Software Size Growth and Uncertainty

Both Affect Estimate Quality and Project Risk

Mike Ross
Galorath Incorporated
100 North Sepulveda Boulevard
Suite 1801
El Segundo, California 90245
(480) 488-8366 (phone) (480) 488-8420 (fax)
mross@galorath.com http://www.galorath.com

Abstract. Examination of currently-accepted software cost, schedule, and defect estimation algorithms reveals a common acknowledgment that estimated software size is the single most influential independent variable. Unfortunately, *"The most important business decisions about a software project are made at the time of minimum knowledge and maximum uncertainty."* This includes minimum knowledge and maximum uncertainty about a software product's effective size at the time when most estimating is done. Further complicating the issue of estimate quality, in the author's opinion, is the lack of a commonly-accepted taxonomy. This paper proposes definitions for and the relationship between two key attributes of software size estimates: ***growth*** and ***estimation process variability***, both being distributions, the dispersions of which decrease as a function of project progress.

Introduction

Purpose

This paper proposes definitions for and the relationship between two key attributes of software size estimates: ***growth*** and ***estimation process variability***, both being distributions, the dispersions of which decrease as a function of project progress.

Scope

This paper focuses on handling size growth and variability with cost and schedule estimation methods that employ parametric estimating techniques; however, in the author's opinion, these ideas could readily be extended to include any cost and schedule estimation method. The issues, assumptions, and propositions presented in this paper apply to all software development projects regardless of application domain or Software Development Life Cycle (SDLC) paradigm.

Background

Examination of currently-accepted software cost, schedule, and defect estimation algorithms reveals a common acknowledgment that ***assumed software size*** is the single most influential independent variable. It follows then that assumed software size has a significant impact on a given estimate's quality or usefulness. Unfortunately, *"The most important business decisions about a software project are made at the time of minimum knowledge and maximum uncertainty."*[5] This includes minimum knowledge and maximum uncertainty about a software product's effective size at the time when most estimating is done [5]. Further complicating the issue of estimate quality, in the author's opinion, is the lack of a commonly-accepted taxonomy.

Relevant Taxonomy and Context

Software Development Taxonomy

Terms defined:[1]

- ***Abstraction –*** A representation of an idea or concept expressed in a particular medium or language.
- ***Desire –*** A want or need.

[1] Term definitions extracted from [5].

- ***[Software] Requirements –*** An abstraction of a desire for which computer technology is thought to be a viable solution; the ***essence*** of a software product.
- ***Software –*** An abstraction of a desire expressed as instructions and data in a form that can be acted upon by a computer.
- ***Process –*** A set of actions or operations conducing to an end [4].
- ***Software Development Process –*** A generalized set of related activities that transform desires into software.
- ***Software [Development] Project –*** A specific instance of a software development process.
- ***Software Product –*** The primary (deliverable) result of a Software Development Project; the ***implementation*** of a software product.

Software Development Process Context

Figure 1 depicts the context of a software development process; i.e., how it interfaces with its environment. All instances of software development processes seek to transform software requirements into a software product. To accomplish this transformation, they consume energy in the form of labor (people doing work) from project initiation to project completion. Since no software development process is a *perfect machine*, it produces some amount of waste or entropy (undesired byproducts).

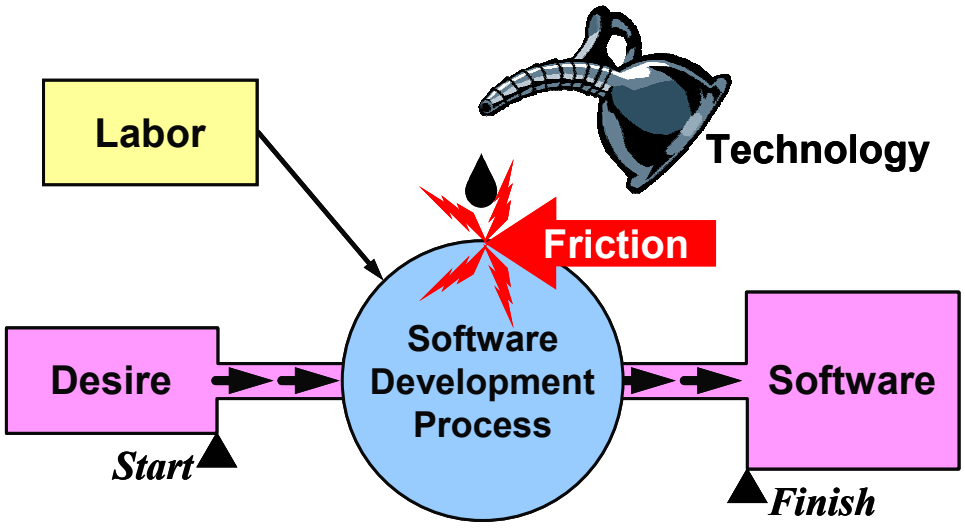

Figure 1: Software Development Context[2]

[2] Figure reprinted with permission from Michael A. ROSS Consulting & Training. All rights reserved.

Measuring the Software Development Process

The key to effectively and efficiently measuring the software development process is to pick measures that quantify the process's connections to its surrounding environment. Just about any core set of software development process measures will include the following:

- ***Size –*** An abstraction's mass, inertia, bigness (as it directly relates to the work that must be done).
- ***Duration –*** The elapsed calendar time between process initiation and process completion.
- ***Effort ➔ Cost, Staffing –*** People doing work during the software development process and their associated cost, over elapsed calendar time.
- ***Quality –*** Defect discovery and removal over elapsed calendar time.

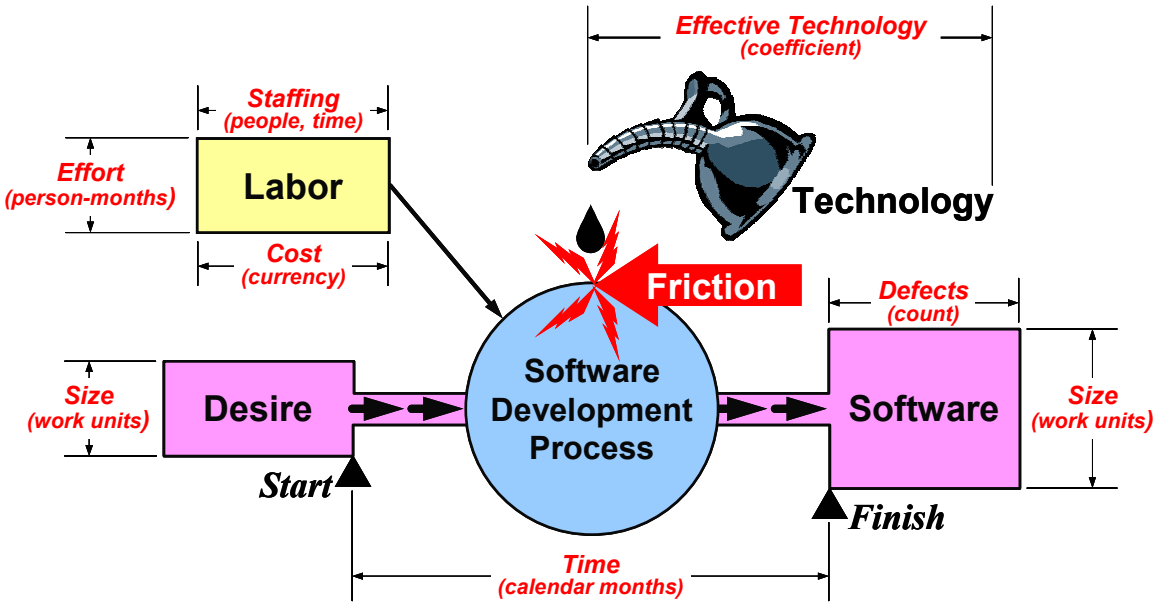

Figure 2: Software Development Process Context with Measurement[3]

A strong indication of the usefulness of these measures is the fact that they address the most frequently asked questions about software development projects:

- *How big will the product be when delivered?*
- *How long is it going to take?*
- *How many people will be needed and when?*

[3] Figure reprinted with permission from Michael A. ROSS Consulting & Training. All rights reserved.

- *How much will it cost?*
- *How reliable will the product be when delivered?*

Attributes of Estimated Software Size

Best Guess Size Estimate Defined

People generally think of software size as a count of the number of lines of code or the number of function points that will eventually be contained by a to-be-developed software product; this count representing some sort of *best guess* S_M . Natural and relevant questions should include, *"What considerations are included, what considerations are omitted, how confident are we in this best guess, how much uncertainty surrounds this best guess, and how might this best guess change over elapsed calendar time during the software development project?"*

First of all, since we are concerned about how this *best guess* might change over elapsed calendar time we change our best guess representation to be a function of progress $S_M(s)$ where progress s in this context is defined to be ***normalized earned value***; i.e., the project starts at $s = 0\%$ complete and finishes at $s = 100\%$ complete [6].

Second, the mention above of confidence and uncertainty and the use of the term *best guess* implies something that has a stochastic nature; i.e., there exists a set (a distribution) of numerous possible outcomes, our best guess being but one element of this distribution. We therefore postulate that a well-formed ***estimate*** is specified in terms of a selected probability distribution and its attributes. It seems reasonable then to assume that our ***best guess*** represents some sort of central tendency of its associated distribution. It also seems reasonable to assume that this distribution is continuous rather than discrete.[4] The next set of questions are, *"What kind of distribution are we talking about (Uniform, Normal, Beta, Triangular, etc.) , what are its attributes (location, dispersion, number of modes, skewness, kurtosis, etc.), and which form of central tendency does this* best guess *represent (mean, median, mode, other)?"*

Distribution Functions

There has been and continues to be much debate over which distribution function best represents estimated software size uncertainty. The leading candidates for this honor are (in no particular order):[5]

---

[4] The author acknowledges that software size, being a count of something, could be viewed as discrete rather than as continuous; however, since the range of possible outcomes is relatively large and the resolution of possible outcomes is relatively fine, the author chooses to view the distribution as continuous.

[5] Equations for these distributions and their attributes can be found at http://www.mathworld.wolfram.com.

- Normal (Gaussian) Distribution
- Bi-Normal Distribution[6]
- Triangular Distribution
- Beta (special case of a Weibull) Distribution

Location (Central Tendency)

We have already suggested that a *best guess* represents some sort of central tendency of its associated uncertainty distribution. Intuitively, of the three most common measures of central tendency (mean, median, and mode), it is the mode that seems to best represent the idea of a best guess. For example, I might say something like, "If I were to run this project many times (approaching infinity), I believe, based on what I know today, that a final size outcome of about 50,000 effective source statements would happen more times than any other final size outcome. In other words, approximately 50,000 effective source statements is thought to be the ***most likely*** or mode value of our size uncertainty distribution. It follows then that ***best guess*** and ***most likely*** are synonymous within the context of this discussion. Mathematically, this value is the global maximum of our size uncertainty distribution's Probability Density Function (PDF).

Dispersion

Of the three most common measures of dispersion; mean deviation, interquartile range, and standard deviation σ; the latter is almost invariably used by statisticians [2]. It is defined as the square root of the second central moment m_2 (variance) which, in turn, is defined as the average of the squared deviations from the mean.

Modes, Skewness and Kurtosis

We make the simplifying assumption that the distributions representing element-level contributors to uncertainty are unimodal; however, combinations of multiple distributions can yield multimodal distributions.

Skewness (asymmetry to the left or right) and kurtosis (peakedness) are measured as functions of the third and fourth central moments m_3 and m_4 respectively. Skewness is presented here since most estimators agree that size uncertainty distributions are asymmetric and tend to be right skewed (long tail on the right side of the PDF). Kurtosis does not seem to be too much of an issue at this point; however, as more data is

[6] Combines the left half of one Normal Distribution's PDF having a standard deviation σ_{Low} with the right half of another Normal Distribution's PDF having a standard deviation σ_{High} in order to model skewness using Normal Distribution math.

collected that can be used to relate size estimates with size outcomes, it may become more relevant to describing the ideal size uncertainty distribution.

Uncertainty Defined

We now introduce an emerging model that defines the notion of uncertainty as a function of variability, risk, and opportunity [3] and use the following conceptual model as a guide for describing size uncertainty[7].

$$U = \Sigma V + (\Sigma R - \Sigma O) \qquad \textbf{Eqn. 1}$$

where:

U	Uncertainty: a random variable representing the uncertainty about a particular value or metric, expressed as a probability distribution of possible outcomes.
V	Variability: a random variable representing the impact on the particular value or metric by an event or events that *will* occur (probability of 1), expressed as a probability distribution of possible outcomes.
R	Risk: a random variable representing the impact on the particular value or metric by a specific *unfavorable* event that may or may not occur (there exists some known probability of occurrence).
O	Opportunity: a random variable representing the impact on the particular value or metric by a specific *favorable* event that may or may not occur (there exists some known probability of occurrence).

Two Key Drivers of Software Size Estimates

Within the software estimation community and its serviced stakeholder organizations, there has, for better or worse, evolved two sometimes complementary and sometimes conflicting terms: ***size growth*** and ***size uncertainty***. We propose the following definitions for these two terms in the hope that some of the inherent conflict can be understood and minimized.

- ***Size Growth –*** *Variability* in the *baseline* estimated software size that results from a change in the common understanding of the required functionality and/or the context in which the software development project and its resultant software product exist. Note that we do not characterize size growth as a *risk* in our

[7] Use of the summation symbols in the conceptual relationship is intended to show aggregation of multiple contributing random variables, each of which may be multivariate in nature. The summation symbols do not imply that simple arithmetic addition is appropriate; it is more likely that simulation techniques such as Monte Carlo will be required to properly evaluate the contributors to uncertainty.

model. We assume that size growth *will* occur and that it embodies the impact of those events not yet known and specified (not yet characterized as risks/opportunities). Note also that we have narrowed the focus of the term size growth to one of a technological and programmatic nature. This definition implies the desirability to find some sort of growth factor function that can predict additional size and its associated uncertainty. As the project matures, we expect that risks and opportunities will become known and specified and, therefore, removed from consideration as a part of variability. This implies a desire to express growth factor as a function of progress on a given project.

- ***Size (Estimation) Uncertainty –*** *Variability* that results from the stochastic nature of human behavior and model behavior associated with the software size estimation process. Note that we have narrowed the focus of the term size uncertainty (hereinafter referred to as ***size estimation variability***) to one of process rather than to one that could be assumed to encompass all estimated size uncertainty; i.e., size growth and size estimation variability are mutually exclusive. Size estimation variability is described by a specific distribution (including its attributes) of possible software size impacts given some common understanding of the required functionality and of the context in which the software development project and its resultant software product exist.

Size Growth

Software project management would be a whole lot simpler if we knew, from the beginning, precisely how big the software will end up being. Issues of efficiency would then be the sole source of cost and schedule uncertainty. Unfortunately, static software size is not reality. A rare project experiences no requirements changes and no expansion of scope. This is a serious issue since variations in software size have the single largest influence on software development time, effort, cost, staffing, and the number of delivered defects [5].

We have previously stated that size growth stems from context volatility. In order to understand these notions of size growth and context volatility we must understand its source. If one were to solicit a list of things that cause software size to grow it might include some of the following:

- The customer doesn't know what he/she wants.
- The customer doesn't understand the problem.
- The mission has changed.
- The regulations that govern how this software should behave have changed.
- The vendor added a few extra features that he/she thought the customer would like.

- The vendor finished early so the customer and/or the vendor thought up a few things to add.

Analysis of the preceding list suggests the following possible organization of issues that influence software size growth:

- Operational Environment Volatility
- Essence (Requirements) Volatility
- Essence Understanding (Requirements Completeness and Correctness)
- Essence versus Implementation Correspondence

All of the preceding issues seem to fall into either the Technical or the Programmatic Risk[8] Driver categories per [1].

We earlier suggested the desire for a growth factor function that can predict additional size as a function of progress on a given project. Analysis of historical data collected by Galorath Incorporated suggests that this growth factor function $\mathrm{G}(s)$ is linear and is approximately:

$$\mathrm{G}(s) = -0.7s + 0.69 \qquad \textbf{Eqn. 2}$$

For example, if we assume that a project's normalized earned value when Software Requirements Analysis is complete (at Software Requirements Review or SRR) to be 11.8%, then[9]:

$$G_{SRR} = \mathrm{G}(11.8\%) = -0.7(11.8\%) + 0.69 = 0.61 \qquad \textbf{Eqn. 3}$$

Since we have already judged the issues impacting size growth to be technical or programmatic in nature, we assume that our size growth factor distribution can best be represented as a Triangular Distribution per [1] described by the parameter vector $\mathbf{G}(s)$:

$$\mathbf{G}(s) = \begin{bmatrix} L & M & H \end{bmatrix} = \begin{bmatrix} 0 & 0 & \mathrm{G}(s) \end{bmatrix} \qquad \textbf{Eqn. 4}$$

where L is the lowest conceivable growth factor value, M is the most likely (mode) growth factor value, and H is the highest conceivable growth factor value [7][10]. If our

[8] Note that this usage of the term Risk is not consistent with our uncertainty model but, rather, refers to terminology used in the cited document.

[9] Note that this example is consistent with the results documented in [7].

[10] The results of [7] suggest $0 = L < M < H$. To preserve consistency with our system of distributions we have transformed the results in [7] to force $M = L = 0$ while maintaining the total area under the Triangular Distribution PDF.

best guess of software size at SRR $S_{M\_SRR} = 50,000$ effective source statements (based on the current common understanding when Software Requirements Analysis is complete of the required functionality and of the current common understanding of the technology to be applied), then the size growth implication distribution is a Triangular Distribution described by the parameter vector $\mathbf{S_G}(s)$:

$$\mathbf{S_G}(s) = \mathrm{S_M}(s)\mathbf{G}(s) = \begin{bmatrix} 0 & 0 & \mathrm{S_M}(s)\mathrm{G}(s) \end{bmatrix} \quad \text{Eqn. 5}$$

$$\mathbf{S_{G\_SRR}} = \begin{bmatrix} 0 & 0 & S_{M\_SRR}(0.61) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 30,500 \end{bmatrix}$$

The Probability Density Function (PDF) for a Triangular Distribution is given by:

$$\mathrm{P_{Triangular}}(x) = \begin{cases} \dfrac{2(x-L)^2}{(H-L)(M-L)} & \text{for } \mathrm{x} \in [L, M] \\ 1 - \dfrac{2(H-x)^2}{(H-L)(H-M)} & \text{for } \mathrm{x} \in [M, H] \end{cases} \quad \text{Eqn. 6}$$

The Cumulative Distribution Function (CDF) for a Triangular Distribution is given by:

$$\mathrm{D_{Triangular}}(x) = \begin{cases} \dfrac{(x-L)^2}{(H-L)(M-L)} & \text{for } \mathrm{x} \in [L, M] \\ 1 - \dfrac{(H-x)^2}{(H-L)(H-M)} & \text{for } \mathrm{x} \in [M, H] \end{cases} \quad \text{Eqn. 7}$$

The Probability Density Function (PDF) and the Cumulative Distribution Function (CDF) for the Triangular Distribution described by $\mathbf{S_G}(s)$ are graphed below in Figure 3 and Figure 4 respectively.

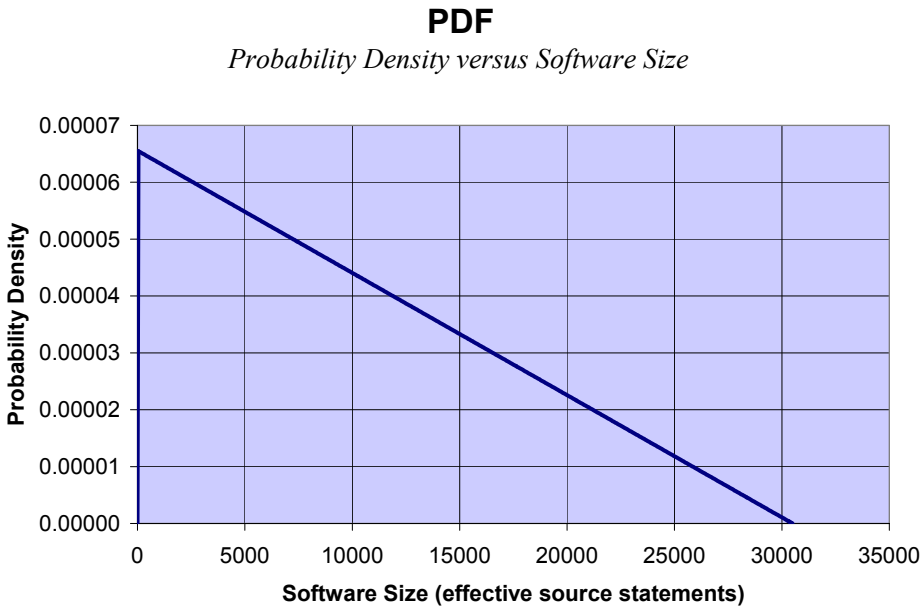

Figure 3: PDF of a Triangular Distribution Described by $\mathbf{S}_{\mathbf{G}}(s)$

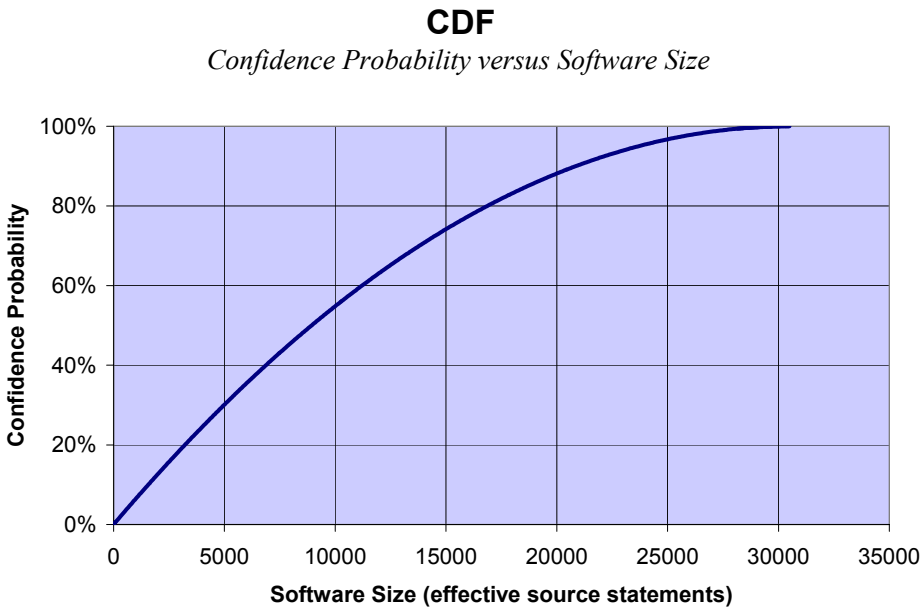

Figure 4: CDF of a Triangular Distribution Described by $\mathbf{S}_{\mathbf{G}}(s)$

Size Estimation Variability

Because, until project completion, software size must be estimated, it follows that software size is uncertain, regardless of whether or not we recognize size growth. The very nature of the word *estimate* implies uncertainty. We assume uncertainty in this context to mean that there exists some distribution (with specific attributes) of possible software size outcomes. Therefore, in order to quantify size estimation variability we must define this distribution and its attributes. Size estimation process and model variability are best represented by a Normal (Gaussian) Distribution [1]. We assume, based on [7], a $(\pm 30\%)\mathrm{S}_{\mathrm{M}}(s)$ conceivable range of this distribution; conceivable being defined as $\pm 2.33\sigma$ (between the 1st to the 99th percentiles).

Continuing our example situation, if our best guess of software size at SRR is $S_{M\_SRR} = 50,000$ effective source statements (based on the current common understanding when Software Requirements Analysis is complete of the required functionality and of the current common understanding of the technology to be applied), then the amount of additional estimated software size due to size estimation variability is a Normal Distribution described by the parameter vector $\mathbf{S}_{\mathbf{EV}}(s)$:

$$\mathbf{S}_{\mathbf{EV}}(s) = [\mu \quad \sigma] = \left[0 \quad \frac{(30\%)\mathrm{S}_{\mathrm{M}}(s)}{(2)(2.33)}\right] \qquad \text{Eqn. 8}$$

$$\mathbf{S}_{\mathbf{EV}} = \left[0 \quad \frac{(30\%)S_{M\_SRR}}{(2)(2.33)}\right] = [0 \quad 3,219]$$

where μ is the arithmetic mean of the distribution (in effective source statements) and σ is the standard deviation of the distribution (also in effective source statements). Note that we specify μ to be 0 in order to center the distribution about 0, the left half (negative) representing size decrease due to estimation variability, the right half (positive) representing size increase due to estimation variability.

The Probability Density Function (PDF) for a Normal (Gaussian) distribution is given by:

$$\mathrm{P}_{\mathrm{Normal}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad \text{for } x \in (-\infty, \infty) \qquad \text{Eqn. 9}$$

There exists no closed form representation of the Cumulative Distribution Function (CDF) for a Normal (Gaussian) Distribution; however, Microsoft® Excel contains a built-in approximation function for this purpose. Additionally, a reasonable second order polynomial approximation is given by:

$$\mathrm{D}_{\mathrm{Normal}}(x) \approx \begin{cases} 0.01 \text{ for } x \in (-\infty, \mu - 2.33\sigma] \\ 0.0903\left(\frac{x-\mu}{\sigma}\right)^2 + 0.4207\left(\frac{x-\mu}{\sigma}\right) + 0.5 \text{ for } x \in [\mu - 2.33\sigma, \mu) \\ 0.5 \text{ for } x = \mu \\ -0.0903\left(\frac{x-\mu}{\sigma}\right)^2 + 0.4207\left(\frac{x-\mu}{\sigma}\right) + 0.5 \text{ for } x \in (\mu, \mu + 2.33\sigma] \\ 0.99 \text{ for } x \in [\mu + 2.33\sigma, \infty) \end{cases} \quad \textbf{Eqn. 10}$$

The Probability Density Function (PDF) and the Cumulative Distribution Function (CDF) for the Normal Distribution described by $\mathbf{S}_{\mathbf{EV}}(s)$ are graphed below in Figure 5 and Figure 6 respectively.

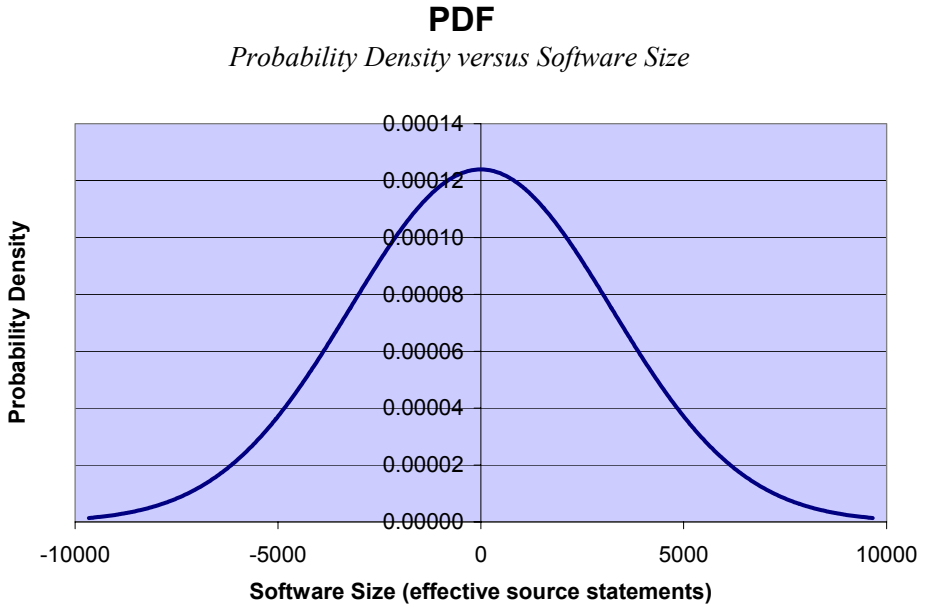

Figure 5: PDF of a Normal Distribution Described by $\mathbf{S}_{\mathbf{U}}(s)$

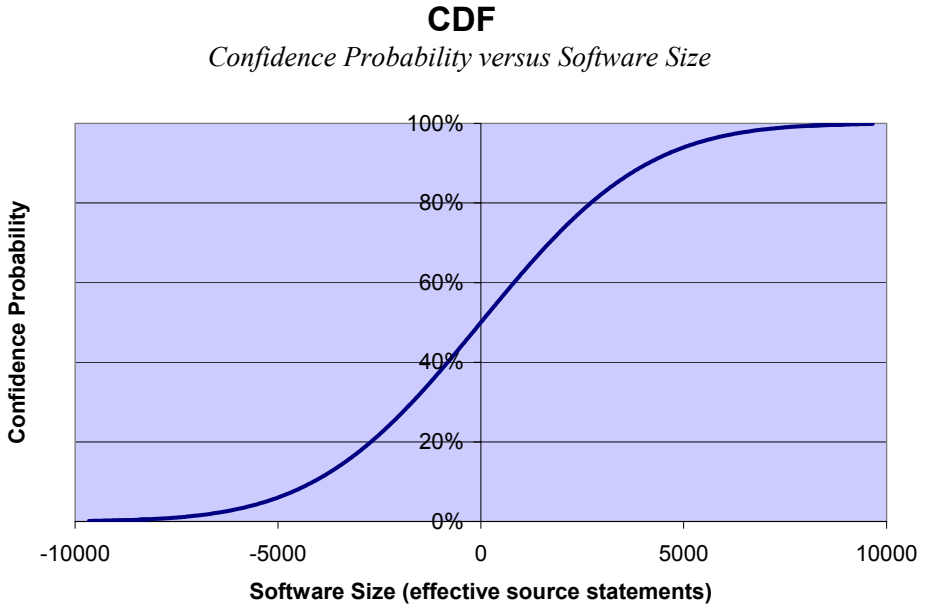

Figure 6: CDF of a Normal Distribution Described by $\mathbf{S}_{\mathrm{U}}(s)$

Combining Size Growth and Size Estimation Variability

Based on our definitions of size growth and of size estimation variability, we can sum our *best guess* size estimate $\mathrm{S}_{\mathrm{M}}(s)$, our size growth, a Triangular Distribution described by $\mathbf{S}_{\mathbf{G}}(s)$, and our size estimation variability, a Normal Distribution described by $\mathbf{S}_{\mathbf{EV}}(s)$; the result being our estimated size distribution of unknown type and described by the parameter vector $\mathbf{S}(s)=[\mu \quad \sigma]$, μ being its arithmetic mean and σ being its standard deviation. In order to solve for μ and σ we can take advantage of two statistical theorems as described in [2] and [1], one for expectation E that yields μ and one for variance V that yields σ^2. Each can be applied to a series of independently[11] distributed random variables X_i:

$$\mathrm{E}\left(\sum_{i=1}^{n} X_i\right)=\sum_{i=1}^{n}\mathrm{E}(X_i) \qquad \textbf{Eqn. 11}$$

and

$$\mathrm{V}\left(\sum_{i=1}^{n} X_i\right)=\sum_{i=1}^{n}\mathrm{V}(X_i) \qquad \textbf{Eqn. 12}$$

[11] Independence is a necessary prerequisite for the variance theorem; however, it is not a necessary prerequisite for the expectation (mean) theorem.

Given our series of independent random variables $\mathrm{S_M}(s)$, $\mathbf{S_G}(s)$, $\mathbf{S_{EV}}(s)$:

$$\mu_{\mathrm{S_M}(s)} = \mathrm{S_M}(s)$$
$$\mu_{\mathbf{S_G}(s)} = \frac{0+0+\mathrm{S_M}(s)\mathrm{G}(s)}{3} = \frac{\mathrm{S_M}(s)\mathrm{G}(s)}{3}$$
$$\mu_{\mathbf{S_{EV}}(s)} = 0$$
$$\therefore \mu_{\mathbf{S}(s)} = \mathrm{S_M}(s) + \frac{\mathrm{S_M}(s)\mathrm{G}(s)}{3} = \frac{\mathrm{S_M}(s)\left(\mathrm{G}(s)+3\right)}{3}$$

Eqn. 13

and

$$\sigma_{\mathrm{S_M}(s)} = 0$$
$$\sigma_{\mathbf{S_G}(s)} = \sqrt{\frac{L^2+M^2+H^2-LH-LM-MH}{18}} = \sqrt{\frac{\left(\mathrm{S_M}(s)\mathrm{G}(s)\right)^2}{18}}$$
$$\sigma_{\mathbf{S_{EV}}(s)} = \frac{(30\%)\mathrm{S_M}(s)}{(2)(2.33)}$$
$$\therefore \sigma_{\mathbf{S}(s)} = \sqrt{\frac{\left(\mathrm{S_M}(s)\mathrm{G}(s)\right)^2}{18} + \left(\frac{(30\%)\mathrm{S_M}(s)}{(2)(2.33)}\right)^2}$$

Eqn. 14

Continuing our example size estimate taken at SRR where our *best guess* $S_{M\_SRR} = 50,000$ and our size growth factor $G_{SRR} = 0.61$:

$$\mathbf{S_{SRR}} = \left[\frac{S_{M\_SRR}\left(G_{SRR}+3\right)}{3} \quad \sqrt{\frac{\left(S_{M\_SRR}G_{SRR}\right)^2}{18} + \left(\frac{(30\%)S_{M\_SRR}}{(2)(2.33)}\right)^2}\right]$$
$$\mathbf{S_{SRR}} = \left[\frac{50,000(0.61+3)}{3} \quad \sqrt{\frac{\left((50,000)(0.61)\right)^2}{18} + \left(\frac{(30\%)(50,000)}{(2)(2.33)}\right)^2}\right]$$
$$\therefore \mathbf{S_{SRR}} = [60{,}167 \quad 7{,}877]$$

Eqn. 15

We now know the mean or expected value (60,167 effective source statements) and standard deviation (7,877 effective source statements) of the statistical sum of the three contributors to our size estimate. If this were one small component in a larger whole consisting of many components, then we could take advantage of the Central Limit Theorem *"which states that the sum of a large number of independent random variables will be approximately normally distributed almost regardless of their*

individual distributions."[2] Unfortunately, we don't have a large number of independent random variables in this example; thus, if we wish to extract probability / confidence information from our size estimate distribution, we are left with a problem that is best solved by a calculator or a software product that uses simulation.

Summary and Conclusion

Purpose Revisited

This paper proposed definitions for and the relationship between two key attributes of software size estimates: ***growth*** and ***estimation process variability***, both being distributions, the dispersions of which decrease as a function of project progress.

Areas for Further Study

The following are suggestions for furthering the discussion of software size growth and uncertainty:

- Collect more (and more continuous) size estimation data and use it to strengthen size growth factor functions.
- Investigate making the conceivable range of the size estimation variability distribution be a function of project progress (i.e., factor in the notion of project maturity and associated learning).
- Investigate relevant methods and techniques, avoiding simulation, that provide probability / confidence information from distributions that are the statistical sum of a small number of constituent distributions (i.e., the resulting distribution is unlikely to be a Normal Distribution).

References

[1] Book, S.A., "Cost-Risk Computations by Hand Calculator"; *Proc. SCEA National Conference & Educational Workshop*, The Society of Cost Estimating and Analysis, Scottsdale, AZ, June 2002.

[2] Bulmer, M.G., *Principles of Statistics*, Dover Publications, Inc., New York, NY, 1979.

[3] Hutchings, Christopher, "Risk is not a four letter word!", Proposed Seminar: Galorath Inc., El Segundo, CA, 2005.

[4] Mish, F. (Editor in Chief), *Merriam-Webster's Collegiate Dictionary*, Tenth Edition, Merriam-Webster, Incorporated, Springfield, MA, 1999.

[5] Ross, M., “Managing Software Size,” *Proc. Joint ISPA / SCEA 2003 Conference*, The International Society of Parametric Analysts and The Society of Cost Estimating and Analysis, Orlando, FL, June 2003.

[6] Ross, M., “Parametric Project Monitoring and Control,” *Proc. Joint ISPA / SCEA 2005 Conference*, The International Society of Parametric Analysts and The Society of Cost Estimating and Analysis, Denver, CO, June 2005.

[7] Tarbet, D., “Software Cost/Schedule Estimation: Code Growth,” Internal White Paper: Galorath Inc., El Segundo, CA, 2002.

Biography

Michael A. Ross has over 30 years of practical experience in software engineering as a developer, manager, process champion, consultant, instructor, and award-winning international speaker.

Mr. Ross is currently the Chief Engineer of Galorath Incorporated, makers of the SEER suite of estimation tools, where, for the past three years, he has been responsible for the advancement and realization of the technology aspects of Galorath's mission and vision.

Prior to joining Galorath, Mr. Ross was Vice President of Education Services for Quantitative Software Management, Inc. (makers of the SLIM suite of software estimating tools). He was responsible for the development and delivery of all QSM training. During his seven-year tenure with QSM, he served as one of the company's primary consultants and analysts working with Fortune 500 companies and government agencies in the areas of software measurement, sizing, estimating, tracking, forecasting, and benchmarking.

Mr. Ross, during 17 years with Honeywell Air Transport Systems (formerly Sperry Flight Systems) and 2 years with Tracor Aerospace, developed or managed the development of embedded software for avionics systems installed various commercial airplanes including the Boeing 737-500, 757, 767, 777, the Douglas MD-11, the Lockheed L1011-500, the British Aerospace BAe-146, the Airbus A320; and for expendable countermeasures systems installed in various military aircraft and missiles. He also co-founded Honeywell Air Transport Systems' process improvement team (later to become its 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. He is a member of the Project Management Institute (PMI), the Institute of Electrical and Electronics Engineers (IEEE), the International Function Points Users Group (IFPUG), the International Society of Parametric Analysts (ISPA), the Society of Cost Estimating and Analysis (SCEA), the Arizona Software Association, and the Phoenix area Software Process Improvement Network.