obtains the expression of the cumulative manpower cost as:

$$C(t) = K[1 - \exp(-at^2)] \tag{6.3}$$

The manning of the project can easily be obtained by differentiating the above equation:

$$m(t) = 2Kat\exp(-at^2) \tag{6.4}$$

which represents a Rayleigh distribution. $m(t)$ is zero at the beginning of the project, has a single peak at a time value which is a function of the constant a, and then asymptotically decreases towards zero. By deriving the function $m(t)$ and finding its zero point, i.e. the peak time, we can find the so called, development time, which is given below:

$$t_d^2 = \frac{1}{2a} \tag{6.5}$$

By substituting the values of t_d in the cumulative cost equation (6.3), the cost at t_d can be found in terms of the total effort spent, K:

$$C(t_d) = K(1 - \frac{1}{\sqrt{e}}) = 0.39K \tag{6.6}$$

This representation, which has frequently been verified in industry on large-scale project developments involving high technology has an important implication. It allows us to carry out the *scalability* of the software development profile. According to the Putnam/Norden model, states that the completion of intermediate software problems must be proportional to the completion of the entire project. For example, if a project is to be completed in half the time, each step must be completed in half the time. This is due to

the fact that the time variable t appears in the cost equation in a simple form and can be easily proven.

COCOMO has its limitations as it does not allow factors to be introduced other than those given in the model. This model considers variations other than those caused by the specified factors of this model to be errors [MIZ93b]. An extension to COCOMO has been proposed in [MIZ93b]. The extension allows users to select factors according to actual data. A data set of software metrics collected from 71 software development projects has been used to derive the cost model. Statistical techniques such as regression analysis and principal component analysis form the basis of this extension. The fact that users can select factors is advantageous, although the representation means of these factors are limited. Team ability, for example, is represented as a set of four values (-2, -1, +1, and +2) where -2 represents the lowest ability level while +2 is for the highest level.

In order to provide a more comprehensive model we need a representation that is time changing or temporal. The reason is that in reality, changing team sizes, mobility of developers, and changing CASE technology have significant impact on the development of a software product and the time-changing parameter must be captured in a succinct manner. Although the Putnam/Norden model deals with a time varying productivity or learning capability, its modeling of human productivity doesn't represent real life situations. For example according to this model during the life cycle of a project, team members acquire more knowledge about the problems of the project. Productivity, or capability to solve problems, continues to increase indefinitely albeit in progressively decreasing increments. There are two reasons that invalidate the linear productivity rate:

- The learning process of human beings about a specific problem tend to saturate.
- Problems in a software development project are finite.

The level of full-time-equivalent software personnel active on a project tends to follow a continuous curve where the instantaneous full-time commitment of a large number of people is an unlikely event [BOE81a]. As mentioned above, the model suggested in [BOE81a] indicates that the labor or manpower distribution for a number of software projects can be approximated by the Rayleigh distribution. From this distribution various important parameters can be obtained including the development time of a project, the peak manpower required, and the project finishing time. The development time, as mentioned in [BOE81a], is the time at which peak manpower occurs. It is the delivery time before which effort is spent on specification, design, coding, testing, and qualification [BOE81a]. Manpower requirements drop after the delivery time since most of the effort from that point on is spent on maintenance and modifications. At the delivery time, the product is considered operational [PUT76,PUT78]. It has frequently been verified in industry on large-scale project developments involving high technology that 39% of a software development project cost is consumed by the delivery time.

6.4 The Proposed Models for Metric-Based Decision/Quality Tradeoffs

As mentioned earlier, we propose two new models for estimating the impact of team productivity and, in general, the impact of the software environment such as the use of CASE technology on the product cost and project quality. Specifically, we illustrate the effects of varying team size and costs on the progress of software projects. We use the

following three software metrics for the measurement of progress. The following metrics identify the quality of the product:

- Design Stability.
- Fault Profiles
- Requirement Stability

6.4.1 The Rapid Learning Model:

In the rapid learning model, productivity is assumed to grow exponentially. In other words, the model emphasizes the fact that during the life cycle of a project, team members' capability to solve problems increases until a final saturation level is reached. When this final level is reached, team's productivity is maintained. Following are the reasons for choosing such an exponentially saturating productivity model:

1. The saturation of the learning process indicates that the team cannot continue to acquire knowledge about the project indefinitely.
2. Since productivity represents the ability to solve problems, and because problems are resolved with the passage of time and thus decrease in number; increase in productivity also decreases with time.
3. When team members have an extensive experience in software development, they may require a small duration to learn about the project. In other words, they start with a high impact on the productivity during the development process. This impact decreases as the project phase approaches towards completion.

A similar impact on the software development process can be extended to arguments about the use of CASE technology. In other words, the rapid learning model represents a nonlinear learning model whose learning rate decreases with time. The linear

learning rate decreases until it becomes negligible that marks the saturation level. A feasible curve that describes such an environment of productivity, would be an exponential curve, $p(t)$, defined as:

$$p(t) = a - b\exp(-ct) \tag{6.7}$$

where a, b, and c are positive constants. It can be noticed that in this case we have three parameters to deal with in contrast to the linear learning curve, where only one parameter controls the goal function, namely the cumulative manpower costs with respect to time. The parameters a and b in our model can be set to be equal to each other in case we need to model the situation where we assume that the team at the beginning of the project has no knowledge about the work to be accomplished. Parameter c indicates the productivity characteristics of the team. The larger the value of c is, the faster is the learning curve.

Using the productivity characteristics model of the team, following the same procedure as adopted in the Putnam/Norden model, we can find cost, manpower, and development time. The four assumptions made in the Putnam/Norden model still hold. The representation of cumulative manpower cost, $C(t)$ expressed in man years, in terms of manpower and the learning rate is given in Equation (6.1).

It is worthwhile to mention that the cumulative manpower cost effort, $C(t)$, is null at the beginning of the project and grows towards the total effort, K. Using the new model (Equation 6.7) with Equation (6.1) and carrying out the integration we find the cumulative manpower cost as follows:

$$C(t) = K\left[1 - \exp\left(-at - \frac{a}{c}\exp(-ct) + \frac{a}{c}\right)\right] \tag{6.8}$$

In this case the manning of the project would be:

$$m(t) = Ka\left(1-\exp(-ct)\right)\times\exp\left(-at-\frac{a}{c}\exp(-ct)+\frac{a}{c}\right) \quad (6.9)$$

As mentioned earlier, *m(t)* is zero at the beginning of the project (when $t = 0$), and it has a single peak at the time, whose value depends on the constants *a* and *c*. *m(t)* then asymptotically decreases towards zero. Deriving *m(t)* and finding its zero point, we get a quadratic equation of the form:

$$a^2\exp(-2ct)-(2a^2+ac)\times\exp(-ct)+a^2=0$$

The solutions to this equation, each representing the peak time which is referred to as the development time, are obtained as:

$$t_d=\left(\frac{1}{c}\right)\times\ln\left(\frac{2a}{2a+c\pm\sqrt{c^2+4ac}}\right) \quad (6.10)$$

By substituting these values of t_d into Equation (6.8), the cumulative cost in terms of the total effort spent, *K*, can be given as:

$$C(t_d)=K\left[1-\exp\left(-\frac{a}{c}\right)\times\ln\left(\frac{2a}{2a+c\pm\sqrt{c^2+4ac}}\right)-\left(\frac{2a+c\pm\sqrt{c^2+4ac}}{2c}\right)+\frac{a}{c}\right] \quad (6.11)$$

We need to set $C(t_d) = 0.39K$, in order to rationalize the industrially verified fact that for large-scale project developments, 39% of the cost occur sat the development time. In other words we must validate our cumulative manpower cost for whole, we can use iterative numerical analysis. The analysis is aimed at finding the list of all possible combinations of *a* and *c* that satisfy the Equation (6.11). In other words;

$$K\left[1-\exp\left(-\frac{a}{c}\right)\times\ln\left(\frac{2a}{2a+c\pm\sqrt{c^2+4ac}}\right)-\left(\frac{2a+c\pm\sqrt{c^2+4ac}}{2c}\right)+\frac{a}{c}\right]=0.39K \quad (6.12)$$

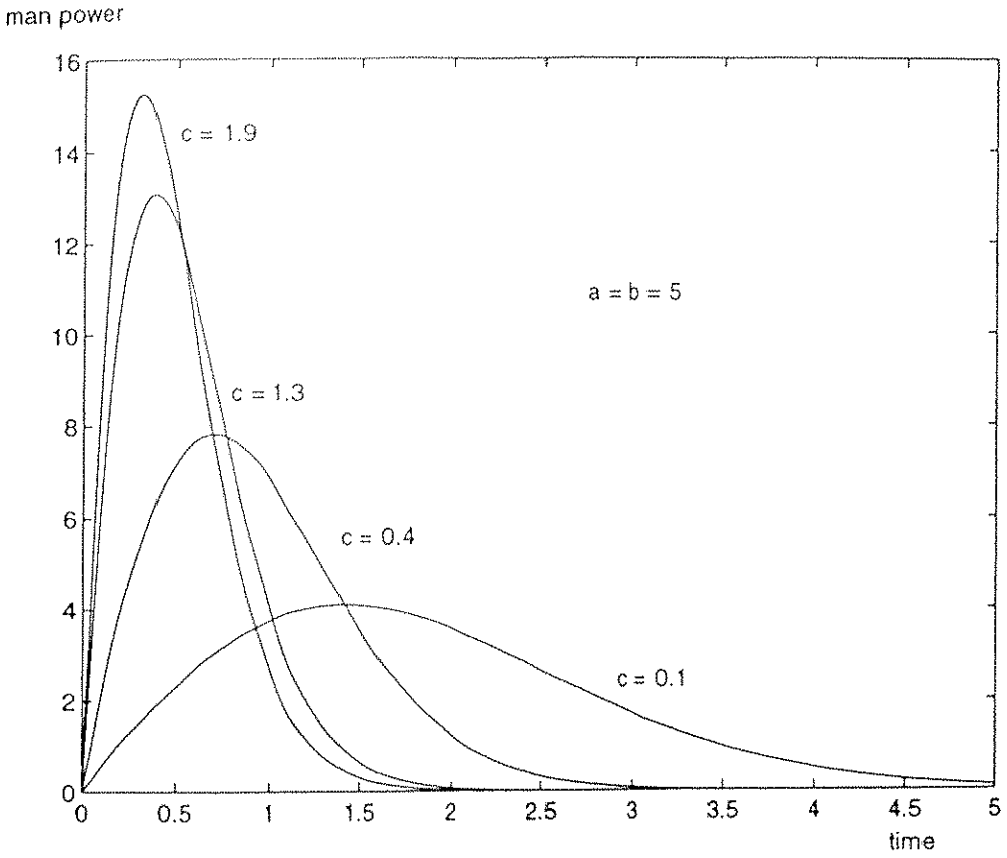

Figure (6.1): Rayleigh Distribution on manpower

The result of the numerical analysis provides a list of possible solution sets for parameter *a* and *c*. Figure (6.1) shows plots of manpower with respect to time as parameter *c* increases. It is important to remember that the total manpower effort, *K*, is assumed to be constant. It can be observed from this figure that *c* affects the overall distribution of manpower over time. It affects the development time as well as the peak manpower at development time. For example, for *c=0.1 (a=b=5)* the development time is as high as 1.5 years, and the maximum required team size is 4 persons. For the other extreme case, among the ones showed in the figure, when *c = 1.9 (a=b=5)* the development time is as low as 0.3 years and the maximum manpower required is 15

persons. Notice that the assumption that K is constant justifies the reason that the peak manpower increases with an increase in the learning rate *c*. This might not seem obvious, because we can think that the same peak team size , i.e.; constant peak manpower, can reduce the development time when the team members are more productive resulting in a large value of *c*. The assumption that *K* is constant requires a constant area under each manpower curve, which when changing *c* changes both the peak manpower and the development time. Figure (6.2) shows that changing *a*, while keeping *c* constant, has little impact on the development time and has a more obvious impact on the maximum manpower required.

The above discussion suggests that *c* is the dominating factor for controlling two important project parameters; development time and peak manpower. The cumulative manpower cost function then should be determined in terms of *c*. So far in our analysis, the new parameter, *K*, the total effort cost in man years, has been assumed to be a constant. In the following discussion we consider a more realistic representation of the total effort *K*. The total cost, as defined by Mizuno in [MIZ93b] and called NEC cost model expression, can depend on various factors such as program size etc. In [MIZ93b], it has been analyzed that the new parameter *Y*, in man-hour, can be given as

$$Y = C \times \prod_{i=1}^{p} 10^{\beta_i \times x_i} \times X^{\beta_i} \qquad (6.13),$$

where

x_i: Value of factor *i*

X: Program size in kilo lines (KL)

C, β_1, β_2: Constants.

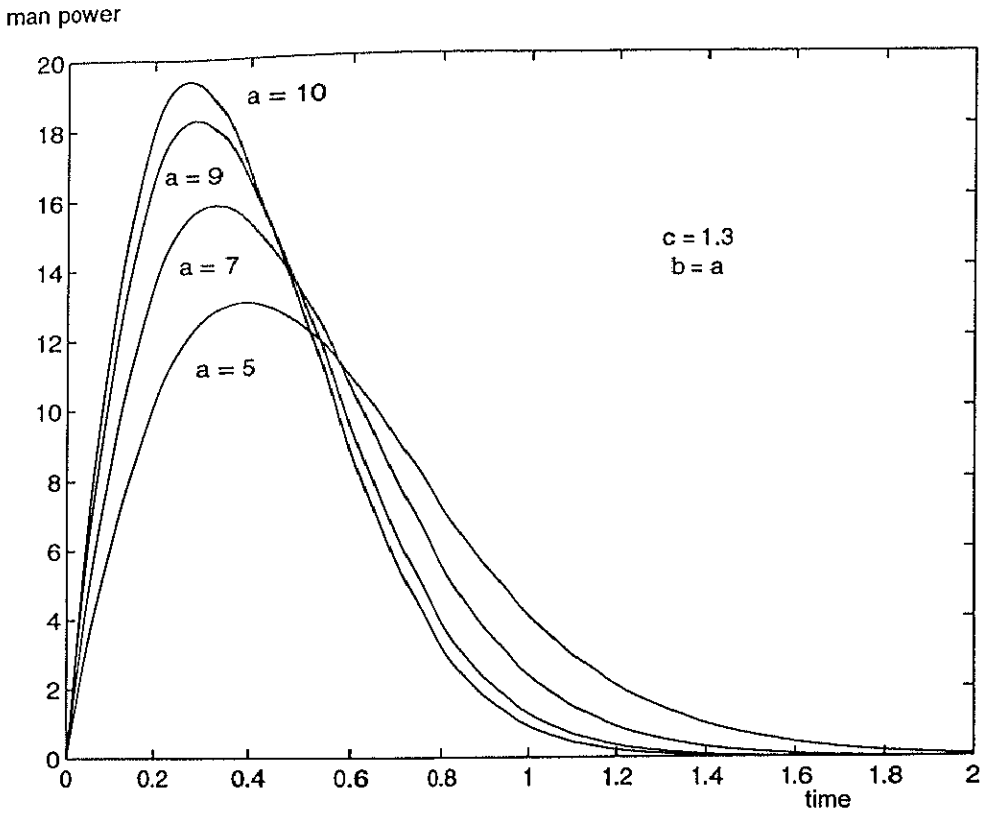

Figure (6.2): Manpower Distribution Over Time When Changing Parameter

For a set of 71 software development projects, the following cost in terms of the program size X has been reported as:

$$Y = 10^{2.418} \times 10^{0.123 \times 1} \times 10^{0.025 \times 2} \times X^{0.958} \text{ (in Man Hours)} \qquad (6.14)$$

The dominant factors in the above mentioned set of 71 projects include the development tools and development techniques/methodologies [MIZ93b]. If we substitute the total effort K by Y, we can then introduce a new dimension to our cost model. When replacing K, which is measured in man years, with Y, which is measured in man hours, we need to convert the unit of K from MH to MY (Man Years). Man month

consists of 152 hours of working time [BOE81a] which means that a man year is 1824 man hours. For the rapid learning model the extended cost expression is thus defined as:

$$C(t) = \frac{390}{1824} \times X^{0.958} \left[1 - \exp\left(-at - \frac{b}{c}\exp(-ct) + \frac{b}{c} \right) \right] \text{ MY} \tag{6.15}$$

As discussed before, we set $a = b$, and

$$C(t) = \frac{390}{1824} \times X^{0.958} \left[1 - \exp\left(-at - \frac{a}{c}\exp(-ct) + \frac{a}{c} \right) \right] \text{ MY}, \tag{6.16}$$

$$m(t) = a \times \left[\frac{390}{1824} \times X^{0.958} \right] \left[1 - \exp(-ct) \right] \exp\left[-at - \frac{a}{c}\exp(-ct) + \frac{a}{c} \right] \tag{6.17}$$

It is important to indicate that the development time is not changed, and remains as:

$$t_d = \frac{1}{c} \ln\left(\frac{2a}{2a + c \pm \sqrt{c^2 + 4ac}} \right) \tag{6.18}$$

Equations (6.16) and (6.17) provide analytical estimations for the cost and the manpower required, both as a function of program size and the effectiveness of software tool and team over a period of time.

6.4.2 The Effect of Rapid Productivity on the Quality

One of our research objectives is to determine how the change in various parameters such as cost, time, and productivity (team/CASE technique) can affect the quality of an ongoing development of a software product. For this purpose, we can use the four software metrics mentioned above, namely; requirement stability, fault profile, design stability, and reliability. As we collect metrics data, we can use this data for evaluation the quality of the project under development.

In this section we illustrate our approach using data from a NASA project [BAS85c]. The sub-metric used in our stability is the percentage of requirements met as we progress through the various development phases of the project. This sub-metric can directly determine the quality of the product, as argued in the previous chapter. Using this data from the design stability metrics, we have found the following relationship between design stability and cost:

$$Q = 2.4093C^3 - 3.4591C^2 + 2.2411C - 0.1192 \qquad (6.18)$$

where C is the cost that ranges between 0 and 1 of the final cost K, and Q is the design stability that ranges between 0 to 1 of the desired design requirement, S. For the cost function in Equation (6.18) we assume that *c has a value of 1.3*. Figure (6.3) shows the above relationship . If we consider the development time, in this case t = 1 year, we find that the cost, from the cost function, is $0.85594K$, and the quality, from the metric Q (Equation 6.18), is 0.85135. If we wish to achieve better quality, say 0.9, to occur at the same time t = 1 year, we need to allow larger cost consumption rate prior to that time. This implies changes in cost assignments and thus a new set of changes toward improved use of CASE tools and deployment of more expert personnel for the task. From the above equation, we can find C=$0.94K$. Since, cost is a function of time and c, and time is known (= 1 year) in this case, thus c could be found to be 1.95. This result is intuitively expected; if we want to achieve better quality, we must improve the effectiveness of our CASE tools and team personnel. Similarly, we can predict quality using sub-metrics for the other two metrics: requirements stability, the degree to which changes in the software requirements affect the development effort; and fault profile, the number of known faults fixed. Using

the NASA data [BAS85c], we have found the following relation between the requirements stability (R), and fault profile (F) in terms the cost, C.

$$R = 1000\left(4.5529C^3 - 6.1759C^2 + 2.6081C - 0.0305\right) \tag{6.19}$$

$$F = 1000\left(2.2699C^3 - 3.1848C^2 + 1.4565C - 0.0401\right) \tag{6.20}$$

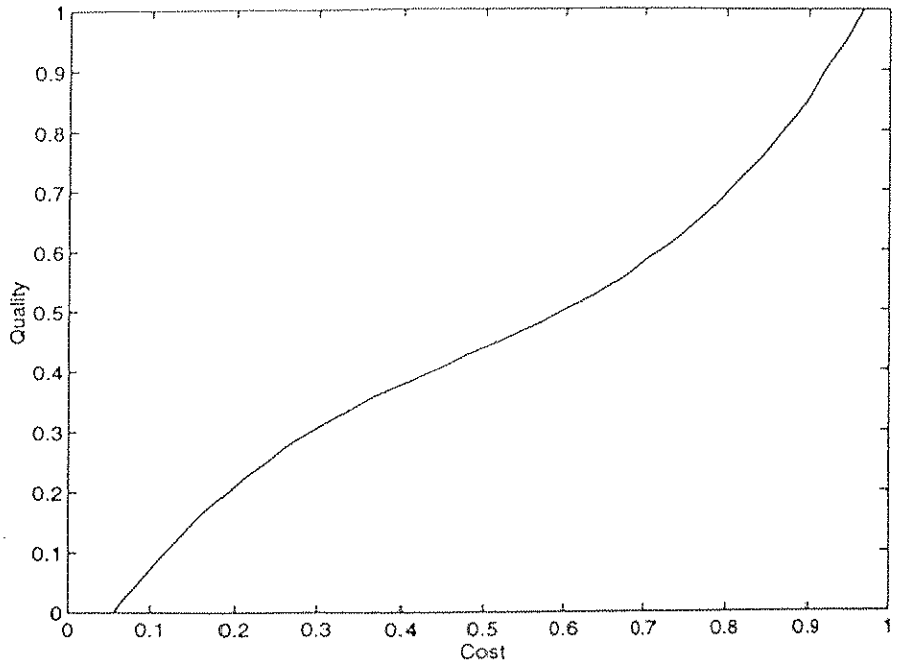

Figure (6.3): Quality as a Function of Cost

These relations have been plotted in Figures 6.4 and 6.5.

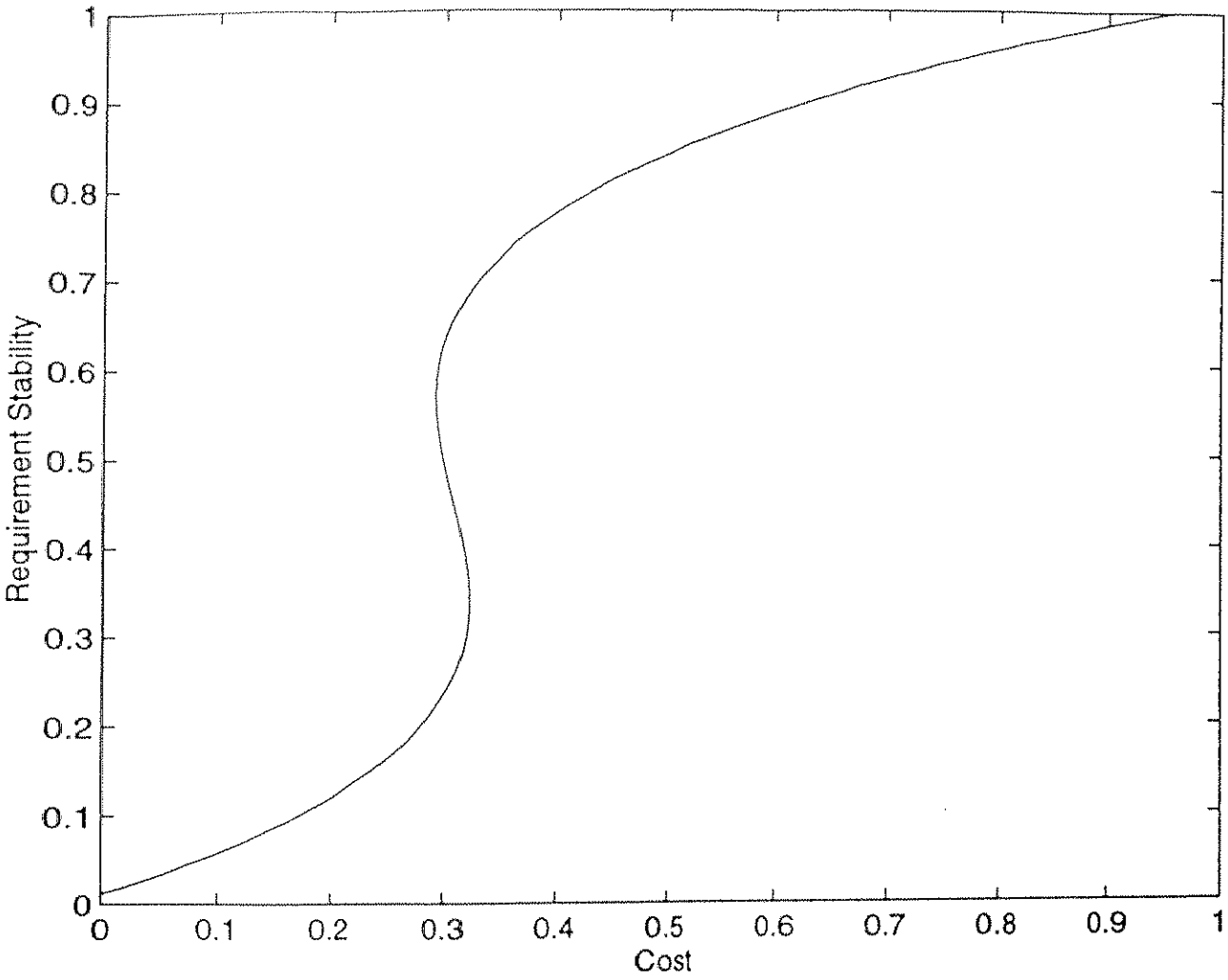

Figure (6.4): Requirement Stability as a Function of Cost

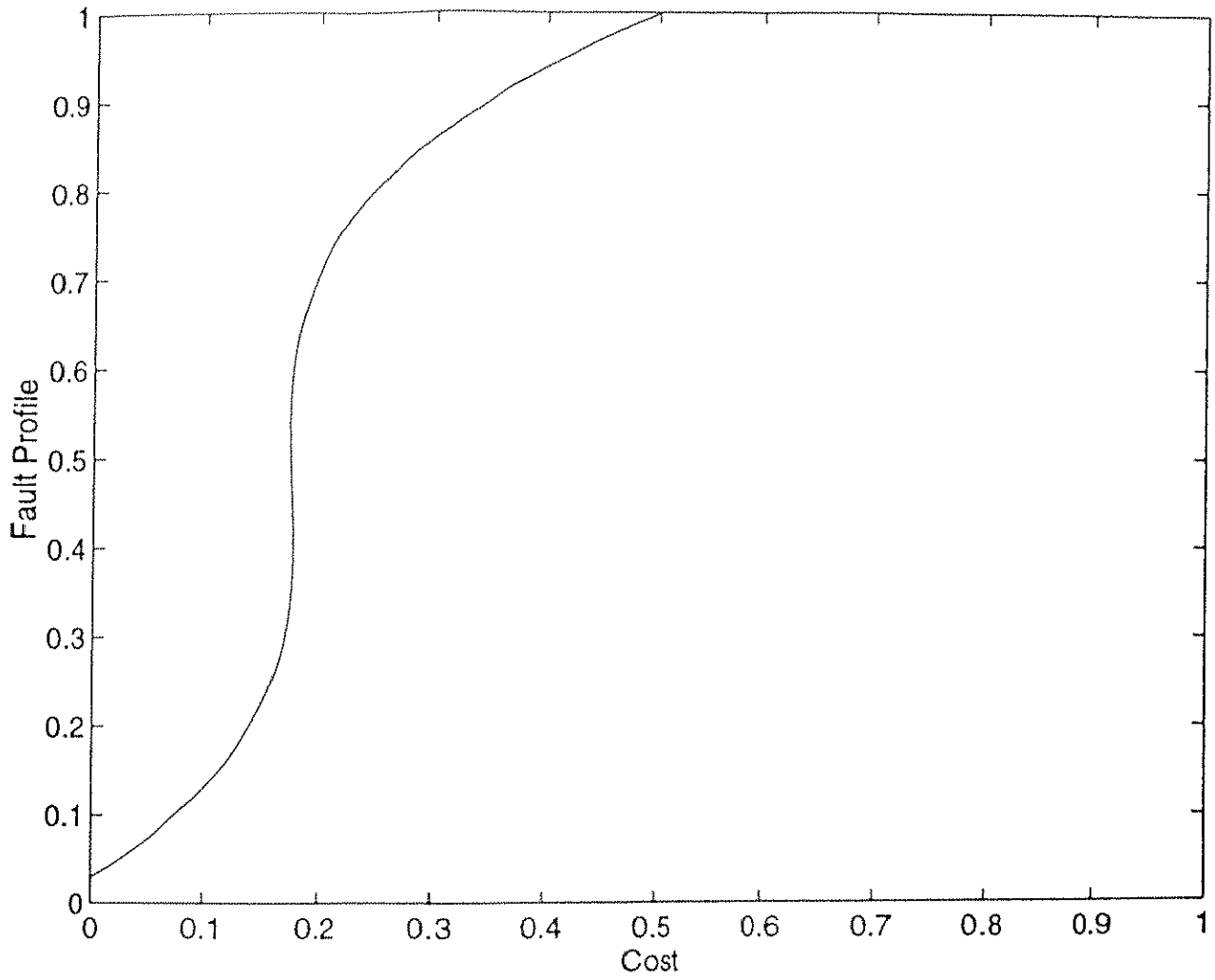

Figure (6.5): Fault Profile as a Function of Cost

6.4.3 Average Productivity Model

In this section we introduce the model to represent an environment that represents the average effectiveness of CASE tools and performance of personnel. In this model, the productivity is assumed to be considerably slow in the beginning of the development process. After a slow start-up, the team members become more productive and their effectiveness starts following the rapid productivity curve discussed in the previous section. Such a model is quite common because it represents the following two phases in the performance of a software development team:

1. During the first phase, team members try to get familiar with the overall requirements and try to achieve proficiency in using the CASE tools. However, the progress of the overall process can be slow. In other words, once a software engineer is given the definition of the overall project, he/she has to understand all the aspects of the given project and gradually solve the problems associated with the project.
2. In the second phase, the team members become more effective in solving the problems and using the CASE tools.

The following function can be chosen to represent such phenomenon mathematically:

$$p(t) = a - \frac{b}{\exp(ct) + \exp(-ct)} \qquad (6.21)$$

where a, b, and c are positive constants. This function can approximate both the two phases mentioned above. As mentioned earlier, the team size at the beginning of the project should be zero. This requires:

$p(0) = 0 \Rightarrow b = 2a$ which reduces Equation (6.21) to:

$$p(t) = a - \frac{2a}{\exp(ct) + \exp(-ct)}$$

Here a is the final productivity level that an individual might attain. This parameter is used to represent the concept of saturation in effectiveness. c represents how slow the team members are in the beginning of the project and how effective they become later. Larger values of c represent short duration for slow phase of productivity and quality shifting the productivity at a faster pace. Figure (6.6) shows different plots for $p(t)$ as c increase.

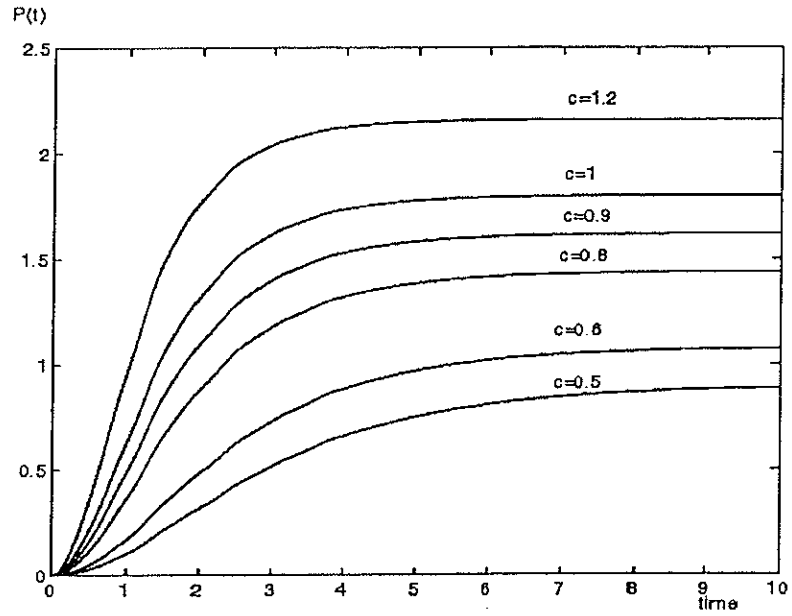

Figure (6.6): Average Productivity Model Curves

The same steps are followed as in the previous section for obtaining *C(t)* and *m(t)*.

$$C(t) = K\left[1 - \exp(-at + \frac{2a}{c}\left(\tan^{-1}(\exp(ct)) - 0.7854\right)\right] \tag{6.22}$$

$$m(t) = -K\left(-a + 2a \times \frac{\exp(ct)}{1+\exp(2ct)}\right) \times \exp\left(-at + \frac{2a}{c}\left(\tan^{-1}(\exp(ct)) - 0.7854\right)\right) \tag{6.23}$$

Finding the derivative of *m(t)* and setting it to zero in order to find the development time, we have the following equation:

$$a^2 \exp(4ct) + \left(-2ac - 4a^2\right)\exp(3ct) + \left(6a^2\right)\exp(2ct) + \left(2ac - 4a^2\right)\exp(ct) + a^2 = 0 \tag{6.24}$$

It is obvious from Equation (6.24) that it is quite difficult to find the development time in terms of a and c analytically. We also have to keep in mind that the condition $C(t_d) = 0.39K$ must be satisfied. Having these two purposes in mind, we need to find a relationship between parameters a and c to simplify the above polynomial. Randomly

choosing a value for a in the above polynomial, we found that for $a = 1.8c$, $C(t_d) = 0.3755K$ can be considered to be acceptable. Accordingly, the solution to the polynomial is as follows:

$$\exp(ct_d) = 3.6707 \Rightarrow t_d = \left(\frac{1}{c}\right)\ln(3.6707)$$

Thus, $$C(t) = K\left[1 - \exp(-1.8ct + 3.6\left(\tan^{-1}\left(\exp(ct)\right) - 0.7854\right)\right], \quad (6.25)$$

$$m(t) = -K\left(-1.8c + 3.8c \times \frac{\exp(ct)}{1+\exp(2ct)}\right) \times \exp\left(-1.8c + 3.6\left(\tan^{-1}\left(\exp(ct)\right) - 0.7854\right)\right) \quad (6.26)$$

The average rate of software team build up is defined as:

$$SBU = \frac{m_0}{t_d} = 1.077 \times K \times c^2 \times \exp(-1.8c + 1.87) \quad (6.27)$$

where m_0 is the peak manpower at delivery time and t_d is the development time.

Figure (6.7) shows the Rayleigh distribution of manpower with respect to time. The total effort K, as before, is assumed to be constant. Higher values of c, which correspond to high productivity rate, result in a smaller development times.

The cost and manpower expressions when using the NEC cost model are given as follows:

$$C(t) = \frac{390}{1824} X^{0.958}\left[1 - \exp(-1.8ct + 3.6 \times \left(\tan^{-1}(\exp(ct)) - 0.7854\right)\right], \quad (6.28)$$

$$m(t) = -\frac{390}{1824} X^{0.958}\left(-1.8c + 3.6c\frac{\exp(ct)}{1+\exp(2ct)}\right) \times \exp\left(-1.8c + 3.6\left(\tan^{-1}(\exp(ct)) - 0.7854\right)\right)$$

(6.29)

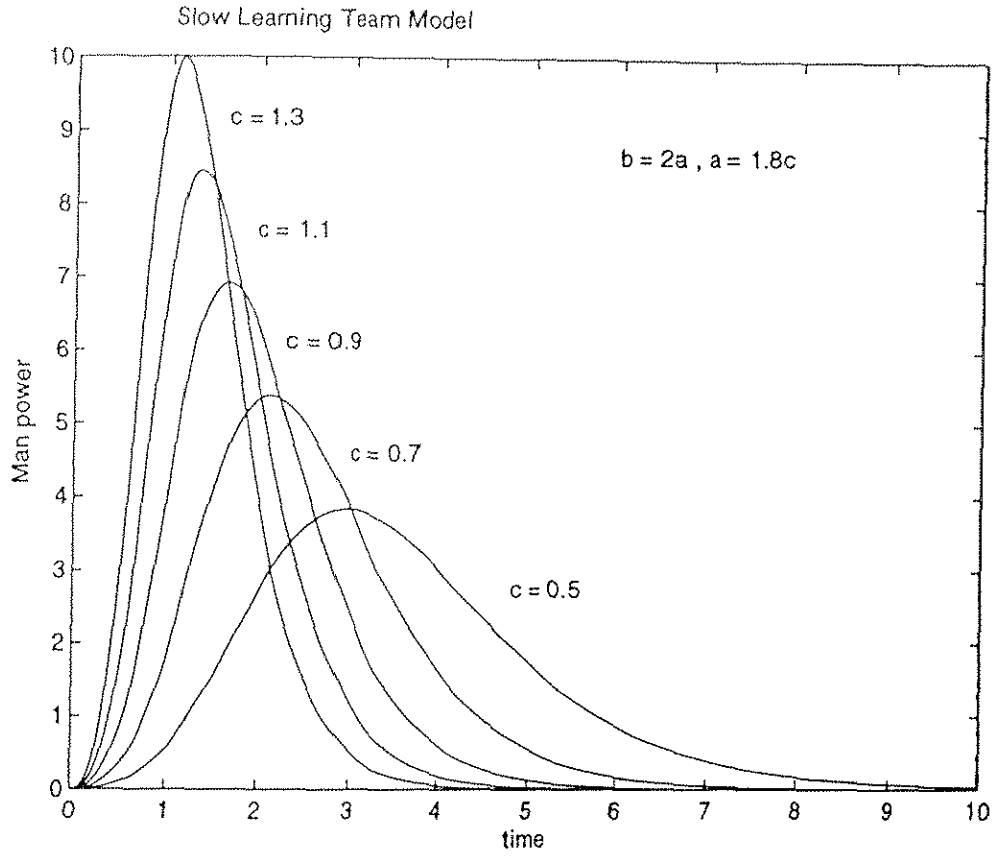

Figure (6.7): Rayleigh Distribution of Manpower based on the Average Productivity Model

Table 6-1 shows how factor c affects the finishing time (FT), the development time (DT), and the slow productivity time (ST) which is the period of time during which the team's productivity rate is slower. The percentage columns, following DT and ST columns, represent the percentage of development time and slow time with respect to finishing time, respectively.

We notice from Table 6-1 that percentage DT first increases and then decrease. A slightly different increase followed by some decrease in percentage ST can be noticed as well. For c = 1.7 notice that DT is 21% and ST is 13.3%. We consider this as the best representation of the average productivity model.

C	FT (yrs)	DT(yrs)	%DT	ST (yrs)	%ST
0.5	9	2.6	28	1.7	18.9
0.7	6.5	1.857	28	1.5	23
0.9	5	1.445	29	1.25	25
1.1	4.2	1.18	28	0.7	16.67
1.3	4	1	25	0.5	12.5
1.5	3.5	0.8669	24	0.45	12.8
1.7	3	0.7649	21	0.4	13.3
1.9	2.5	0.6844	27	0.35	14
2.1	2.3	06192	27	0.3	13.4

Table 6-1: A Profile of the Average Productivity Curve

6.4.4 The Effect of Average Productivity Environment on The Quality

We now find the relationship between the metrics mentioned earlier and cost. For the design stability metric, we find the following relationship between the design stability (Q) and cost C:

$$Q = 2.4428C^3 - 3.8706C^2 + 2.7698C + 0.0706 \qquad (6.30)$$

Equation (6.30) represents the relationship between cost and design stability when cost ranges between 0 and 1 of the final cost K, and Q ranges between 0 and 1 of the desired design requirement. Figure (6.8) shows the above relationship between quality and cost. This cost function assumes a value of 1.7 for c. If we consider the development time in the latter case to be $t_d = 9$ *months*, we find that the cost is $0.3614K$, and the

quality, from the metric, is 0.6317. If we wish to achieve a better quality, 0.8, at the same time $t_d = 9$ *months*, we need to allow larger cost consumption rate prior to that time. This indirectly implies a new team with high productive personnel. From the above equation, we can find C=0.555K. Since cost is a function of time and c and in this case time is known (=9 months), c can be found to be equal to 2.15. This result is intuitively expected; if we want to achieve better quality, we have to hire a more productive team or use a more effective CASE tool environment.

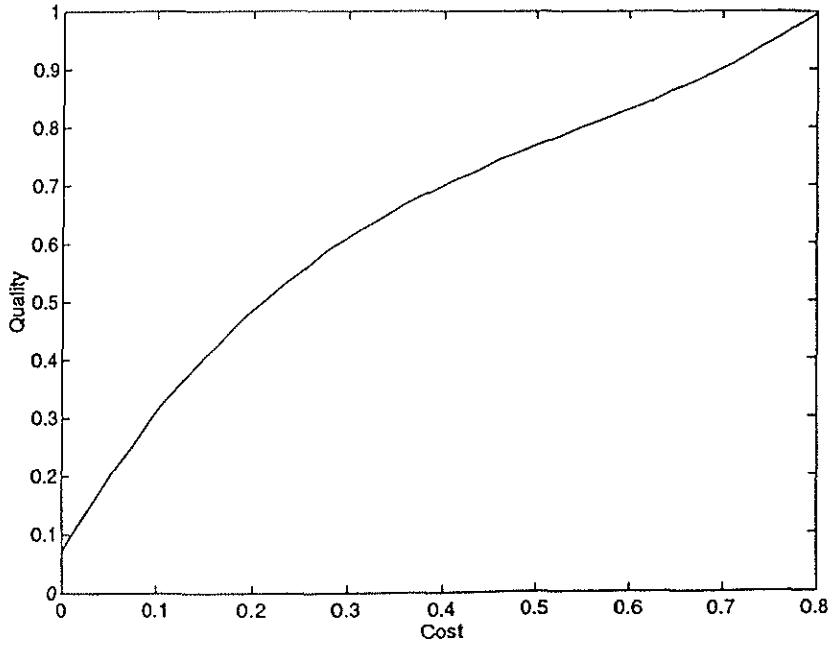

Figure (6.8): Quality as a Function of Cost

Similar predictions for the other two metrics namely: requirements stability and fault profile can be made. As a case study, we use the NASA data [BAS85c] and find the equations for R, requirements stability, and F, fault profile in terms the cost, C, as:

$$R = 1000\left(2.2065C^3 - 3.1434C^2 + 1.4738C + 0.0484\right) \tag{6.31}$$

$$F = 1000\left(3.1112C^3 - 4.1855C^2 + 2.039C + 0.1284\right) \tag{6.32}$$

These equations have been plotted in Figures 6.9 and 6.10.

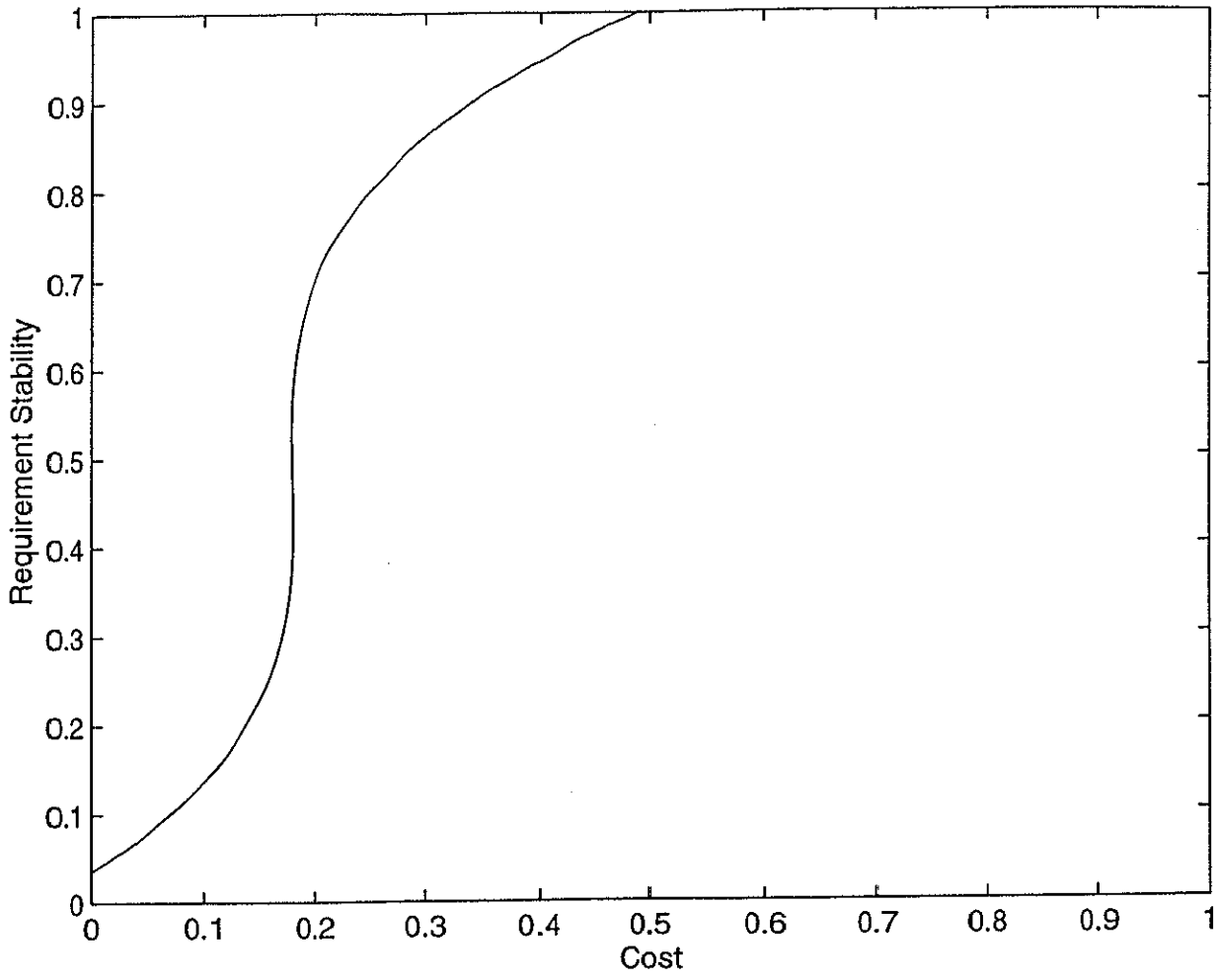

Figure (6.9): Requirement Stability as a Function of Cost

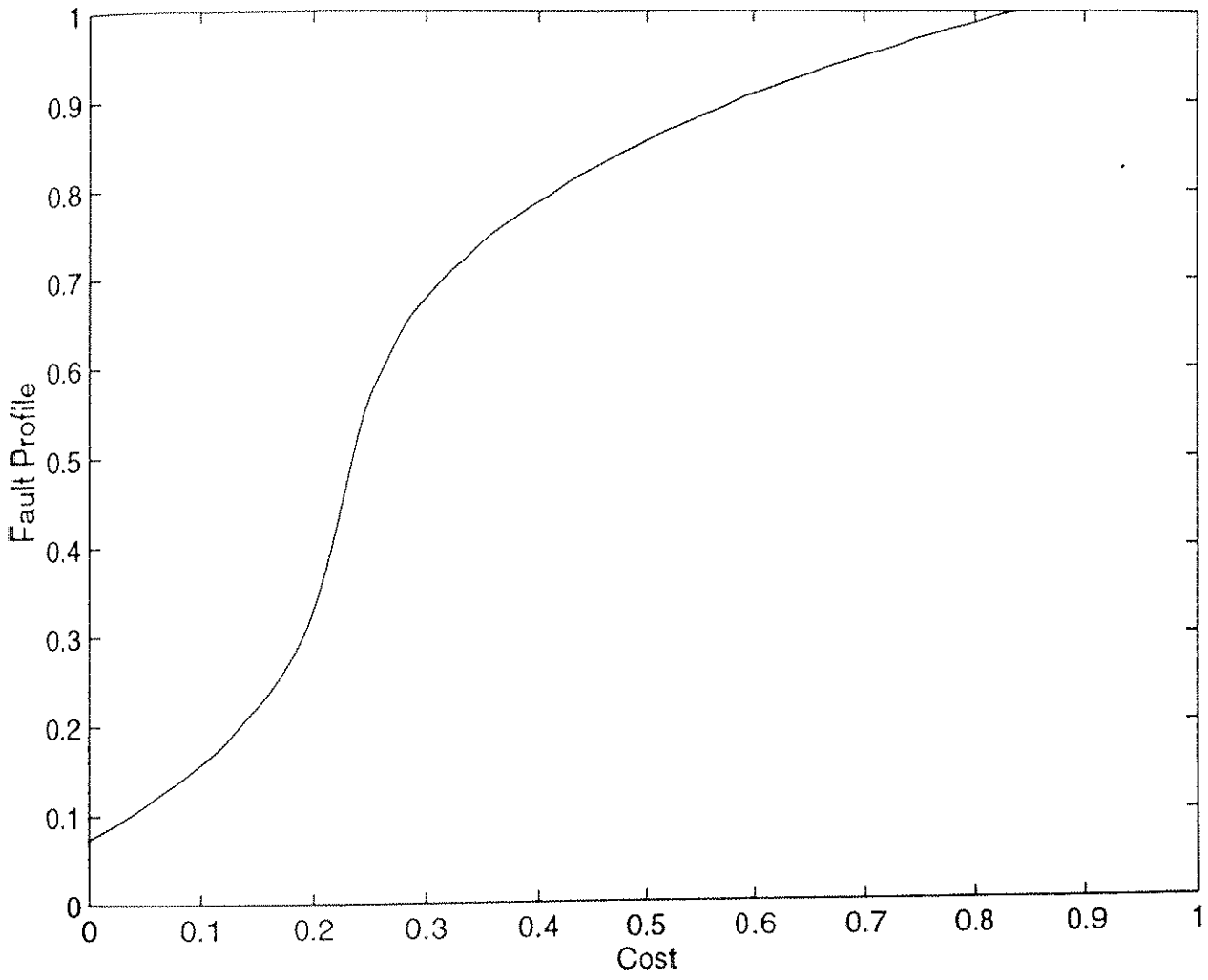

Figure (6.10): Fault Profile as a Function of Cost

6.5 Discussion

It is important to compare our two newly proposed models with the Putnam/Norden model. Figure (6.11) shows the rapid learning/productivity model with productivity rate c=1.3 and two linear models with productivity rates a=5 and a=1.5.

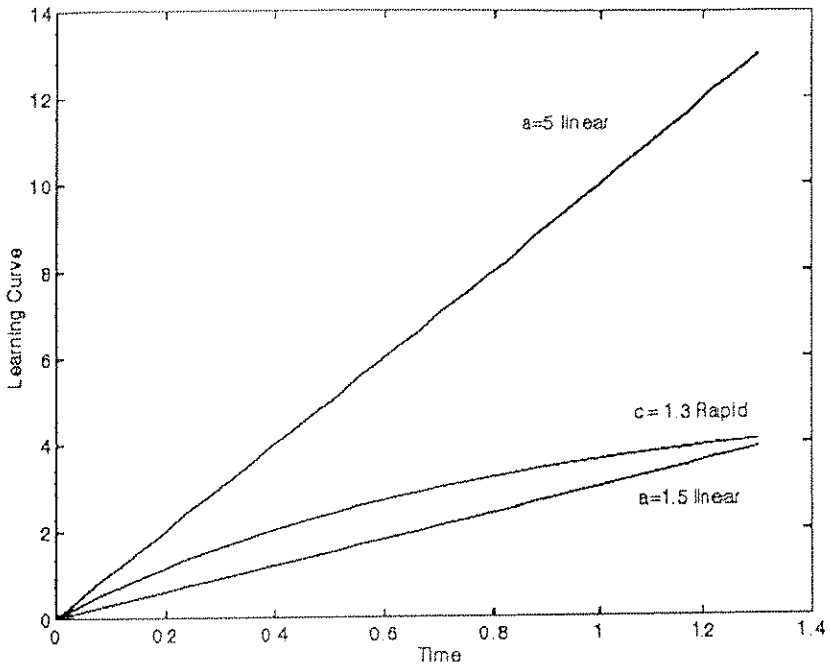

Figure (6.11): Rapid Learning/Productivity Curve in Comparison to Two Linear Learning/Productivity Curves

Figure (6.12) provides a similar comparison between the average learning/productivity model with rate c=1.7 and two linear learning/productivity models with same rates as mentioned above. The corresponding manpower distributions are given in Figures (6.13) and (6.14). It is known that the coding activity phase typically contributes to 42% of the overall project activities, and maintenance activities including rewriting parts of the code contribute to 48% of the project activities [LON87].

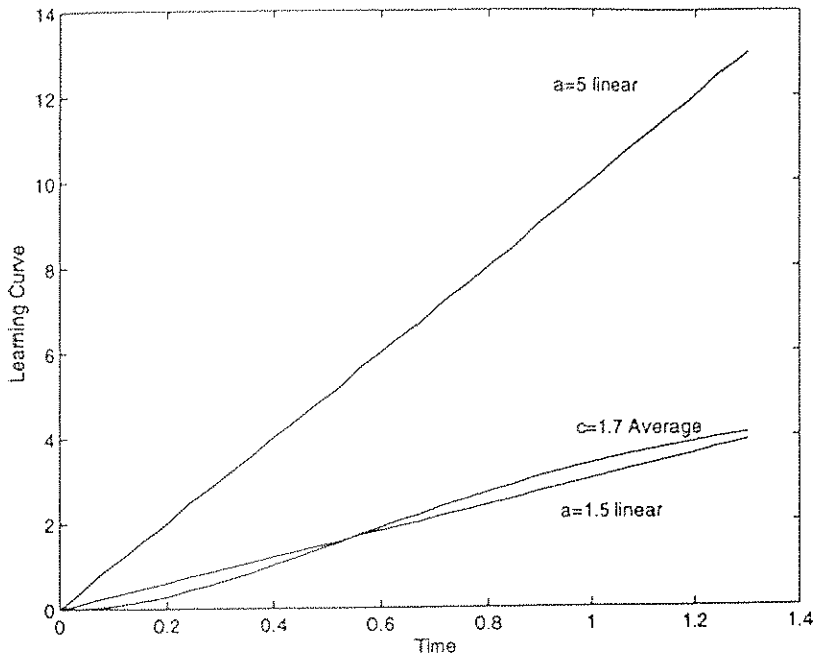

Figure (6.12): Average Learning Curve in Comparison to Two linear Learning Curves

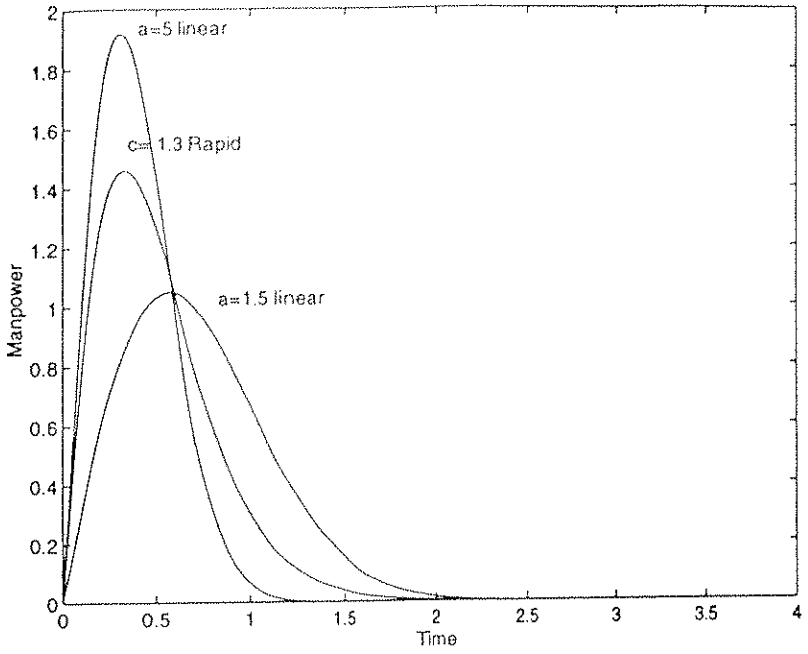

Figure (6.13): Manpower Distribution of Rapid Model in Comparison to Two Linear models

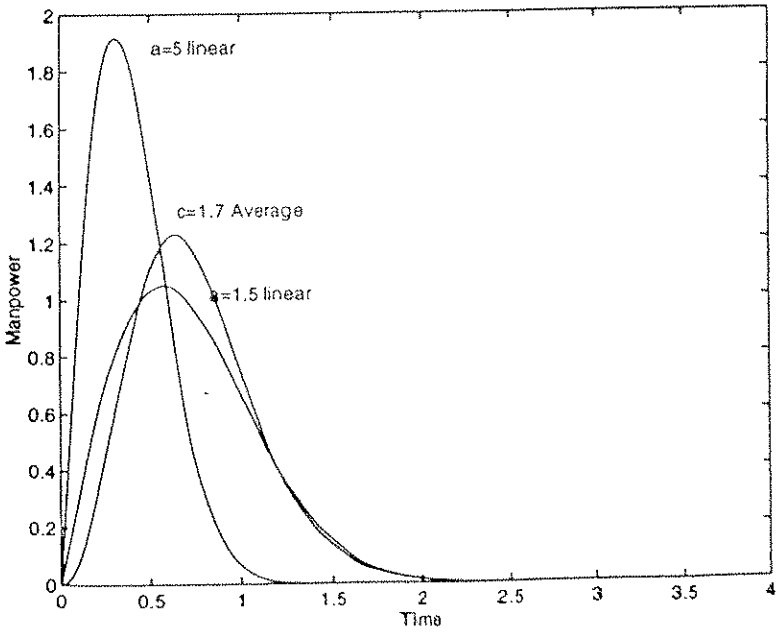

Figure (6.14): Manpower Distribution of Average model in Comparison to Two Linear models.

In other words, the development time should typically be at about 42% of the finishing time of the project which is observed in the rapid learning model. For a high linear productivity rate a=5, the manpower distribution tends to be symmetric with respect to development time which is not quite realistic for the reason given above. Additionally, peak manpower is significantly high which is not desirable due to communication and interaction problems among members of a large team. In Figure (6.14), on the other hand, we can see that the development and finishing times of the average model and the linear model with low productivity rate are close. For software development projects having project duration in the range of one to two years, the rapid productivity model represents a realistic representation of a highly qualified/productive team and an effective use of the CASE environment. For larger software development projects, the average productivity model is expected to behave better than the linear productivity model with low learning rates due to the fact that average team personnel improve their performance later in the project.

6.6 Experimental Validation of the Proposed Models

In this section, we analyze the relative merits of the Putnam-Norton's linear learning model and the newly proposed learning models, namely the average and the rapid models, via experimental best-fits on various sets of manpower data. A single model cannot be expected to fit better under all circumstances, because of the nature of generalized data fitting. For example, two sets of just 3 data points each {(0,1,2),(0,1,4)} fit better under $f(x)=x$ and $f(x)=x^2$, respectively. This is particularly true for the calculation of Root-Mean Square (RMS) value which is generally used for such fitting. Several experiments were performed to determine the relative efficacy of the two

proposed models (average and rapid learning) and the classical linear Putnam model. The analysis was performed predominantly in Mathematica. The objective was to determine the precise values of the parameters in manpower models so that the RMS value between a selected data set and the model was minimum. The equations for the manpower models used for this analysis are 6.4 (Putnam-Norton's linear learning model), 6.9 (the rapid learning model) and 6.26 (the average learning model).

Table 6.2a: NSDIR Manpower Data Set (Program month, Actual Manpower)

{0,0},{27,665},{29,1649},{30,1205},{60,2849},{88,3591},{90,3890},{91,770},
{121,4361},{122,272},{152,5327},{173,0},{176,2800},{180,5842},{182,1537},
{183,812},{202,252},{211,25395},{213,1947},{229,363},{244,2725},{247,30489},
{272,3322},{275,3126},{295,6500},{303,3973},{321,691},{333,4643},{334,4457},
{358,835},{364,5268},{387,53853},{394,5915},{395,6029},{419,1360},{425,6857},
{430,19665},{456,64655},{487,77060},{511,279763},{532,92290},{590,111531},
{665,43324},{681,129263},{682,139257},{701,139257},{713,147987},{738,147987},
{817,191813},{897,244215},{924,244215},{938,259490},{987,300921},{999,300921},
{1023,315627}

For the first experiment, a manpower data set from the National Software Data and Information Repository (NSDIR, URL:http://nsdir.cards.com/NSDIR) was used. This data set, as shown in Table 6.2a, provides the manpower distribution as a function of program months. For this set the analysis demonstrated that a set of parameters values can

be found for the average model (Equation 6.26) that provided the best-fit over the other two models in terms of the RMS value, and in fact 50% better than the linear model.

For the second experiment, a total of 11 manpower data sets were analyzed. These data sets were provided by the Nippon Electric Company (NEC), and are listed in Table 6.2b. Ten of these eleven sets exhibited better fits by the proposed average model (equation 6.26). For the 11 NEC data sets, the analysis and parameters values that result in the least RMS value are summarized in Table 6.3. One set (NEC-DS15) exhibited a better fit for the linear model (equation 6.4). In this case, the linear model outclassed the average model by a narrow margin of 0.04 percent. From a pragmatic point of view, the difference in the RMS values of these two models is not significant and the average model may be considered to have tied with the linear model even in this case.

Based on these data sets, overall, 75% of the cases were fit best by the proposed average model, 17% by the proposed rapid model, and the remaining 8% by the Putnam's linear model. In other words, the experimental results show that the proposed models perform better in about 92% of the cases, and in 8% of the cases, the proposed models perform almost as well as the classical Putnam linear model. These experiments demonstrate the superiority and the validity of these two proposed models (the average and the rapid) over the linear Putnam model.

The detailed analysis of NSDIR and NEC data sets is given in Appendix B.

Table 6.2b: NEC Data Sets (Actual Manpower)

Data Set	*Product Design*	*Detail Design*	*Programming*	*Testing*	*Conversion*
NEC-DS1	49181	108250	195735	156126	0
NEC-DS2	16507	42787	84383	55426	967
NEC-DS3	314	476	666	877	173
NEC-DS4	480	636	1000	500	145
NEC-DS5	213	263	885	182	60
NEC-DS6	1330	1730	2580	1350	300
NEC-DS7	833	1053	1587	784	0
NEC-DS8	30	30	40	35	221
NEC-DS9	6004	7674	18879	6022	5
NEC-DS10	172	234	352	164	1573
NEC-DS11	975	535	511	534	5330

Table 6.3: RMS values for the linear model and two proposed models, along with parameters which generated the best-fit for the manpower data sets.

Data Set	*Linear:*	*Average:*	*Rapid:*	*Optimal*
	RMS Value	*RMS Value*	*RMS Value*	*Model*
	K, a	K, c	K, c, a	
NSDIR	234655	157188	241595	Average
	1.1e6, 1e-5	1.18e12, 3.3e-5	1e-14, 3.5e-4, 1e-3	
DS1	110279	93726	141864	Average
	1.9e7, 1e-7	3e6, 5e-3	2e8, 9e-3, 1.4e-4	
DS2	45903	37840	38141	Average
	3e7, 1e-6	2e6, 5e-3	2.09e7, e-3, e-3	
DS8	408	389	461	Average
	3.07e7, e-5	3e7, 9e-6	3.18e5, .9e-2, 1.9e-2	
DS9	331	324	397	Average
	4e5, e-5	3.9e6, 1.6e-2	2.05e6, 8e-3, 8e-3	
DS10	525	488	646	Average
	5e6, 1e-4	2e5, 9.5e-5	2e7, 0.9, 1e-7	
DS11	836	851	523	Rapid
	5e5, 9e-6	3e6, 1e-2	3e6, 5e-4, 5e-4	
DS12	1673	627	1032	Average

Data Set	*Linear:*	*Average:*	*Rapid:*	*Optimal*
	RMS Value	*RMS Value*	*RMS Value*	*Model*
	K , a	*K , c*	*K , c, a*	
	1e8, 1e-9	2e6, 1e-2	5e5, 1.9, 2e-4	
DS13	122	95	166	Average
	4e7, 1e-7	1e8, 1e-4	5e7, 1, 2e-6	
DS14	10039	9344	13760	Average
	5e9, 9e-5	1.5e6, 6e-3	3e8, 1, 1e-7	
DS15	408	426	849	Linear
	1e9, 1e-4	1.6e6, 9.99e-2	2.85e6, 4.9e-4, 4.98e-4	
DS25	3339	2713	1242	Average
	4e9, 1e-9	2.9e12, 1.1e-5	2.7e6, 5.7e-4, 5.7e-4	

6.7 Conclusion

In summary, our main goal in this chapter was to represent accurately the human productivity factor in the development of software projects. In previous sections this factor historically was considered to be either indefinitely growing or was always constant. Our proposed models introduce a time changing human productivity that grows for some time and then saturates. It should also be noted that the final cost of a software development project can be given in terms of program size derived from the NEC model. Software development engineers using these proposed models can change quality requirements as needed. The price is the extra cost for paying more experienced labor. In this latter case new cost distribution and schedule can be determined a priori. The rapid

learning productivity model, as discussed above, is a more realistic representation of manpower distribution than the linear model with high learning rate; whereas the average learning model is comparable to the linear model with low learning rate in representing the manpower distribution.