

Next Generation Software Estimating Framework:

25 Years and Thousands of Projects Later

Michael A. Ross

President & CEO

r2Estimating, LLC

7755 E. Evening Glow Dr.

Scottsdale, AZ 85266-1295

480-488-8366

mike.ross@r2estimating.com

<http://www.r2estimating.com>

^{1,2,3}**Abstract**— It’s about time we in the software development community revisit the assumptions, relationships, and flexibility contained in our currently-available software estimating models. Most of the current models still implement fundamental relationships that are based on at least 25-years-old data and assumptions. In the meantime, data from many thousands of projects have since been collected and offer an opportunity to revisit old assumptions and relationships. This paper documents the basis, assumptions, and derivations behind a set of general software effort, duration, and defects estimating relationships that are based on the notion that software development is the cumulative effect of people laboring to do work (effort) over some duration (period of elapsed calendar time) that produces a desired software product (size or content) and unwanted byproducts (defects). This set of relationships is derived from several evidently-good correlations, the primary three being: 1) effort generally trends upward with increasing size, 2) duration generally trends upward with increasing effort, and 3) effort generally trends upward with increasing defects. This derivation ultimately yields three limited tradeoff relationships: one between effort and duration, one between cost and duration, and one between defects and duration.

TABLE OF CONTENTS

1. INTRODUCTION.....	2
2. OBSERVATIONS AND HYPOTHESES.....	5
3. DEFINING EFFORT, DURATION, AND SIZE.....	9
4. EFFORT-DURATION TRADEOFF RELATIONSHIP.....	10
5. DESCRIBING A PARTICULAR EFFORT-DURATION SOLUTION.....	14
6. MINIMUM DURATION.....	17
7. MINIMUM EFFORT.....	20
8. DEFECTS.....	23

¹ © 2008 r2Estimating®, LLC. All rights reserved.

² 2nd Edition, November 1, 2008

³ “Ross Chart™”, “r2 Software Estimating Framework™”, “r2SEF™”, and “r2Estimator™” are all trademarks of r2Estimating®, LLC. All rights reserved.

9. MODEL EMULATION	28
10. SUMMARY AND CONCLUSION	30
REFERENCES.....	30
BIOGRAPHY	32

1. INTRODUCTION

Purpose

The purpose of this paper is to document the basis, assumptions, and derivations behind a set of general software effort, duration, and defects estimating relationships that are based on the notion that software development is the cumulative effect of people laboring to do work (effort) over some duration (period of elapsed calendar time) that produces a desired software product (size or content) and undesired byproducts (defects).

Scope

The derivations, assumptions, and resulting framework described by this paper establish relationships between size, efficiency, defect vulnerability, effort, duration, and defects that are henceforth collectively referred to as the *r2 Software Estimating Framework (r2SEF)*TM and are implemented in the *r2ESTIMATOR*TM estimating tool⁴.

Background

Most of our current software estimating models still implement fundamental relationships that are based on at least 25 years old data and assumptions. In the meantime, data from many thousands of projects have since been collected and offer an opportunity to revisit old assumptions and relationships. To claim that this new data continues to “re-validate” these old assumptions and relationships is to claim that software development as a process is static and is to ignore its evolution with respect to management techniques, team behavior, host and target platforms, development methodologies, functionality abstraction, etc. Many of the parameters in these older models now seem anachronistic in light of current environments and practices. Additionally, analysis of this new data against these old relationships suggests that the underlying old assumptions are inappropriately restrictive. There are aspects of the old models that are held constant; the new data suggesting that these aspects should be variable according to the particular organization proposing to do the work and the type of project being proposed; *i.e.*, *one size does not necessarily fit all*.

Current Model Types

To facilitate discussion of software estimating models, we suggest the following categorization scheme which was inspired by Jensen⁵, introduced by Ross (2006), and updated here to accommodate the findings of this paper.

Type 0—Dart board, dice, roulette wheel, tarot cards, crystal ball, OuijaTM board.

⁴ *r2ESTIMATOR*TM is developed and distributed by *r2ESTIMATING*[®], LLC; <http://www.r2estimating.com>.

⁵ This categorization was inspired by and is very similar to that found in (Jensen, et al., 2006); the difference is this categorization’s emphasis on the existence of an effort-time tradeoff relationship as a type criterion.

Type 0.5—Engineering judgment (educated guessing).

Type 1—A univariate relationship that assumes total process effort E_p to be the sum of two components, one directly proportional to a linear function of effective software size S_e , where the constant of proportionality ϖ^{-1} represents average productivity⁶ (effective software size per unit of effort), and the optional offset c represents fixed (i.e., size-independent) effort.

$$E_p = \frac{S_e}{\varpi} + c \quad (1)$$

Type 2—A pair of univariate independent power estimating relationships: the first assumes total process effort E_p to be directly proportional to a power function of effective software size S_e and the second assumes total process duration t_p to be directly proportional to a power function of process effort E_p where the constants of proportionality χ_1 and χ_2 represent complexity scale factors⁷ as in, for example, COCOMO (Boehm, 1981), and its derivatives.

$$E_p \propto f_1(S_e) \text{ and } t_p \propto f_2(E_p) \quad (2)$$

where

$$f_1(x) = x^{a_1} \text{ and } f_2(x) = x^{a_2} \quad \therefore E_p = \chi_1 S_e^{a_1} \text{ and } t_p = \chi_2 E_p^{a_2} \quad (3)$$

The symbol \propto indicates that the two sides are proportional.

Type 3—A bivariate power relationship that assumes process average productivity $\varpi \equiv S_e/E_p$ is inversely proportional to a power function of the project's maximum staffing rate⁸ \dot{P}_{\max} , the maximum staffing rate assumed to be $\dot{P}_{\max} \equiv K/t_p^2$ where K is total life cycle effort and is assumed to be $E_p/0.3839$. Here the dot indicates differentiation with respect to time t . The constant of proportionality c_k represents an efficiency scale factor⁹ sometimes referred to as effective technology or productivity parameter as in, for example, Seer (Jensen, 1983A), SLIM (Putnam, 1980), and their derivatives.

⁶ Source of the ubiquitous lines per day and lines of code per person-month metrics.

⁷ COCOMO (Boehm, 1981), (Boehm, et al., 2000) uses a system of units that measures effective software size in source lines of code (SLOC), effort in person-months, and duration in calendar months.

⁸ This assumption is based on phenomenological observations made by Norden (1977) and elaborated for software development by Putnam (1980) that suggest both development and life cycle staffing follow the probability density function form of the Rayleigh distribution and that these shapes overlap in such a way that, as can be seen by the function's first derivative (rate function), the project's maximum staffing rate occurs at project start and is equal to life cycle effort divided by the square of development duration.

⁹ Jensen (1983A) and Putnam (1980) both use a system of units that measures effective software size in source lines of code (SLOC), effort in person-years, and duration in calendar years.

$$\bar{w} \propto \frac{1}{f(\dot{P}_{\max})} \text{ where } f(x) = x^a \quad \therefore \bar{w} = c_k \frac{1}{\dot{P}_{\max}^a} \quad (4)$$

$$\therefore E_p^{(1-a)} t_p^{2a} = \frac{S_e}{0.3839^a c_k} \text{ or } E_p = \left(\frac{S_e}{0.3839^a c_k} \right)^{\left(\frac{1}{1-a}\right)} t_p^{-\left(\frac{2a}{1-a}\right)}$$

Type 4—A bivariate relationship that assumes the product of power expressions for both process effort E_p and process duration t_p to be proportional to a power expression for effective software size S_e where the reciprocal of the constant of proportionality is assumed to be an efficiency scale factor η . Then

$$E_p^{\alpha_E} t_p^{\alpha_t} = \frac{S_e}{\eta} \text{ or } E_p = \left(\frac{S_e}{\eta} \right)^{\frac{1}{\alpha_E}} t_p^{-\left(\frac{\alpha_t}{\alpha_E}\right)} \quad (5)$$

The remainder of this paper describes the derivation of a Type 4 model hereinafter referred to as the **r2 Software Estimating Framework (r2SEF)**.

Note that the Type 2 form can be converted to an instantiation of the Type 4 form by multiplicatively combining the two resultant Equations (3).

$$E_p = \chi_1 (S_e)^{a_1} \text{ and } t_p = \chi_2 (E_p)^{a_2}$$

$$\therefore E_p^{\left(\frac{1-a_2}{a_1}\right)} t_p^{\left(\frac{1}{a_1}\right)} = (\chi_1 \chi_2)^{\left(\frac{1}{a_1}\right)} S_e \text{ or } E_p = \left(\frac{(\chi_1 \chi_2)^{\left(\frac{1}{a_1}\right)} S_e}{t_p^{\left(\frac{1}{a_1}\right)}} \right)^{\left(\frac{a_1}{1-a_2}\right)} \quad (6)$$

This Type 2 instantiation of Type 4 implies

$$\alpha_E = \frac{1-a_2}{a_1}, \alpha_t = \frac{1}{a_1}, \text{ and } \eta \propto \frac{1}{(\chi_1 \chi_2)^{\left(\frac{1}{a_1}\right)}} \quad (7)$$

Note also that the Type 3 form (resultant Equation (4)) can be viewed as an instantiation of the Type 4 form where the exponents of effort and duration are constrained by the Rayleigh-based assumption that process average productivity is inversely proportional to a power function of the project's maximum staffing rate.

This Type 3 instantiation of Type 4 implies

$$\alpha_E = 1-a \text{ and } \alpha_t = 2a \text{ and } \eta \propto 0.3839^a c_k \quad (8)$$

where for Seer (Jensen, 1983A) $a \equiv 0.5$ and for SLIM (Putnam, 1980) $a \equiv 0.6667$. Note that these are constants, the implication being that model nonlinearity is fixed and not sensitive to potential data set differences in the degree of nonlinearity between effort and duration.

2. OBSERVATIONS AND HYPOTHESES

Fundamental Observations

Intuition, personal experience, and analysis of historical project data (effort, duration, size, and defects) using ordinary least squares (OLS) regression in log-log space yields the following observations:

- *No free software*—Effort (and hence cost) trends upward as a function of increasing size (see **Figure 1** below).

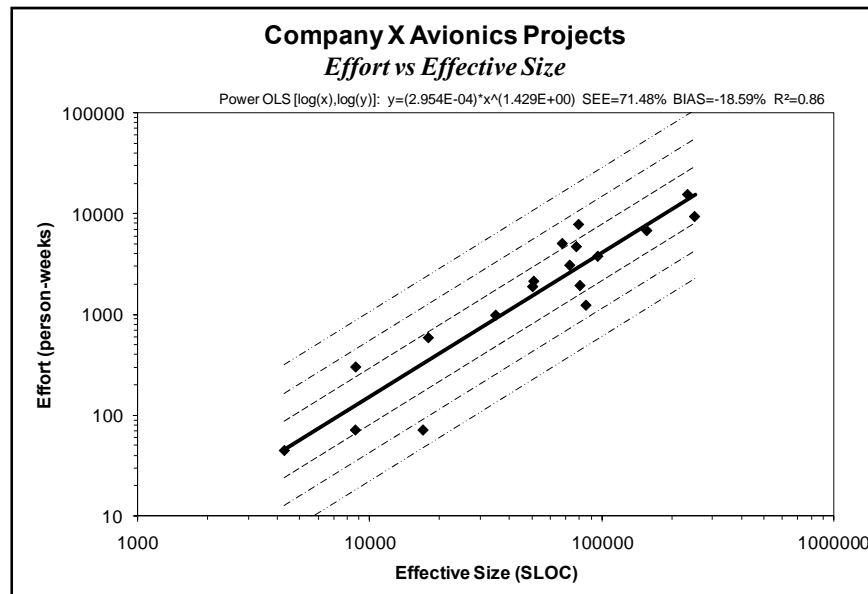


Figure 1. Effort Trends Upward as Size Increases^{10,11}

¹⁰The scatter charts in this paper are annotated with the following quality-of-fit metrics: SEE is the percentage standard error of estimate and is defined as the sample standard deviation of the percentage errors between the actual y_i 's and the estimated y_i 's ($f(x_i)$'s) (Book, 2006); BIAS is defined as the mean of the percentage errors between the actual y_i 's and the estimated y_i 's ($f(x_i)$'s) (Book, 2006); and R^2 is the square of the Pearson product-moment correlation coefficient between the actual y_i 's and the estimated y_i 's ($f(x_i)$'s) (Book, et al., 2006) p. 94.

¹¹For the Company X Avionics Projects data set used in this example, size is measured in effective Source Lines of Code (SLOC), sometimes referred to as Deliverable Source Statements (DSS) or logical lines of code, of the particular 3rd generation programming language used (Ada, C, Pascal, or PLM) according to a certain set of consistently-applied counting rules while effort and duration are measured in person-weeks and calendar weeks respectively for a specific set of life cycle activities. The r2SEF methodology presented in this paper is not limited to specific size, effort, and duration units. Size could be measured in function points according to International Function Points Users Group (IFPUG) standards or perhaps the number of use cases, objects, or web pages to be developed. Effort could also be measured in person-hours, person-months, person-quarters, or person-years and could include any subset of the activities from project inception to product delivery and beyond. Duration could also be measured in calendar months, quarters, or years. What matters is that the size unit and counting rules reasonably represent the actual work being done and that the size, effort, and duration of each project in a data set are measured in consistent units with consistent definitions and rules.

- *No instant software*—Duration trends upward as a function of increasing size (see Figure 2 below).

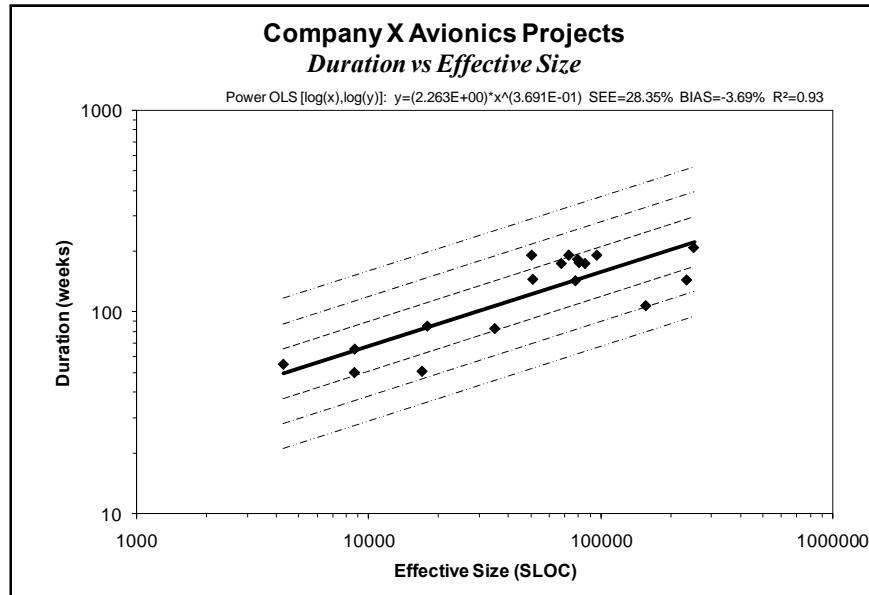


Figure 2. Duration Trends Upward as Size Increases

- *No perfect software*—Effort trends upward as a function of increasing defect count (see Figure 3 below).

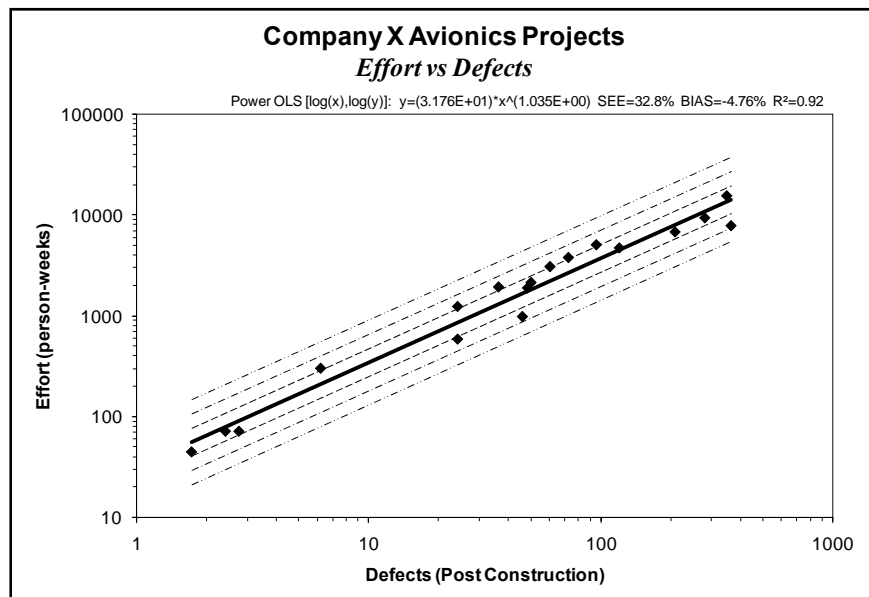


Figure 3. Effort Trends Upward as the Number of Defects Increases

- *No perfect software*—Duration trends upward as a function of increasing defect count (see **Figure 4** below).

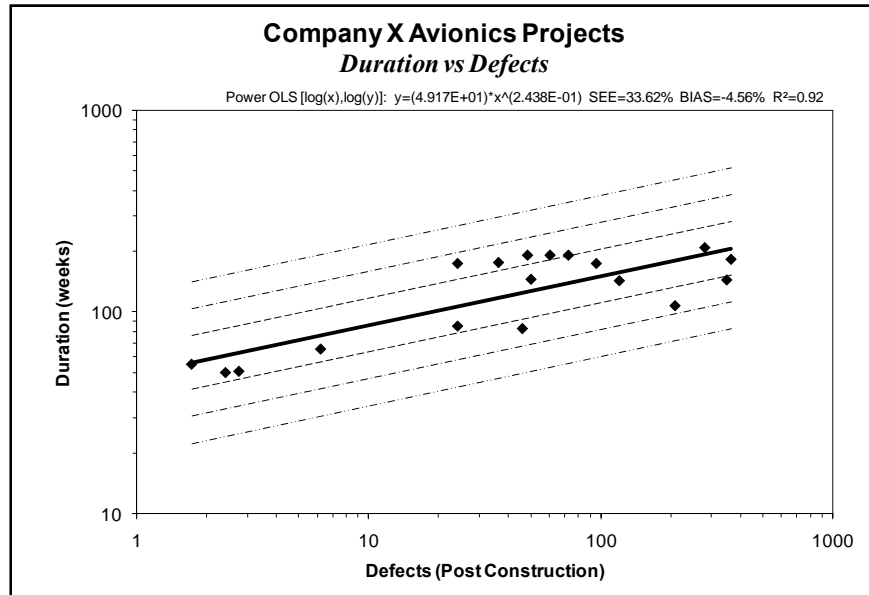


Figure 4. Duration Trends Upward as Number of Defects Increases

- *Smaller teams are more productive*—Productivity trends upward as a function of decreasing team size (see **Figure 5** below).

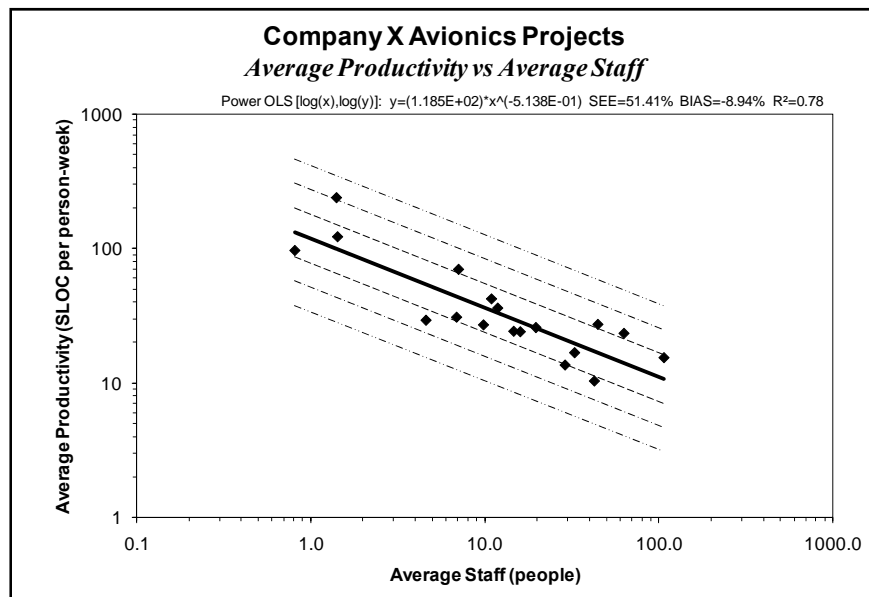


Figure 5. Productivity Trends Upward as Team Size Decreases

- **Smaller teams produce fewer defects**—Defect count trends upward as a function of increasing team size (see **Figure 6** below).

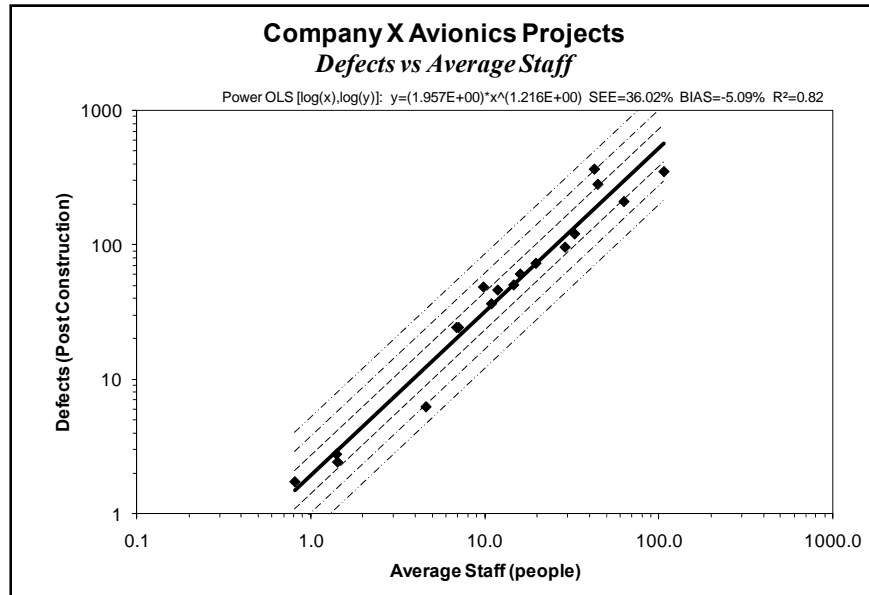


Figure 6. Defects Trend Upward as Team Size Increases

- **Projects seek a balance**—Irrespective of size, duration and effort are reasonably correlated, which suggests some inherent equilibrium between the two (see **Figure 7** below).

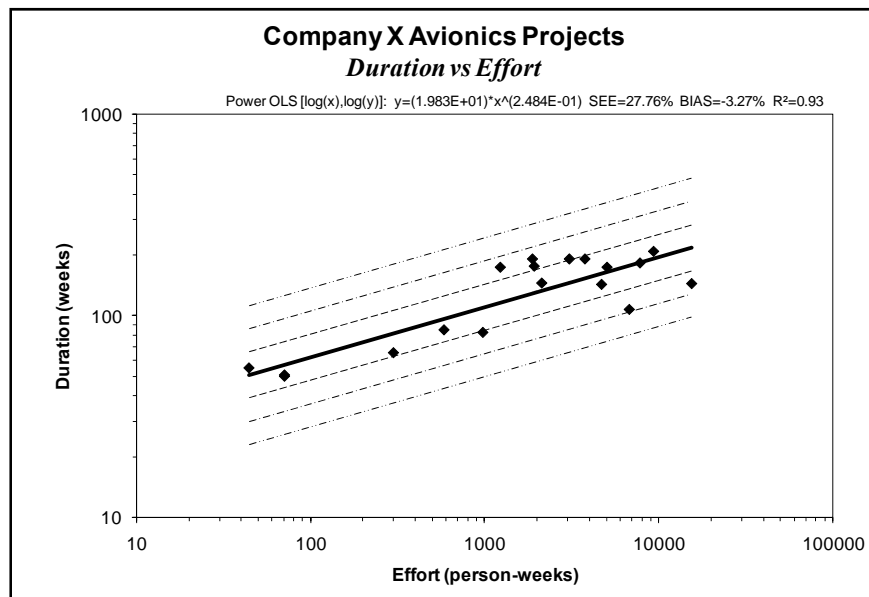


Figure 7. Equilibrium Between Duration and Effort

Resulting Suggested Hypotheses

Content (software size) is the desired product of labor and time—The amount of software produced is directly related to the resource *amount* (labor and time) applied; i.e., the *product* of labor and time.

Defects are the unwanted byproduct of labor and time—The number of defects produced is directly related to resource *intensity* (labor and time); i.e., the *ratio* of labor and time.

3. DEFINING EFFORT, DURATION, AND SIZE

Staffing, Duration, and Process Duration

We first conceptually define staffing to be some function P of elapsed calendar time t that describes, for a particular instance of some process, the application of people over time within the process's time interval. We define this time interval in absolute terms as $[T_{\text{start}}, T_{\text{finish}}]$ where T_{start} and T_{finish} represent the start and finish dates, respectively, of the process. In the interest of generalization, we prefer to use a T_{start} -relative frame to describe this interval; therefore, T_{start} relative to T_{start} is $T_{\text{start}} - T_{\text{start}} = 0$ and T_{finish} relative to T_{start} is $T_{\text{finish}} - T_{\text{start}}$, the value of which we will represent as t_p . The resulting T_{start} -relative process interval is $[0, t_p]$. Note that the value of t_p represents not only the T_{start} -relative point in time where the process finishes, it also represents the duration (elapsed calendar time) of the process interval.

Effort

With our conceptual definition of a staffing function P , we now define the concept of effort to be some function E of elapsed calendar time t that describes, for a particular process, the accumulated result of people laboring to do work over elapsed time t

$$E \equiv \int P dt \quad (9)$$

and is, in particular, the area under the staffing curve from 0 to the end of the elapsed time period.

Using Equation (9) as our definition of an effort function with respect to its associated staffing function we can now define an instantaneous staffing function with respect to its associated effort function by differentiating Equation (9) with respect to time t and then solving for P :

$$\begin{aligned} \frac{dE}{dt} &= \frac{d}{dt} \int P dt \\ P &= \frac{dE}{dt} \end{aligned} \quad (10)$$

Process Effort

We have already defined $[0, t_p]$ to be the T_{start} -relative process time interval where t_p represents process duration. We now define process effort E_p to be the change in effort within this process time interval.

$$E_p \equiv E(t_p) - E(0) \quad (11)$$

where $E(0) \equiv 0$; i.e., no process effort can be spent before the process starts.

Effective Software Size (Content)

We describe the software development process as transforming one abstraction (the desire or the requirements) to another abstraction (the software product). Each and every abstraction, be it expressed in a natural language, a programming language, or even as graphic constructs; consists of primitive elements that we refer to as *size units*. Examples of commonly-used size units include Source Lines of Code (SLOC), function points, use cases, objects, methods, classes, and web pages; basically something that can be *consistently* counted and reasonably represents the work that must be done. We choose here to define the notion of *effective* software size S_e of a particular abstraction to be the number (count) of size units in the abstraction that are considered to be directly related to the resources (labor over time) necessary to develop said software; this includes developing new software plus selecting, understanding, incorporating, changing, and/or verifying any included legacy software¹².

4. EFFORT-DURATION TRADEOFF RELATIONSHIP

Software Development Process Law

Software is made by people laboring to do work over some period of time; the result being neither free, instant, nor perfect. We have already shown that effort E_p and duration t_p trend upward (and in most cases non-linearly) as functions of increasing effective software size (see **Figure 1** and **Figure 2**). Our first empirically-suggested hypothesis states that *content (software size) is the desired product of labor and time*—the amount of software produced is directly related to the resource amount (labor and time) applied; i.e., the product of labor and time. We therefore propose the following generalized relationship:

$$f_E(E_p) f_t(t_p) \propto f_S(S_e) \quad \therefore f_E(E_p) f_t(t_p) = b f_S(S_e) \quad (12)$$

where b represents the constant of proportionality.

Performing ordinary (linear) least squares (OLS) regression in log-log space on data from past projects suggests that both effort and duration are reasonably correlated with effective software size (see **Figure 1** and **Figure 2**). These correlations can be generally and reasonably modeled by power functions described as

¹² Examples of legacy software include Commercial Off-The Shelf (COTS) software, reused software, and software from a previous build, increment, or release.

$$f_E(x) \equiv x^{a_E}, f_t(x) \equiv x^{a_t}, \text{ and } f_S(x) \equiv x^{a_S} \quad (13)$$

Substituting Equations (13) into Equation (12) yields

$$E_p^{a_E} t_p^{a_t} = b(S_e)^{a_S} \quad (14)$$

We then scale the exponents a_E and a_t to force the exponent on size to unity by replacing

$$a_E \text{ by } \alpha_E a_S \text{ and } a_t \text{ by } \alpha_t a_S \quad (15)$$

Note the distinction between the English letters “a” and the Greek letters α (alpha). Substituting Equations (15) into Equation (14) yields

$$E_p^{\alpha_E a_S} t_p^{\alpha_t a_S} = b(S_e)^{a_S} \quad \therefore E_p^{\alpha_E} t_p^{\alpha_t} = b^{\left(\frac{1}{a_S}\right)} S_e \quad (16)$$

We next introduce the concept of data sample **mean efficiency** $\bar{\eta}$ (lower-case Greek eta bar) and choose to define it as being equal to the reciprocal of the above coefficient of effective software size:

$$\bar{\eta} \equiv \frac{1}{b^{\left(\frac{1}{a_S}\right)}} \quad \therefore b^{\left(\frac{1}{a_S}\right)} = \frac{1}{\bar{\eta}} \quad (17)$$

Substituting Equation (17) into Equation (16) yields

$$E_p^{\alpha_E} t_p^{\alpha_t} = \left(\frac{1}{\bar{\eta}}\right) S_e \quad \therefore E_p^{\alpha_E} t_p^{\alpha_t} = \frac{S_e}{\bar{\eta}} \quad (18)$$

Calibrating Equation (18) for a particular data set; i.e., estimating values for α_E , α_t , and $\bar{\eta}$ that provide an acceptable estimating relationship for a specific historical data set presents an interesting challenge. The approach we take is a two-step process where we first estimate relationship exponent values α_E and α_t and then estimate a value for mean efficiency $\bar{\eta}$ that minimizes the percentage Standard Error of Estimate (SEE) with zero percent bias.

We begin with two general regressions using the Minimum Percent Error – Zero Percent Bias (MPE-ZPB) technique for *power* estimating relationships of the form $y = bx^a$ described by Book (2006).

$$E_{p_actual} \text{ versus } S_{e_actual} \text{ and } t_{p_actual} \text{ versus } E_{p_actual} \quad (19)$$

The resulting two power functions are respectively:

$$E_{p_actual} = b_1 S_{e_actual}^{a_1} \quad (20)$$

and

$$t_{p_actual} = b_2 E_{p_actual}^{a_2} \quad (21)$$

The resulting values and statistics from these regressions are:

$a_1 = 1.340$	$a_2 = 0.2681$	(22)
$b_1 = 0.0009182$	$b_2 = 17.74$	
$R^2 = 0.8684$	$R^2 = 0.9289$	
$SEE = 57.13\%$	$SEE = 26.49\%$	
$BIAS = 0$	$BIAS = 0$	

Multiplicatively combining Equation (20) and Equation (21) yields

$$E_{p_actual} t_{p_actual} = b_1 S_{e_actual}^{a_1} b_2 E_{p_actual}^{a_2} \quad \therefore E_{p_actual}^{\left(\frac{1-a_2}{a_1}\right)} t_{p_actual}^{\left(\frac{1}{a_1}\right)} = \frac{S_{e_actual}}{\left(\frac{1}{b_1 b_2}\right)^{\left(\frac{1}{a_1}\right)}} \quad (23)$$

Instantiating the general form Equation (18) with the regression-derived Equation (23) implies the following assignments on each of α_E , and α_t :

$$\alpha_E = \frac{1-a_2}{a_1} \quad (24)$$

$$\alpha_t = \frac{1}{a_1} \quad (25)$$

Substituting the Company X Avionics Projects data set results from (22) above into Equations (24) and (25) we get

$$\alpha_E = \frac{1-0.2681}{1.340} = 0.5462 \quad (26)$$

$$\alpha_t = \frac{1}{1.340} = 0.7463 \quad (27)$$

Instantiating the general form Equation (18) with the values in Equations (26) and (27) yields the Company X Avionics Projects data-set-specific estimating relationship:

$$E_p^{0.5462} t_p^{0.7463} = \frac{S_e}{\bar{\eta}} \quad (28)$$

We can treat this estimating relationship (28) as a *factor* estimating relationship of the form $y = ax$ where

$$x \equiv S_e \quad y \equiv E_p^{0.5462} t_p^{0.7463} \quad a \equiv \frac{1}{\bar{\eta}} \quad (29)$$

We now perform the MPE-ZPB general regression technique for *factor* estimating relationships described by Book (2006) in order to find the value for a and hence $\bar{\eta}$ that results in minimum percentage Standard Estimate of Error with zero percent bias. The result of this regression is illustrated in **Figure 8** below.

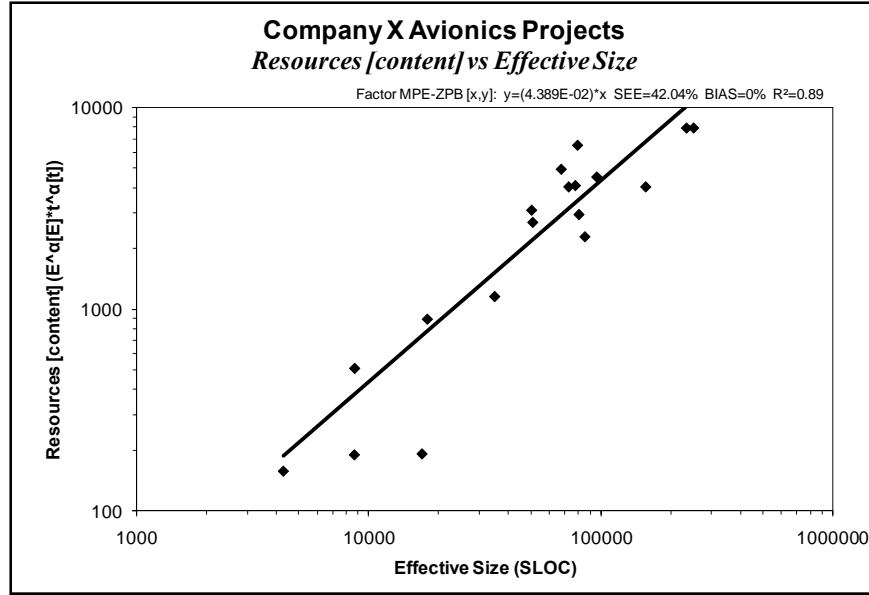


Figure 8. Resources versus Size

Since **Figure 8** shows $a = 0.04389$, it follows from the third definition in (29) that $\bar{\eta} = 1/a = 22.79$. Therefore, the instantiated estimating relationship (28) for the Company X Avionics Projects data set is:

$$E_p^{0.5462} t_p^{0.7463} = \frac{S_e}{22.79} \quad (30)$$

Note in **Figure 8** the relatively good relationship quality values (SEE, BIAS, and R^2) which are indications of a reasonably good estimating relationship.

Specific Efficiency

We now introduce the notion of a process's *specific efficiency* η (lower-case Greek eta). Each instance of a software development process has a unique value for specific efficiency within the context of a particular data set; i.e., a particular instantiation of Equation (18). Practically speaking, this means a value for specific efficiency of a process instance relates that process instance to other process instances in the particular historical data set for which Equation (18) has been instantiated. We write this instantiation and its various solved forms as:

$$\left[E_p^{\alpha_E} t_p^{\alpha_t} = \frac{S_e}{\eta} \right]_A, \left[E_p = \left(\frac{S_e}{\eta t_p^{\alpha_t}} \right)^{\left(\frac{1}{\alpha_E} \right)} \right]_A, \left[t_p = \left(\frac{S_e}{\eta E_p^{\alpha_E}} \right)^{\left(\frac{1}{\alpha_t} \right)} \right]_A, \quad (31)$$

$$\left[\eta = \frac{S_e}{E_p^{\alpha_E} t_p^{\alpha_t}} \right]_A, \text{ or } \left[S_e = \eta E_p^{\alpha_E} t_p^{\alpha_t} \right]_A$$

We use the square bracket symbols with a postfix subscript to mean “within the context of the data set named **A**”.

Effort-Duration Equation in Terms of the Software Product

The equations in (31) are various forms of the *fundamental content productivity equation*. It describes the tradeoff relationship between total process effort E_p and total process duration t_p as a function of effective software size S_e and the process's specific efficiency η within the context of some relevant historical data set \mathbf{A} .

Figure 9 illustrates an example of the content productivity equation (Equation (31)) instantiated with the values from our Company X Avionics Projects data set example and assuming an effective size of 50,000 Source Lines of Code (SLOC) with average efficiency ($\eta = \bar{\eta}$):

$$\left[E_p^{0.5462} t_p^{0.7463} = \frac{50,000 \text{ (SLOC)}}{22.79} \right]_{\text{Company X Avionics Projects}} \quad (32)$$

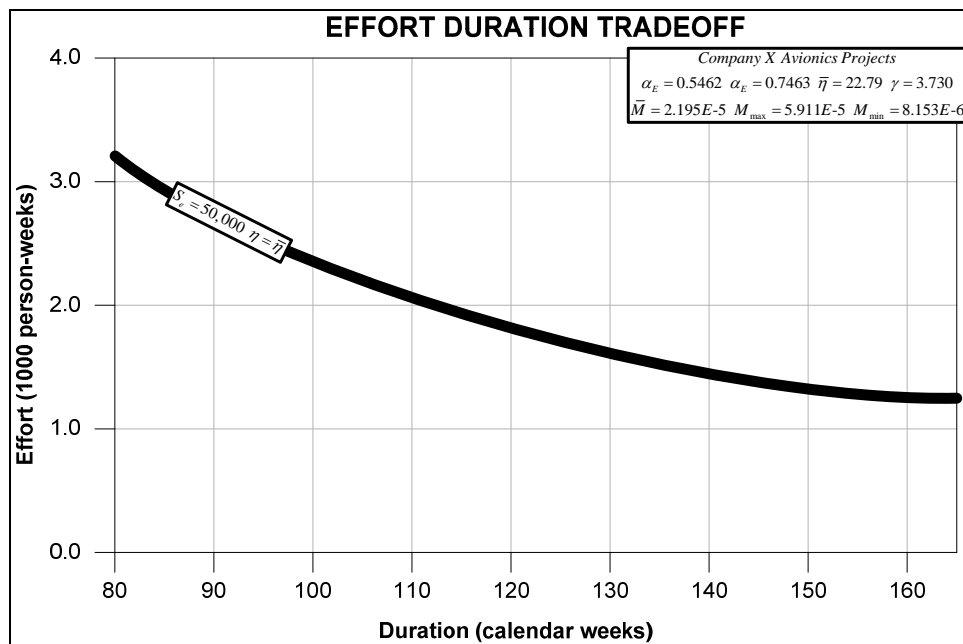


Figure 9. Example Tradeoff Curve (Content productivity Law)

5. DESCRIBING A PARTICULAR EFFORT-DURATION SOLUTION

Management Stress

The notion of management stress was suggested by Jensen (1983A) and described as the inherent equilibrium between effort and duration for software development processes, this equilibrium being independent of effective software size and specific efficiency and being constrained by the earlier-described Rayleigh-shape staffing assumption (see footnote 8).

We choose to redefine this notion of management stress by eliminating the Rayleigh-shape staffing assumption constraint and by more-generally postulating that process duration t_p is proportional to some function f of process effort E_p . In other words, for all process instances in a

particular data set, ignoring the variety of effective software sizes and of specific efficiencies, as the process effort increases, the process duration tends to increase and *vice versa*. Stated mathematically

$$t_p \propto f(E_p) \quad \therefore t_p = bf(E_p) \quad (33)$$

where b represents the constant of proportionality.

Performing OLS regression in log-log space, i.e., $\log(t_{p_actual})$ versus $\log(E_{p_actual})$ on various past project data sets (stratified by neither size nor efficiency) indicates that process duration trends upward (and in most cases non-linearly) as a function of increasing process effort. An example of this is shown in **Figure 7** and the result is expressed as Equation (21). We use Equations (24) and (25) to express Equation (21) in terms of the content productivity equation's exponents α_E and α_t :

$$t_p = b_2 E_p^{a_2} \quad t_p = b_2 E_p^{(1-\alpha_t \alpha_E)} \quad t_p = b_2 E_p^{\left(\frac{\alpha_t - \alpha_E}{\alpha_t}\right)} \quad \therefore \frac{1}{b_2^{\left(\frac{\alpha_t}{\alpha_t - \alpha_E}\right)}} = \frac{E_p}{t_p^{\left(\frac{\alpha_t}{\alpha_t - \alpha_E}\right)}} \quad (34)$$

Renaming the exponent of t_p in Equation (34) to γ (lower-case Greek gamma) yields

$$\frac{1}{b_2^\gamma} = \frac{E_p}{t_p^\gamma} \quad (35)$$

Practically speaking, Equation (35) implies that as the process duration is increased, the resultant process effort trends upward, this trend characterized by the parameters γ and b_2 . From a practical standpoint, γ represents the *economy* or diseconomy associated with higher process durations. Note that Type 2 models such as COCOMO (Boehm, 1981) imply $\gamma \equiv 1/a_2$. Note also that Seer (Jensen, 1983A) and SLIM (Putnam, 1980) both assume $\gamma \equiv 3$, a constant rather than being data set dependent.

We now introduce the concept of data sample *mean management stress* \bar{M} and choose to define it as being equal to the left side of Equation (35) above:

$$\bar{M} \equiv \frac{1}{b_2^\gamma} \quad (36)$$

Substituting Equation (36) into Equation (35) yields

$$\bar{M} = \frac{E_p}{t_p^\gamma} \quad (37)$$

Values for γ and \bar{M} in Equation (37) for a specific historical data set can now be determined from the a_2 and b_2 values of the previously-described MPE-ZPB general regression of Equation (21) using Equations (34), (24), and (25) as follows:

$$\gamma = \frac{\alpha_i}{\alpha_i - \alpha_E} = \frac{1}{a_2} \quad (38)$$

$$\bar{M} = \frac{1}{b_2^\gamma} \quad (39)$$

If we use the Company X Avionics Projects data set and the MPE-ZPB regression results in (22) to supply the values $a_2 = 0.2681$ and $b_2 = 17.74$ and then substitute these assignments into Equations (38) and (39), we get

$$\gamma = \frac{1}{0.2681} = 3.730 \quad (40)$$

$$\bar{M} = \frac{1}{17.74^{3.730}} = 2.195\text{E-}5 \quad (41)$$

Specific Management Stress

We now introduce the notion of a project's *specific management stress* M . Each instance of a software development process has a unique value for specific management stress within the context of a particular data set; i.e., a particular instantiation of Equation (37). Practically speaking, this means a value for specific management stress of a process instance relates that process instance to other process instances in the particular historical data set for which Equation (37) has been instantiated. We write this instantiation as:

$$\left[M = \frac{E_p}{t_p^\gamma} \right]_A, \left[t_p = \left(\frac{E_p}{M} \right)^{\left(\frac{1}{\gamma} \right)} \right]_A, \text{ or } \left[E_p = M t_p^\gamma \right]_A \quad (42)$$

Size-Independent Effort-Duration Equation

Equation (42) is the *fundamental management stress equation*. It describes the relationship between process effort and process duration, within the context of a particular data set, which is independent of effective size and specific efficiency. It is used in the next two sections to isolate particular effort-duration solutions for given values of effective software size and specific efficiency. It is also used later in the paper as the basis for determining the feasible limits of the content productivity equation (minimum duration solution and minimum effort solution).

Solving for Process Duration

Substituting the solved-for-effort form of Equation (42) into Equation (31) we get process duration, this being a function of the specific management stress, the effective software size, and the specific efficiency.

$$\left[\left(M t_p^\gamma \right)^{\alpha_E} t_p^{\alpha_i} = \frac{S_e}{\eta} \right]_A \quad \therefore \left[t_p = \left(\frac{1}{M} \right)^{\left(\frac{\alpha_E}{\gamma\alpha_E + \alpha_i} \right)} \left(\frac{S_e}{\eta} \right)^{\left(\frac{1}{\gamma\alpha_E + \alpha_i} \right)} \right]_A \quad (43)$$

Solving for the Process Effort Associated with a Particular Process Duration

Substituting the solved-for-time form of Equation (42) into Equation (43) we get the process effort associated with a given process duration as a function of the effective management stress, the effective software size, and the specific efficiency.

$$\left[\left(\frac{E_p}{M} \right)^{\left(\frac{1}{\gamma} \right)} = \left(\frac{1}{M} \right)^{\left(\frac{\alpha_E}{\gamma\alpha_E + \alpha_t} \right)} \left(\frac{S_e}{\eta} \right)^{\left(\frac{1}{\gamma\alpha_E + \alpha_t} \right)} \right]_A \quad \therefore \quad \left[E_p = (M)^{\left(\frac{\alpha_t}{\gamma\alpha_E + \alpha_t} \right)} \left(\frac{S_e}{\eta} \right)^{\left(\frac{\gamma}{\gamma\alpha_E + \alpha_t} \right)} \right]_A \quad (44)$$

Figure 10 uses Equations (43) and (44) instantiated with the r2SEF parameters from our Company X Avionics Projects data set example to illustrate how the management stress equation is used to locate a particular effort-duration solution for a particular instance of the content productivity equation, in this case locating the average stress solution.

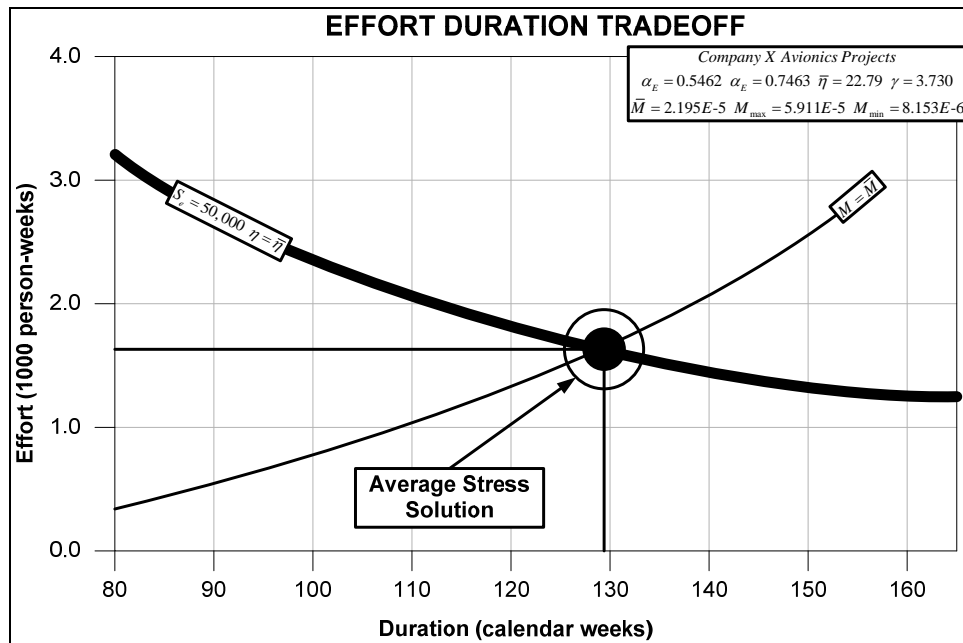


Figure 10. Average Stress Solution

6. MINIMUM DURATION

Brooks' Law

Adding manpower to a late software project makes it later (Brooks, 1995). Each and every instance of a software development process, by its nature (divisibility or potential for concurrency), can effectively handle only so much management stress (only so many people); therefore, there exists, for each and every instance of a software development process, some minimum achievable process duration. Software development, like the aging of a fine wine, takes time and

cannot be rushed beyond a *certain point*.¹³ Jensen (1983A), Putnam (1980), and others have analyzed historical project data and concluded that this *certain point* can be defined in terms of a project's maximum achievable specific management stress; a metric that attempts to quantify a particular process's staffing complexity, divisibility, and amenability to task parallelism.

Maximum-Achievable Specific Management Stress

It seems reasonable to assume that for each instance of a process there exists some theoretical limit on how quickly it can be completed. One possible explanation for this limit, especially with a labor-intensive process such as software development, is that the complexity and divisibility of a problem and the capability of the project's management each tend to limit how many people can effectively work together on that problem; i.e., only so much task parallelism and management span of control are possible. Exceeding this limit will likely increase the effort without reducing the duration and may, according to Brooks (1995), *increase* the duration. Since we have defined management stress as a ratio of effort to duration, independent of size and efficiency, where large effort and short duration indicate high stress while small effort and long duration indicate low stress, we choose to describe the aforementioned minimum duration limit in terms of a specific management stress value M_{\max} within the context of a particular historical data set.

Mathematically, the M_{\max} limit can be expressed as

$$\left[M_{\max} \geq M \right]_{\mathbf{A}} \quad (45)$$

Since this notion of a *maximum-achievable* value is somewhat theoretical, we suggest a conservatively-practical value for M_{\max} in estimation situations to be

$$\left[M_{\max} = \left(e^{\text{stdev}(\ln(\mathbf{M}_i))} \right)^{(+1.0)} \bar{M} \right]_{\mathbf{A}} \quad (46)$$

where

\mathbf{M}_i = the vector of M values from the projects in data set \mathbf{A} .

Substituting Equation (42) into Equation (45) yields

$$\left[M_{\max} \geq \frac{E_p}{t_p^\gamma} \right]_{\mathbf{A}} \quad \text{or} \quad \left[E_p \leq M_{\max} t_p^\gamma \right]_{\mathbf{A}} \quad (47)$$

Figure 11 shows the region excluded by Equation (47) in crosshatch for an example M_{\max} value. The curve described by the margin between the crosshatch region and the white region is the minimum duration limiting function (Equation (47) as an equality).

¹³ Analogy frequently used by Dr. Randy Jensen in numerous presentations on this topic.

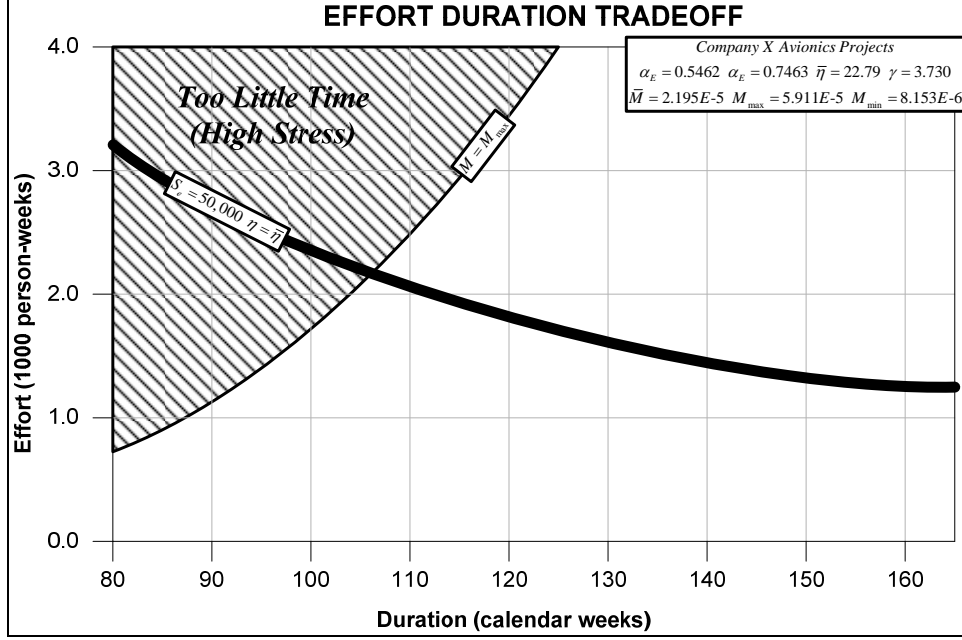


Figure 11. Minimum Duration Limit

Since maximum-achievable specific management stress M_{\max} bounds the maximum amount of parallelism that can be incorporated into project scheduling, Equation (47) implies the existence of a minimum-achievable duration $t_{p\min}$ with associated effort $E_{t_{p\min}}$.

Solving for Minimum Process Duration

Instantiating Equation (43) with $t_{p\min}$ and M_{\max} we get the minimum process duration, this being a function of the maximum-achievable specific management stress, the effective software size, and the specific efficiency.

$$\left[t_{p\min} = \left(\frac{1}{M_{\max}} \right)^{\left(\frac{\alpha_E}{\gamma\alpha_E + \alpha_t} \right)} \left(\frac{S_e}{\eta} \right)^{\left(\frac{1}{\gamma\alpha_E + \alpha_t} \right)} \right]_A \quad (48)$$

Solving for Effort Associated with Minimum Process Duration

Instantiating Equation (44) with $E_{t_{p\min}}$ and M_{\max} we get the effort associated with the minimum process duration.

$$\left[E_{t_{p\min}} = (M_{\max})^{\left(\frac{\alpha_t}{\gamma\alpha_E + \alpha_t} \right)} \left(\frac{S_e}{\eta} \right)^{\left(\frac{\gamma}{\gamma\alpha_E + \alpha_t} \right)} \right]_A \quad (49)$$

Figure 12 shows the minimum process duration and its associated effort as one of the solutions on our example effort versus duration tradeoff curve. The minimum duration solution is defined

as the intersection of the fundamental content productivity equation (Equation (31)) and the Brooks' Law or minimum duration limiting function (Equation (47) as an equality).

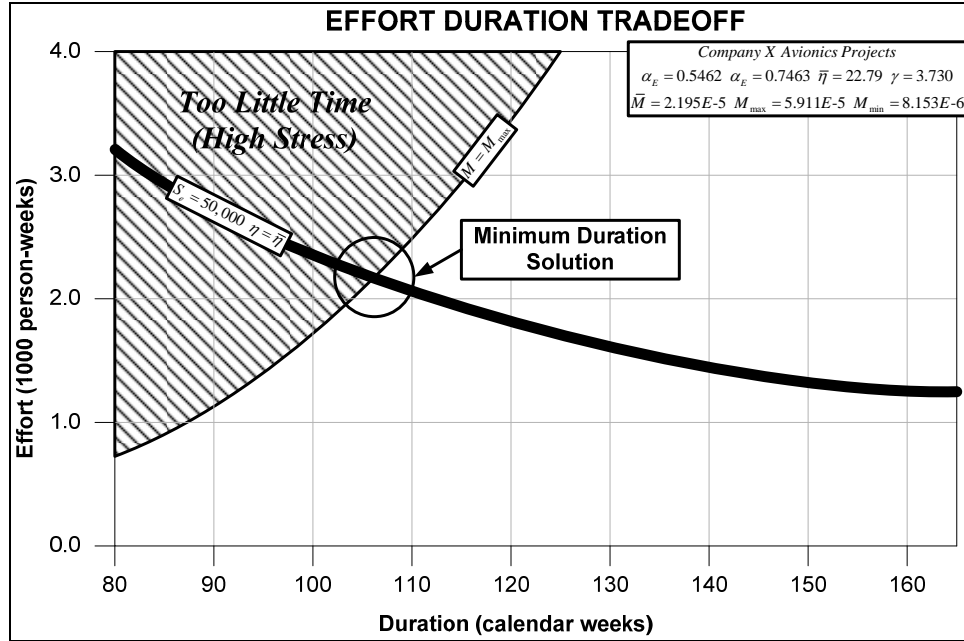


Figure 12. Minimum Duration Solution

7. MINIMUM EFFORT

Parkinson's Law

Work expands so as to fill the time available for its completion (Parkinson, 1958). Theoretically, a specific instance of a software development process is not limited by some maximum process duration. Rare is the software engineer who complains about having too much time to develop software. However, we argue that there exists, for each and every project, some duration that yields maximum productivity; i.e., some duration that represents the most efficient combination of project decomposition and corresponding use of labor.

Minimum-Practical Specific Management Stress

For each instance of a software development process, we submit that maximum productivity occurs at some point of minimum-practical specific management stress. This point of minimum practical specific management stress M_{\min} defines the optimum use of people over time, and represents a practical limit to the benefit of schedule relaxation.

Mathematically, the M_{\min} limit can be expressed as

$$[M_{\min} \leq M]_A \quad (50)$$

Since this notion of a *minimum-practical* value is somewhat theoretical, we suggest a conservatively-practical value for M_{\min} in estimation situations to be

$$\left[M_{\max} = \left(e^{\text{stdev}(\ln(M_i))} \right)^{(-1.0)} \bar{M} \right]_{\mathbf{A}} \quad (51)$$

where

\mathbf{M}_i = the vector of M values from the projects in data set \mathbf{A} .

Substituting Equation (42) into Equation (50) yields

$$\left[M_{\min} \leq \frac{E_p}{t_p^\gamma} \right]_{\mathbf{A}} \quad \text{or} \quad (52)$$

$$\left[E_p \geq M_{\min} t_p^\gamma \right]_{\mathbf{A}}$$

Figure 13 shows the region excluded by Equation (52) in crosshatch given a value for minimum specific management stress M_{\min} . Note that the curve described by the margin between the crosshatch region and the white region is the minimum effort limiting function (Equation (52) as an equality).

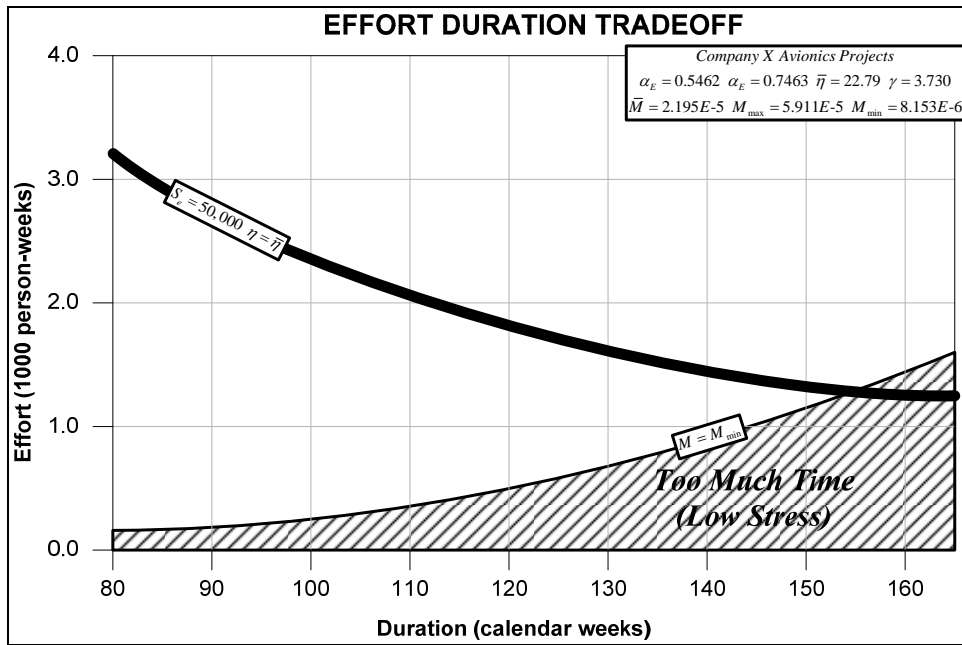


Figure 13. Minimum Effort Limit

Since minimum specific management stress M_{\min} bounds the minimum effective amount of parallelism that can be assumed before Parkinson's law overrides the cost benefit of schedule relaxation, Equation (52) implies the existence of a minimum-achievable effort $E_{p_{\min}}$ with associated duration $t_{E_{p_{\min}}}$.

Solving for Minimum Process Effort

Instantiating Equation (44) with $E_{p\min}$ and M_{\min} we get the minimum process effort, this being a function of the minimum specific management stress, the effective software size, and the specific efficiency.

$$\left[E_{p\min} = (M_{\min})^{\left(\frac{\alpha_t}{\gamma\alpha_E + \alpha_t}\right)} \left(\frac{S_e}{\eta}\right)^{\left(\frac{\gamma}{\gamma\alpha_E + \alpha_t}\right)} \right]_A \quad (53)$$

Solving for Duration Associated with Minimum Process Effort

Instantiating Equation (43) with $t_{E_{p\min}}$ and M_{\min} we get the duration associated with the minimum process effort.

$$\left[t_{E_{p\min}} = \left(\frac{1}{M_{\min}}\right)^{\left(\frac{\alpha_E}{\gamma\alpha_E + \alpha_t}\right)} \left(\frac{S_e}{\eta}\right)^{\left(\frac{1}{\gamma\alpha_E + \alpha_t}\right)} \right]_A \quad (54)$$

Figure 14 shows the minimum process effort and its associated duration as one of the solutions on our example effort versus duration tradeoff curve. Note that the minimum effort solution is defined as the intersection of the fundamental content productivity equation (Equation (31)) and the Parkinson's Law or minimum effort limiting function (Equation (52) as an equality)).

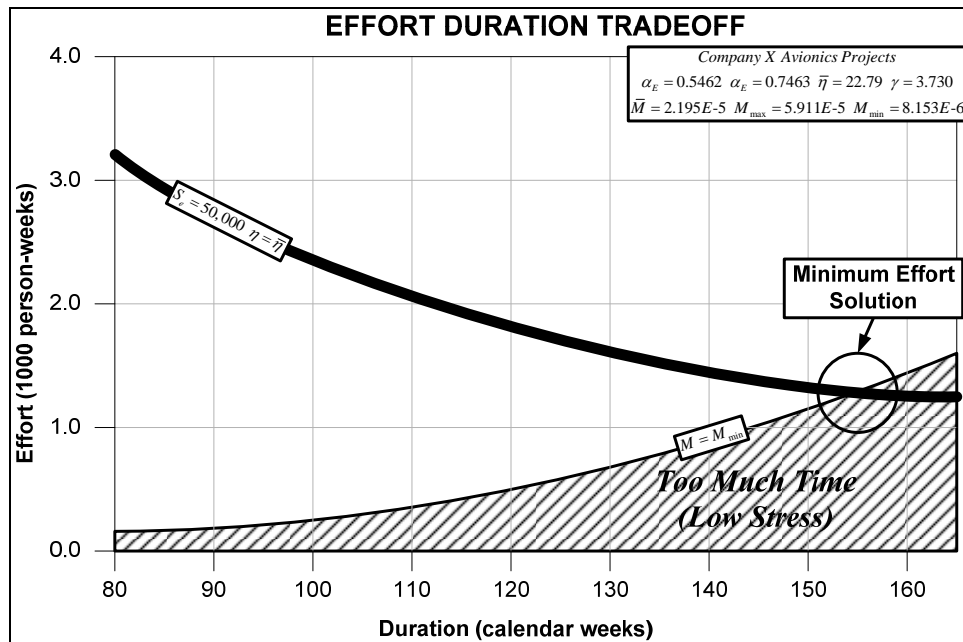


Figure 14. Minimum Effort Solution

8. DEFECTS

Taxonomy

In order to derive the defects estimating portion of the r2SEF it is useful to establish definitions of the terminology associated with software reliability. For the most part, we choose to adopt Musa's (2004) definitions with minor modification:

Failure—“...departure of the external results of system operation from user [desires].”¹⁴

Fault—“...defective, missing, or extra [expression unit] or set of related [expression units] that is the cause of one or more actual or potential failures.”¹⁵

Error—“...incorrect or missing action by a person or persons that causes a fault in a program.”¹⁶

Problem or Issue—The *documented* observation of some undesired aspect or aspects of a software product and/or its related documentation; typically contained in the form of an artifact; e.g., problem report, trouble report, issue report, etc. Problems, according to this definition, are very imprecise (such is the nature of problem reporting systems) and can contain, within their scope, various combinations of failures, faults, errors, and suggested enhancements. Duplicates and overlaps are not uncommon.

Defect—The *documented* result of a problem review process that seeks to identify and isolate the fault(s) associated with a particular problem. This definition implies a review process with one of its objectives being to achieve a near one-to-one correspondence between defects and faults.

Unacceptable Defect—A defect, the removal of which, is considered a necessary condition for software product acceptance (e.g., certification, customer buy-off, general availability, contract satisfaction, etc.).

Defect Occurrence Function

We first conceptually define cumulative defect occurrence to be some function Φ (upper-case Greek phi) of elapsed calendar time t that describes, for a particular process, the accumulation of discovered defects over elapsed calendar time within a software development process time interval $[0, t_p]$.

Defect Occurrence Rate Function

With our conceptual definition of cumulative defect occurrence Φ , we now define the concept of defect occurrence rate (sometimes referred to as instantaneous defect occurrence) to be some function $\dot{\Phi}$ of elapsed calendar time t that describes, for a particular project, the rate that defects are being discovered at a particular point in time within the process time interval $[0, t_p]$.

$$\dot{\Phi} \equiv \frac{d}{dt} \Phi \quad (55)$$

¹⁴ (Musa, 2004 p. 208)

¹⁵ (Musa, 2004 p. 213)

¹⁶ (Musa, 2004 p. 215)

which implies

$$\Phi = \int \dot{\Phi} dt \quad (56)$$

Software Development Process Law (Revisited)

Software is made by people doing work over some period of time; the result being neither free, instant, nor perfect. We have already concluded, from our fundamental observations, that defect count $\Phi_{[a,b]}$ (number of defects discovered during the T_{start} -relative time interval $[t_a, t_b]$) trends upward (and in most cases non-linearly) as a function of increasing effort. Practically speaking, this implies that a software development process has two primary outputs: the deliverable product software and the defects that it contains; i.e., defects, like code, can be considered products (albeit undesirable) of the process. Our second empirically-suggested hypothesis (introduced earlier in this paper) states that **defects are the unwanted byproduct of labor and time**—the number of defects produced is directly related to resource *intensity* (labor and time); i.e., the *ratio* of labor and time. We therefore propose the following generalized relationship:

$$\frac{\mathbf{g}_E(E_p)}{\mathbf{g}_t(t_p)} \propto \mathbf{g}_\Phi(\Phi_{[a,b]}) \quad \therefore \frac{\mathbf{g}_E(E_p)}{\mathbf{g}_t(t_p)} = b \mathbf{g}_\Phi(\Phi_{[a,b]}) \quad (57)$$

where b represents the constant of proportionality.

Performing OLS regression in log-log space on data from past projects indicates that both effort and duration are reasonably correlated with total defects (see **Figure 3** and **Figure 4**). This non-linear behavior is best predicted (yields the highest correlation) by power functions described as

$$\mathbf{g}_E(x) \equiv x^{a_E}, \mathbf{g}_t(x) \equiv x^{a_t}, \text{ and } \mathbf{g}_\Phi(x) \equiv x^{a_\Phi} \quad (58)$$

Substituting Equations (58) into Equation (57) yields

$$E_p^{a_E} t_p^{-a_t} = b \Phi_{[a,b]}^{a_\Phi} \quad (59)$$

We then scale the exponents a_E and a_t to force the exponent on total defects to unity by replacing

$$a_E \text{ by } \varphi_E a_\Phi \text{ and } -a_t \text{ by } \varphi_t a_\Phi \quad (60)$$

Substituting Equations (60) into Equation (59) yields

$$E_p^{\varphi_E a_\Phi} t_p^{\varphi_t a_\Phi} = b \Phi_{[a,b]}^{a_\Phi} \quad \therefore E_p^{\varphi_E} t_p^{\varphi_t} = b^{\left(\frac{1}{a_\Phi}\right)} \Phi_{[a,b]} \quad [\varphi_t \leq 0] \quad (61)$$

We next introduce the concept of data sample mean defect vulnerability $\bar{\delta}_{[a,b]}$ (lower-case Greek delta bar) and choose to define it as being equal to the reciprocal of the above coefficient of total defects.

$$\bar{\delta}_{[a,b]} \equiv \frac{1}{b^{\left(\frac{1}{a_s}\right)}} \quad \therefore b^{\left(\frac{1}{a_s}\right)} = \frac{1}{\bar{\delta}_{[a,b]}} \quad (62)$$

Substituting Equation (62) into Equation (61) yields

$$E_p^{\varphi_E} t_p^{\varphi_t} = \left(\frac{1}{\bar{\delta}_{[a,b]}} \right) \Phi_{[a,b]} \quad \therefore E_p^{\varphi_E} t_p^{\varphi_t} = \frac{\Phi_{[a,b]}}{\bar{\delta}_{[a,b]}} \quad [\varphi_t \leq 0] \quad (63)$$

We can estimate values for φ_E , φ_t , and $\bar{\delta}_{[a,b]}$ in Equation (63) for a specific historical data set using the two-step process described earlier in this paper for the content productivity equation. In this case we first estimate relationship exponent values φ_E and φ_t and then estimate a value for mean defect vulnerability $\bar{\delta}_{[a,b]}$ that minimizes SEE with no bias.

We begin with two general regressions using the MPE-ZPB technique for *power* estimating relationships of the form $y = bx^a$.

$$E_{p_actual} \text{ versus } \Phi_{[a,b]_actual} \text{ and } t_{p_actual} \text{ versus } E_{p_actual} \quad (64)$$

The resulting two power functions are respectively:

$$E_{p_actual} = b_3 \Phi_{[a,b]_actual}^{a_3} \quad (65)$$

and

$$t_{p_actual} = b_2 E_{p_actual}^{a_2} \quad (66)$$

The resulting values and statistics from these regressions are:

$a_3 = 1.043$	$a_2 = 0.2681$	(67)
$b_3 = 32.32$	$b_2 = 17.74$	
$R^2 = 0.9161$	$R^2 = 0.9289$	
$SEE = 30.92\%$	$SEE = 26.49\%$	
$BIAS = 0$	$BIAS = 0$	

Ratio combining Equation (65) and Equation (66) yields

$$\frac{E_{p_actual}}{t_{p_actual}} = \frac{b_3 \Phi_{[a,b]_actual}^{a_3}}{b_2 E_{p_actual}^{a_2}} \quad \therefore E_{p_actual}^{\left(\frac{a_2+1}{a_3}\right)} t_{p_actual}^{-\left(\frac{1}{a_3}\right)} = \frac{\Phi_{[a,b]_actual}}{\left(\frac{b_2}{b_3}\right)^{\left(\frac{1}{a_3}\right)}} \quad (68)$$

Instantiating the general form Equation (63) with the regression-derived Equation (68) implies the following assignments on each of φ_E , and φ_t :

$$\varphi_E = \frac{a_2 + 1}{a_3} \quad (69)$$

$$\varphi_t = -\left(\frac{1}{a_3}\right) \quad (70)$$

Substituting the Company X Avionics Projects data set results from (67) above into Equations (69) and (70) we get

$$\varphi_E = \frac{0.2681+1}{1.043} = 1.216 \quad (71)$$

$$\varphi_t = -\left(\frac{1}{1.043}\right) = -0.9589 \quad (72)$$

Instantiating the general form Equation (63) with the values in Equations (71) and (72) yields the Company X Avionics Projects data-set-specific estimating relationship:

$$E_p^{1.216} t_p^{-0.9589} = \frac{\Phi_{[a,b]}}{\bar{\delta}_{[a,b]}} \quad (73)$$

We can treat this estimating relationship (73) as a *factor* estimating relationship of the form $y = ax$ where

$$x \equiv \Phi_{[a,b]} \quad y \equiv E_p^{1.216} t_p^{-0.9589} \quad a \equiv \frac{1}{\bar{\delta}_{[a,b]}} \quad (74)$$

We now perform the MPE-ZPB general regression technique for *factor* estimating relationships in order to find the value for a and hence $\bar{\delta}_{[a,b]}$ that results in minimum SEE with no bias. The result of this regression is illustrated in **Figure 15** below.

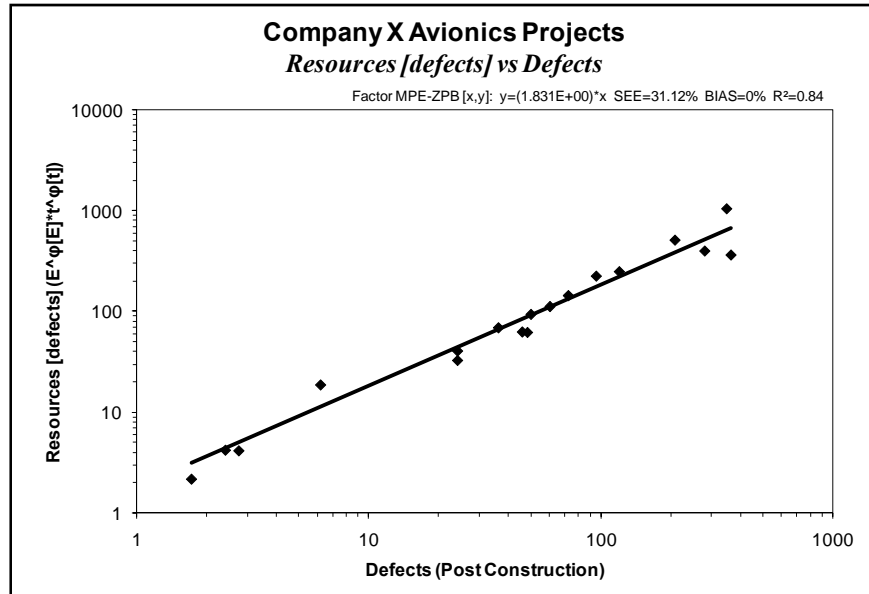


Figure 15. Resources versus Defects

Since **Figure 15** shows $a = 1.831$, it follows from the third definition in (74) that $\bar{\delta}_{[a,b]} = 1/a = 0.5460$. Therefore, the instantiated estimating relationship (73) for the Company X Avionics Projects data set is:

$$E_p^{1.216} t_p^{-0.9589} = \frac{\Phi_{[a,b]}}{0.5460} \quad (75)$$

Note in **Figure 15** the relatively good relationship quality values (SEE, BIAS, and R^2) which are indications of a reasonably good estimating relationship.

Specific Defect Vulnerability

We now introduce the notion of a project's specific defect vulnerability $\delta_{[a,b]}$ (lower-case Greek delta). Each instance of a software development process has a unique value for specific defect vulnerability within the context of a particular data set; i.e., a particular instantiation of Equation (63). Practically speaking, this means a value for specific efficiency of a process instance relates that process instance to other process instances in the particular historical data set for which Equation (63) has been instantiated. We write this instantiation and its various forms as:

$$\begin{aligned} & \left[E_p^{\varphi_E} t_p^{\varphi_t} = \frac{\Phi_{[a,b]}}{\delta_{[a,b]}} \right]_{\mathbf{A}}, \quad \left[E_p = \left(\frac{\Phi_{[a,b]}}{\delta_{[a,b]} t_p^{-\varphi_t}} \right)^{\left(\frac{1}{\varphi_E} \right)} \right]_{\mathbf{A}}, \quad \left[t_p = \left(\frac{\Phi_{[a,b]}}{\delta_{[a,b]} E_p^{\varphi_E}} \right)^{\left(\frac{1}{\varphi_t} \right)} \right]_{\mathbf{A}}, \\ & \left[\Phi_{[a,b]} = \delta_{[a,b]} E_p^{\varphi_E} t_p^{\varphi_t} \right]_{\mathbf{A}}, \text{ or } \left[\delta_{[a,b]} = \frac{\Phi_{[a,b]}}{E_p^{\varphi_E} t_p^{\varphi_t}} \right]_{\mathbf{A}} \quad [\varphi_t \leq 0] \end{aligned} \quad (76)$$

Once again we use square bracket symbols with a postfix subscript to mean “within the context of the data set named \mathbf{A} ”.

Effort-Duration Equation in Terms of Defects Produced

The equations in (76) are various forms of the *fundamental software defect propensity equation*. It describes the relationship between total process effort E_p and total process duration t_p as a function of the process's defect count $\Phi_{[a,b]}$ and the process's specific defect vulnerability $\delta_{[a,b]}$ within the context of some relevant historical data set \mathbf{A} .¹⁷

Solving for Defects as a Function of Duration and Independent of Effort

Substituting the solved-for-effort form of Equation (42) into Equation (76) yields

$$\left[\Phi_{[a,b]} = \delta_{[a,b]} \left(M t_p^\gamma \right)^{\varphi_E} t_p^{\varphi_t} \right]_{\mathbf{A}} \quad \therefore \left[\Phi_{[a,b]} = \delta_{[a,b]} M^{\varphi_E} t_p^{(\gamma\varphi_E + \varphi_t)} \right]_{\mathbf{A}} \quad (77)$$

Solving for Duration Independent of Effort

Solving Equation (77) for process duration t_p yields

¹⁷ The fairly general way in which defect count and defect vulnerability are defined allow for defect count to be specified within any range in a software development process. Note that the scale of defect vulnerability is always associated with the particular defect count range. Note also that it is possible to use defect density at some particular point in a development process in place of defect count as long as defect vulnerability is scaled accordingly.

$$\left[t_p = \left(\frac{\Phi_{[a,b]}}{\delta_{[a,b]} M^{\varphi_E}} \right)^{\left(\frac{1}{\gamma\varphi_E + \varphi_t} \right)} \right]_{\mathbf{A}} \quad (78)$$

Defects with Respect to Duration at Maximum Management Stress

Substituting Equation (76) into Equation (47) and solving for expected number of discovered defects $\Phi_{[a,b]}$ during a software development process yields

$$\left[\left(\frac{\Phi_{[a,b]}}{\delta_{[a,b]} t_p^{\varphi_t}} \right)^{\left(\frac{1}{\varphi_E} \right)} \leq M_{\max} t_p^{\gamma} \right]_{\mathbf{A}} \quad \therefore \left[\Phi_{[a,b]} \leq \delta_{[a,b]} M_{\max}^{\varphi_E} t_p^{\gamma(\varphi_E + \varphi_t)} \right]_{\mathbf{A}} \quad (79)$$

Defects with Respect to Duration at Minimum Management Stress

Substituting Equation (76) into Equation (52) and solving for expected number of discovered defects $\Phi_{[a,b]}$ during a software development process yields

$$\left[\left(\frac{\Phi_{[a,b]}}{\delta_{[a,b]} t_p^{\varphi_t}} \right)^{\left(\frac{1}{\varphi_E} \right)} \geq M_{\min} t_p^{\gamma} \right]_{\mathbf{A}} \quad \therefore \left[\Phi_{[a,b]} \geq \delta_{[a,b]} M_{\min}^{\varphi_E} t_p^{\gamma(\varphi_E + \varphi_t)} \right]_{\mathbf{A}} \quad (80)$$

9. MODEL EMULATION

Table 1 contains a list of several software estimation models with their corresponding instantiation values for the r2 Software Estimating Framework (r2SEF) equations (i.e., for the r2SEF parameters α_E , α_t , γ , φ_E , φ_t , M_{nom} , M_{max} , and M_{min}). Note that instantiating the generalized r2SEF equations with a parameter set from a particular model causes the r2SEF to emulate the behavior of that particular model. The table also includes several r2SEF instantiations that are directly based on the results of the previously-described regression process being applied to corresponding stratifications of collected historical data.

Table 1. r2SEF Equation Parameters for Various Models / Data Sets

<i>Model or Data Set</i>	$a[E]$	$a[t]$	γ	$\phi[E]$	$\phi[t]$	$M[nom] / (100\%t[nom])$	$M[max] / \%t[nom]$	$M[min] / \%t[nom]$
COCOMO 81 (Boehm, 1981) Organic Mode²¹	0.59	0.95	2.63	na	na	8.15E-3	2.76E-2 / 75%	1.11E-3 / 160%
COCOMO 81 (Boehm, 1981) Semi-Detached Mode²¹	0.58	0.89	2.86	na	na	4.76E-3	1.69E-2 / 75%	6.03E-4 / 160%
COCOMO 81 (Boehm, 1981) Embedded Mode²¹	0.57	0.83	3.13	na	na	2.51E-3	9.42E-3 / 75%	2.90E-4 / 160%
COCOMO II (Boehm, et al., 2000) Least Difficult^{18,21}	0.79	1.10	3.57	na	na	2.20E-4	9.16E-4 / 75%	2.14E-5 / 160%
COCOMO II (Boehm, et al., 2000) Most Difficult^{18,21}	0.54	0.82	2.91	na	na	1.36E-3	4.87E-3 / 75%	1.69E-4 / 160%
Seer (Jensen, 1983A), (Jensen, et al., 1983) Least Complex^{19,21}	0.50	1.00	3.00	na	na	4.62E-3	4.62E-3 / 100%	1.80E-3 / na
Seer (Jensen, 1983A), (Jensen, et al., 1983) Most Complex^{19,21}	0.50	1.00	3.00	na	na	5.78E-4	5.78E-4 / 100%	2.77E-4 / na
SLIM (Putnam, 1992), (Putnam, 1997) Business Applications^{20,21}	0.33	1.33	3.00	na	na	9.43E-2	5.92E-1 / 77%	2.44E-2 / 131%
SLIM (Putnam, 1992), (Putnam, 1997) Avionic Applications^{20,21}	0.33	1.33	3.00	na	na	1.15E-5	7.23E-5 / 77%	1.76E-6 / 131%
r2Estimating Database In-House SW on Standard HW (1)²¹	0.96	1.69	2.33	2.89	-2.02	5.38E-2	1.69E-1 / 76%	1.72E-2 / 132%
r2Estimating Database In-House SW on Custom HW (2)²¹	0.84	1.34	2.67	2.61	-1.90	6.36E-3	2.12E-2 / 75%	1.91E-3 / 133%
r2Estimating Database Product SW on Standard HW (3)²¹	0.86	1.34	2.84	1.72	-1.27	1.05E-2	4.49E-2 / 72%	2.47E-3 / 139%
r2Estimating Database Product SW on Custom HW (4)²¹	0.84	1.42	2.46	2.30	-1.64	1.28E-2	4.04E-2 / 76%	4.07E-3 / 132%
Int'l SW Benchmarking Stds Gp²² (ISBG) 3GL on Midrange Platform	0.72	1.14	2.70	3.77	-2.75	7.34E-3	1.97E-2 / 79%	2.73E-3 / 126%
Int'l SW Benchmarking Stds Gp²² (ISBSG) 4GL on Mainframe	0.46	0.76	2.56	1.55	-1.12	1.00E-2	3.79E-2 / 73%	2.67E-3 / 137%
r2Estimating Data Set Company X Avionics Projects²¹	0.55	0.75	3.73	1.22	-0.96	2.20E-5	5.91E-5 / 82%	8.15E-6 / 121%

¹⁸ Based on COCOMO II.1999 scale factor values. Foundation for SoftStar's Costar and Cost Xpert's Cost Xpert.

¹⁹ Foundation for Software Engineering Inc.'s Sage and Galorath Inc.'s SEER-SEM.

²⁰ Foundation for QSM's SLIM-Estimate & Estimate Express and Borland's Estimate Professional.

²¹ Size measured in effective source lines of code.

²² Size measured in International Function Points Users Group (IFPUG) Unadjusted Function Points (UFP).

10. SUMMARY AND CONCLUSION

Purpose and Scope Revisited

This paper documents the basis, assumptions, and derivations behind a set of general software effort, duration, and defects estimating relationships that are based on the notion that software development is the application of effort (labor) over some duration (period of elapsed calendar time) that produces a desired software product (size) and undesired byproducts (defects). The derivations, assumptions, and resulting model described by this paper establish relationships between size, efficiency, effort, duration, and defects and are collectively referred to as the *r2 Software Estimating Framework (r2SEF)*.

Areas for Further Study

The following are the author's observations and suggest possible opportunities for enhancing r2SEF capabilities:

- The behavior of most currently-available software project estimating models (e.g., COCOMO 81 (Boehm, 1981), COCOMO II (Boehm, et al., 2000), Seer (Jensen, 1983A), and SLIM (Putnam, 1980)) can be emulated by the r2SEF equations given a corresponding instantiation of the r2SEF variables α_E , α_t , γ , φ_E , φ_t , M_{\min} , M_{nom} , and M_{\max} . Instantiation values can be derived by algebraic manipulation of the particular model's estimating equation(s). We reasonably assume that the ability to emulate the behavior of multiple models plus the ability to estimate based on relevant historical data (both possible with the r2SEF) should provide increased overall estimate credibility.²³
- The primary independent variables *effective software size*, *specific efficiency*, and *defect vulnerability* are uncertain until project completion. This suggests a stochastic dimension to the problem; i.e., effective software size, specific efficiency, and defect vulnerability should all be treated as random variables. The results of this treatment should be distributions of possible outcomes for effort, duration, cost, and discovered defects with associated confidence probabilities.
- Labor (the motive force), effective software size (the evolving product), and discovered defects are each dynamic; i.e., they vary as the project progresses. This implies *estimates at completion* and *estimates to complete* are similarly dynamic. While the model described in this paper is adequate to estimate values for a total software development process, it begs for a time-range differential calculus approach to provide in-process estimates once the project is under way.

REFERENCES

Boehm, Barry W. 1981. *Software Engineering Economics*. Englewood Cliffs : Prentice-Hall, Inc., 1981. ISBN 0-13-822122-7.

²³ A recurring theme in recent International Society of Parametric Analysts conferences (particularly the "Renew Your Training" segments) has been the desire for multiple estimating model cross-checking in order to enhance estimate credibility and confidence.

- Boehm, Barry W., et al. 2000.** *Software Cost Estimation with COCOMO II*. Upper Saddle River : Prentice-Hall, Inc., 2000. ISBN 0-13-026692-2.
- Book, Stephen A. and Young, Philip H. 2006.** The Trouble with R^2 . *Journal of Parametrics*. Verdugo City, California, USA : International Society of Parametric Analysis, Summer 2006. Vol. XXV, 1, pp. 87-114. ISSN 1015-7891.
- Book, Stephen A. 2006.** Unbiased Percentage-Error CERs with Smaller Standard Errors. *The Journal of Cost Analysis & Management*. Alexandria, Virginia, USA : The Society of Cost Estimating and Analysis, Fall 2006. pp. 55-71. ISSN 0882-3871.
- Brooks, Frederick P., Jr. 1995.** *The Mythical Man-Month: Essays on Software Engineering*. Anniversary. Reading : Addison-Wesley Publishing Company, 1995. ISBN 0-201-83595-9.
- Jensen, Randall W. 1983B.** A Comparison of the Jensen and COCOMO Schedule and Cost Estimation Models. *Proceedings of the Fifth International Society of Parametric Analysts Conference*. St. Louis, Missouri, USA : The International Society of Parametric Analysts, April 26-28, 1983B.
- . **1983A.** An Improved Macrolevel Software Development Resource Estimation Model. *Proceedings of the Fifth International Society of Parametric Analysts Conference*. St. Louis, Missouri, USA : The International Society of Parametric Analysts, April 1983A. pp. 88-92.
- Jensen, Randall W. and Lucas, Suzanne. 1983.** Sensitivity Analysis of the Jensen Software Model. *Proceedings of the Fifth International Society of Parametric Analysts Conference*. St. Louis, Missouri, USA : The International Society of Parametric Analysts, April 1983.
- Jensen, Randall W., Putnam, Lawrence H., Sr. and Roetzheim, William. 2006.** Software Estimating Models: Three Viewpoints. *CrossTalk: The Journal of Defense Software Engineering*. Hill AFB, Utah, USA : Software Technology Support Center, February 2006. Vol. 19, 2.
- Musa, John D. 2004.** *Software Reliability Engineering: More Reliable Software Faster and Cheaper*. 2nd. Bloomington : AuthorHouse, 2004.
- Norden, Peter V. 1977.** Project Life Modeling: Background and Application of Life Cycle Curves. *Proceedings, Software Life Cycle Management Workshop*. Airlie, VA, USA : Sponsored by USACSC, 1977.
- Parkinson, Cyril Northcote. 1958.** *Parkinson's Law: The Pursuit of Progress*. London : John Murray, 1958. ISBN-10 071951049X, ISBN-13 978-0719510496.
- Putnam, Lawrence H. 1997.** *Industrial Strength Software: Effective Management Using Measurement*. Los Alamitos : IEEE Computer Society Press, 1997. ISBN 0-8186-7532-2.
- . **1992.** *Measures for Excellence: Reliable Software On Time, Within Budget*. Englewood Cliffs : Prentice Hall, Inc., 1992. ISBN 0-13-567694-0.
- . **1980.** *Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers*. New York City : IEEE Computer Society, 1980. IEEE Catalog No. EHO 165-1, Library of Congress No. 80-83083.
- Ross, Michael A. 2006.** Software Project Dynamics: A Productivity versus Staffing Approach. *Proceedings, ISPA 2006 Conference*. Seattle, Washington, USA : The International Society of Parametric Analysts, May 2006.

BIOGRAPHY

Michael A. Ross has over 32 years of experience in software engineering as a developer, manager, process champion, consultant, instructor, and award-winning international speaker. Mr. Ross is currently the President and CEO of r2Estimating, LLC. Mr. Ross's previous experience includes three years as Chief Engineer of Galorath Inc. (makers of the SEER suite of estimation tools), seven years with Quantitative Software Management, Inc. (makers of the SLIM suite of software estimating tools) where he was Vice President of Education Services, and 17 years with Honeywell Air Transport Systems (formerly Sperry Flight Systems) and 2 years with Tracor Aerospace where he developed or managed the development of embedded software for avionics systems installed various commercial airplanes and for expendable countermeasures systems installed in various military aircraft. He also co-founded Honeywell Air Transport Systems' SEPG, served as its focal for software project management process improvement, and served as a Honeywell corporate SEI CMM assessor. Mr. Ross did his undergraduate work at the United States Air Force Academy and Arizona State University, receiving a Bachelor of Science in Computer Engineering.

