Can the Parr Curve Help with Manpower Distribution and Resource Estimation Problems?*

Victor R. Basili and John Beane

Department of Computer Science, University of Maryland

This paper analyzes the resource utilization curve developed by Parr. The curve is compared with several other curves, including the Rayleigh curve, a parabola, and a trapezoid, with respect to how well they fit manpower utilization. The evaluation is performed for several projects developed in the Software Engineering Laboratory of the 6–12 man-year variety. The conclusion drawn is that the Parr curve can be made to fit the data better than the other curves. However, because of the noise in the data, it is difficult to confirm the shape of the manpower distribution from the data alone and therefore difficult to validate any particular model. Also, since the parameters used in the curve are not easily calculable or estimable from known data, the curve is not effective for resource estimation.

INTRODUCTION

Two important problems face the project manager at the beginning of the software development process. First, the manager must estimate the basic quantities of concern: the cost of the system, the duration of the project, and the size of the development team. The techniques for estimating cost have received more attention, but perhaps the crucial quantity in determining the success of the project is the schedule. The initial estimate of duration is often incapable of being changed because many contracts now include deadlines, with financial penalties for missing them. The mistake of underestimating the project duration can have dire effects. Brooks [1] points out that the common practice of increasing the production team when a project is late can involve more trouble than benefit. Putnam [2] has presented a model that illustrates in a quantitative way that the tradeoff of manpower for time is not free. Further, there are limits as to how far a schedule can be shortened depending on the difficulty of the development effort. Scheduling decisions cannot be made arbitrarily as a matter of convenience.

Once the estimates of the project cost, schedule, and team size are made, the next problem facing the project manager is how to distribute the total effort (represented by cost and team size) over the course of the project schedule. This problem has been solved for some large-scale projects using the Putnam model. Previous work has been done at the Software Engineering Laboratory (SEL) at the University of Maryland to decide whether the early prototype of the Putnam model, designed for large projects, could be applied to small- and medium-scale projects as well. The results have been mixed. To understand better why this model is less effective, it is important to consider the characteristics of the SEL environment.

The Software Engineering Laboratory collects and analyzes the data from projects built by Computer Sciences Corporation for the Goddard Space Flight Center (NASA). The goals of the laboratory are

1. to provide management with a mechanism to monitor the status of current projects;
2. to collect data to study the software process, to find what parameters can be isolated (understood), and to build measures incorporating these parameters; and
3. to compare the effects of various techniques upon system development [3, p. 116].

The seven projects used in this study are all attitude determination packages for satellite systems. They range in size from 50,900 to 111,900 lines of delivered source code (including comments). The code is mostly written in FORTRAN, with a small portion written in assembly language. The cumulative effort varied from 3

*Research supported in part by National Aeronautics and Space Administration grant NSG-5123 to the University of Maryland.

Address correspondence to Victor R. Basili, Department of Computer Science, University of Maryland, College Park, Maryland 20742.

0164-1212/81/010059-11$02.75

Table 1. Statistics About the Projects

	Project						
	1	2	3	4	5	6	7
Total lines[a]	111.9	55.2	50.9	75.4	75.4	85.4	89.5
New lines[a]	84.7	44.0	45.3	49.3	20.1	76.9	62.0
Developed lines[a]	90.2	46.2	46.5	54.5	31.1	78.6	67.5
Effort (man-months)	115.7	95.9	78.9	90.7	39.6	98.6	98.3
Duration (months)	15.8	11.5	13.2	12.5	8.7	17.4	14.3
Average staff size	7.3	8.3	6.0	7.3	4.6	5.7	6.9

[a]In thousands of lines of source code.

to 10 man-years and lasted 9–18 months. A complete set of statistics is given in Table 1. All these projects fall into the medium-size range. It should be noted that new projects are often upgraded versions or other modifications of existing systems. The implications of this are twofold. Many projects can use some of the design and even the code of previous systems, and the organization as a whole has great experience with the application area (since for them the problems are well defined). In contrast, large-scale projects can be characterized as needing more than "2 years of development time, 50 man-years of effort or greater, and a hierarchical management structure of several layers" [2, p. 302].

The work described in this paper is a continuation of the studies of Basili and Zelkowitz [3] and Mapp [4]. Their analysis can be divided into two parts. In the first part, they asked whether the Putnam model could be used as an estimation tool. They took the Rayleigh equation (which is the central part of the Putnam model) and derived a relationship among three important quantities of the software process: the total effort K, in man-hours; the number of weeks T_a until acceptance testing; and the maximum staffing Y_d, in man-hours per week. During the requirements phase the contractor estimates each quantity, and the data are reported on the general project summary (GPS) form. Given any two of these estimates, a prediction of the third quantity can be based on the Rayleigh equation. The most interesting quantity (as we mentioned before) is the project duration, since NASA budgets fix the total resources each year in advance, and the contractor assigns a fixed number of people to the effort. The predictions of the time to acceptance testing were quite good when compared with the actual dates, in contrast to the original estimates [3]. The GPS estimates were consistently too low. Thus, the Rayleigh equation provides a check to ensure the project duration is not underestimated.

The second part of the analysis considered how well the Rayleigh curve fit the shape of actual staffing data over time. The Rayleigh equation can be rewritten in the form of a line for the variables y/t and t^2. After this, a line can be fit to the transformed data using linear regression. When Basili and Zelkowitz tried this, they found that the resulting curves did not follow the general shape of the data. At a glance it was clear that other curves could have fit the data better, and the quantities (T_a and K) taken from the fitted curve were unreliable as predictors.

Tom Mapp carried the curve-fitting comparison one step further. He tested four curves (a parabola, a trapezoid, a horizontal line, and the Rayleigh curve). The measure of comparison was the average squared (vertical) distance between the curve and the data points. Mapp used two techniques to find a best fitting curve for the Rayleigh equation and the parabola, the linear technique and a blind search. The blind search systematically sampled values from a bounded portion of the parameter space. The parameter set that yielded the best error measure became the center of a smaller "search box." When a new iteration failed to improve the error measure, the search was terminated. In every case the search technique produced a better fit than the linear method. The best curve, determined from the rank orderings of the final error measures for four projects, was the parabola. The study concluded that the Putnam model was successful at predicting milestones but did not fit the staffing data for our environment.

In this paper we analyze a new dynamic staffing model proposed by Parr [5]. To begin we review the general differences between static distribution models, based on a work breakdown structure, and dynamic distribution models, derived from a theory about problem solving. Then we examine the two theoretical (dynamic) models of Parr and Putnam to illuminate the critical assumptions that shape the curves and how they differ. Finally, we consider the claims made by dynamic staffing models and attempt to validate them using data from our environment. In particular, how well do dynamic models actually fit our data, and can the Parr model be used to predict project duration (in a manner similar to the Putnam model)?

DISTRIBUTION MODELS, CLAIMS, AND LIMITATIONS

Static distribution models start with a general description of the activities that constitute the software development process for a given environment. Then the tasks that comprise each activity are grouped under the right development phase. The important step is to distribute the total effort across these tasks. Each task is given a percentage, based on the skill and intuition of the model builder, and any available accounting data (assuming it reflects a similar environment of software methodologies). The percentages can be divided further to take into account different types of personnel (managers, analysts, programmers, or librarians) and different levels of experience that will be needed for the job. An example of a work breakdown structure is given by Wolverton [6]. When the functional specifications are complete, some adjustments will often be made in the baseline percentages to reflect the special demands of the particular project.

The static model provides a detailed staffing algorithm once an estimate is made for the total effort and the project schedule. A staffing algorithm is an excellent tool to monitor the progress of the project. First, the manager can use the algorithm to anticipate the fluctuations in his manpower needs before it becomes a problem. Because hiring new people is difficult, an increase in staffing requires some warning. In addition, the milestones of the schedule work like a sequence of checkpoints. When a milestone is not met, the work breakdown highlights which tasks are in trouble and possibly need more people.

The impulse to add more people to a late task is a natural one, but it can cause the task to be even later. It is unexpected phenomena like this that motivated the development of dynamic staffing models based on a theory of how we build software. Dynamic models propose assumptions to help explain such behavior. For example, adding more people to a working group increases the number of communication lines. The job of keeping people informed is more costly in terms of time and effort. Also, new people require an adjustment period, to get acquainted with the task, and will probably divert some of the energy of the original team members. On the other hand, there are inherent constraints in the software problem itself. A partial ordering of the individual subproblems exists, which limits the amount of work that can be done at the same time (and the number of people who can be effectively used). All these assumptions could help to explain why a part of the effort that is applied to a task does not result in any actual progress.

Dynamic distribution models are not alternatives to static staffing methods but instead complement them. Dynamic models deal with the kind of macroscopic quantities that are needed to use a static model. Used alone, dynamic methods lack the necessary detail to be an effective staffing algorithm, but they provide a glimpse of the overall picture. Dynamic models can estimate critical milestones in the schedule. They can serve as a means of checking the reasonableness of the percentages in the work breakdown structure with regard to the weekly effort expended. One aspect of the Putnam model even shows how a change in the specifications at any time will effect the schedule, the total effort, and the size of the code. Next we consider how a theoretical model derives an effort distribution.

SHAPING A DISTRIBUTION CURVE; THE THEORIES

The dynamic models rely on a set of assumptions describing how we build software. A software project consists of solving a bounded set of problems. Each problem represents some aspect of the design or implementation for which a decision must be made between possible alternatives. We are concerned with the constraints describing when effort can be effectively applied to solving these problems. The dynamic models of Parr and Putnam agree that the reason for a decrease in effort at the end of the project is an exhaustion of the problem set. This decrease reflects the nature of the debugging task. "Debugging is '99 percent complete' most of the time" [1, p. 154]. We do not have adequate measures to decide when a project is done, or even how much longer it will take. System debugging does not lend itself to people working in parallel, because errors tend to be discovered sequentially. The correction of one error uncovers another. It requires a small number of people working over an extended period to complete this phase of the project. Both models use an exponential tail to describe this situation.

The models disagree over what constrains the distribution curve at the start of the project. Putnam argues that progress can only be made once the development team becomes familiar with the problem and the proposed method of solution. The familiarization (or learning) rate that fit his data best was a straight line whose slope is determined by management's staffing decisions. However, there are practical limits as to how fast the buildup can be. First, it is hard to obtain new people, whether by hiring them or transferring them from other projects. Second, there is an organizational limit on the number of people that can communicate and work effectively with one another. The rate of the initial buildup also has implications for the duration of the project. Given a fixed amount of total effort, the faster the rise in staffing, the shorter the schedule. The size and the complexity of the problem fixes a minimum time period for the project duration.

Parr feels that these considerations focus on the wrong issue. It is important to understand how the problem itself limits the effort that can be effectively applied before considering those management decisions that are economically motivated. In that way we can examine an optimal staffing plan for the problem without concern for practical considerations, whose impact can be analyzed separately. Parr suggests that there are dependencies between the problems, so that the work on a particular task cannot begin until others have been completed. These dependencies form a partial ordering. At any given time a subset of the unsolved problems exists, called the *visible* set, consisting of those that are ready to be worked on; in other words, all of the tasks on which a visible problem depends have been solved. The size of the visible set is the quantity that management is aware of, and (provided there are enough people to work on all of the visible problems at once) it should determine an optimal level of staffing.

The Rayleigh curve rises in a straight line from the origin to a rounded peak and then falls in an exponential tail. The formula for the Rayleigh curve is

$y'(t) = 2Kate^{-at^2}$,

where

y' is the effort in man-hours expended per week,

t the time in weeks,

K the development effort in man-hours (the area under the curve), and

a a shaping parameter.

a determines the slope of the rising portion of the curve and equals $-\frac{1}{2}t_d^2$, where t_d is the point of maximum manning. When K represents the life-cycle cost of the system, t_d corresponds roughly to the development time up to acceptance testing. This equation makes explicit the inverse relationship between the learning rate and the project duration. In the discussion to follow K represents the development cost (that is, we assume no maintenance or enhancement).

The normalized Parr curve is bell shaped (symmetric about the origin) and trails off exponentially on both sides. The formula for the Parr curve is

$y'(t) = \alpha K'Ae^{-\alpha\gamma t}/(1 + Ae^{\alpha\gamma t})^{1+1/\gamma}$,

where

y' is the effort in man-hours expended per week,

t the time in weeks,

K' the development effort in man-hours,

A the horizontal shift factor,

α the time normalization factor, and

γ a structuring index.

γ measures the extent to which formal structured techniques are a part of the development process. When $\gamma > 1$ the peak of the curve is skewed to the right. The purpose of structured programming is to delay implementation decisions through the use of abstraction and information hiding. These practices result in more time being spent in the specification and design phases so that the coding and testing phases will be simpler (partitioned in such a way as to minimize interfaces and allow maximum parallel effort). α stretches or shrinks the time variable onto a unitless scale, and A shifts the curve horizontally along the time axis.

K' has a different interpretation than K. Putnam assumed that each project has an official starting date prior to which no money or people are budgeted. This was reasonable in his environment, because a separate organization handled the preliminary work. If the starting date is $t = 0$, then the Rayleigh curve must pass through the origin. Parr argues that there is always some effort expended before the official project start. This early work serves the important function of defining the problem set that represents the desired software system (through feasibility studies and requirements analysis), establishing its internal structure (through functional specifications), and solving the top-level problems (through preliminary design) on which all the others depend. The positive effect of these activities is to expand the visible set of unsolved problems that is available to be worked on at the project start. General experience with the application area, specific design, or even code contributes to the structuring process. Thus the Parr curve does not pass through the origin. If K_0 represents the initial effort (the area under the curve before $t = 0$), then $K' = K + K_0$, and K_0, along with the shaping parameters α and A, determines $y'(0)$, the level of initial staffing. More will be said concerning this relationship in the section on estimating the Parr curve parameters.

A second difference between the two curves relates to the degree of flexibility in positioning the point of maximum staffing. As we mentioned previously, this point is determined by the slope of the initial rise (t_d being directly related to a). A large slope implies an early peak and a rapid fall in staffing. Conversely, a small slope implies a late peak with little or no decrease before acceptance testing. Basili and Zelkowitz comment that for medium-sized projects "the resource curve is mostly a step function." The Rayleigh curve seems inappropriate to this shape, and "variations in the basic curve so that it is flatter in its mid-range" are being investigated. The Parr curve is one possibility. In the next section we present our results from the curve fitting comparison between Parr and Rayleigh.

DO DYNAMIC MODELS ACTUALLY FIT OUR DATA?

We considered two paradigms in our analysis of curve fitting. First, we wanted to be able, given the data on the effort associated with a project, to tell what staffing algorithm (actually, what distribution curve) had been used in its development. The curve we were looking for would fit the data better than the others. In particular, we set up a comparison between the two theoretical curves (those based on a software theory) of Parr and Putnam, as well as two control curves with reasonable characteristics (initial rise to a peak and then a fall). If a theoretical curve did better, then this would tend approximately to validate the assumptions made by the model.

The possibility remained that our data contained so much noise that none of the curves would stand out as better than the rest. In that case, a second paradigm is to be considered: Given the effort data and a staffing algorithm, we can decide whether in fact the algorithm had been used for the development process. This paradigm was tried with a staffing rule of thumb supplied by the contractor.

The noise comes from several sources. Since the data is weekly, weeks that contain holidays involve less total effort. If one member of the team is sick or on vacation, there is a drop in the weekly effort. If there is a problem, several people will work overtime and create a rise in the weekly effort. This is especially true when the average staff size is between 4.6 and 8.3 on the projects studied. To eliminate the noise we tried smoothing the data and combining four-week intervals. Unfortunately, this had little effect.

The nature of our effort data made it necessary to use an error measure for comparing curve fits; often a visual comparison was not possible. We chose the same measure as Mapp had used, namely the standard error

$$SE = \sum_{t=1}^{N} \frac{[f(t) - x(t)]^2}{N},$$

where

N is the number of data points,

$x(t)$ the effort in man-hours expended in week t (the data), and

$f(t)$ the distribution curve evaluated at t.

The technique to minimize SE involved two routines that were borrowed from the IMSL (International Math and Statistics Library) package. The first, ZXMIN, uses a quasi-Newtonian method to calculate the minimum of a user-supplied function. The routine requires an initial guess for the parameters of the function and then in an iterative fashion generates a new set of parameters from the old set. In order to avoid converging to a local minimum, we started the search at a large number of points. The second routine, XSRCH, did the selection of the initial parameter sets from a search box of reasonable values fixed by the user. There are two conditions that control the termination of the search process, the number of iterations and the differences between consecutive values of the parameters. One more twist was added to force the search to stay within the initial search box.

Table 2. An Initial Curve-Fitting Comparison: Rayleigh vs Parr Using SE

	Project				
	1	2	3	4	Average
Parr curve	939	2356	2204	2928	2204
Rayleigh curve	3379	4501	3926	4758	3926

In the first curve-fitting comparison the Parr curve did much better than the Rayleigh curve on each of four projects (see Table 2). The average of the standard error across the projects showed that the Parr fits were nearly twice as good (2204 to 3926). But these results were not very interesting, because the Parr curve has four parameters and the Rayleigh curve has only two. We had the suspicion that any curve with four parameters would have done better than one with two. In order to make our comparisons meaningful we decided to examine curves with an equal number of parameters. In the next test we therefore removed a parameter from the Parr curve and added one to the Rayleigh curve. We also included two more control curves. If a control did as well or better than the curves based on a theoretical distribution model, we could conclude that for our environment there was nothing special about the curve shapes.

The choice of parameter to remove from the Parr curve was decided once we noticed that the parameter A could change by several orders of magnitude and the curve still maintain a good fit (see Appendix A). The other parameters seemed to compensate in such a way as to suggest the power relationship

$$A = f(\alpha,\gamma) = 2^{100\alpha\gamma-2}.$$

A second possibility for removing A was to set it equal to a constant. For the projects in question the constant 1000 produced good fits. The results from comparing these three-parameter variations to the basic Parr curve (see Table 3) show that a constant A does almost as well as an extra parameter.

To increase the flexibility of the Rayleigh curve we

Table 3. Three- and Four-Parameter Fits

	Project				
	1	2	3	4	Average
$A = 1000$	939	2356	2913	3060	2317
$A = f(\alpha\gamma)$	991	7096	2704	3303	3524
Four-parameter curve	939	2356	2592	2928	2204

Table 4. Three-Parameter Curve Fit

	Project							
	1	2	3	4	5	6	7	Average
SE								
Parr	938	2356	2913	3060	2367	2072	2864	2367
Rayleigh	1837	3853	2517	3713	2980	2580	3350	3007
Parabola	1115	2298	2505	3497	3078	2508	3203	2601
Trapezoid	974	2265	2588	2981	2517	2722	3066	2445
Rank ordering								
Parr	1	3	4	2	1	1	1	13
Rayleigh	4	4	2	4	4	3	4	25
Parabola	3	2	1	3	3	2	3	17
Trapezoid	2	1	3	1	2	4	2	15

incorporated a horizontal shift factor so that the curve was not forced to pass through the origin. We borrowed the control curves from the Mapp study, a parabola and a "trapezoid" consisting of three straight lines. The exact formulation for these curves is given in Appendix B. The three-parameter curve fitting comparison (including graphs and tables of SE values and rankings) is presented in Table 4 and Figures 1–7. With respect to average SE values, the Rayleigh fits had improved, but Parr still did better. It is reassuring to note that the Parr curve also did better than either of the controls in terms of average fit and total rankings. However, we question whether the differences between these evaluations are large enough to be significant.

We concluded that the large fluctuations in the data for projects of this size effectively covered up the inherent shape of the effort distribution, if in fact such a shape exists as Putnam and Parr suggest. Consider our experience: When fitting the four-parameter Parr curve, the noise in the data allowed us to change the characteristics of the curve considerably (as reflected in the parameter A) and still retain a reasonable fit. Also, three very different curves, the trapezoid, the parabola, and Parr did equally well in the three-parameter comparison. The very data seem to contradict the assumption that there are constraints on staffing, whether introduced by management concerns or by problem dependencies. It therefore appears that we cannot tell what kind of software environment or staffing algorithm was used given only the effort data for a given project.

The second paradigm attempts to validate a staffing

Figure 1. Three-parameter fit of man-hours to weeks for project 1. Key: – – –, Parr curve; · · ·, Putnam curve; – · –, parabola; ———, trapezoid.

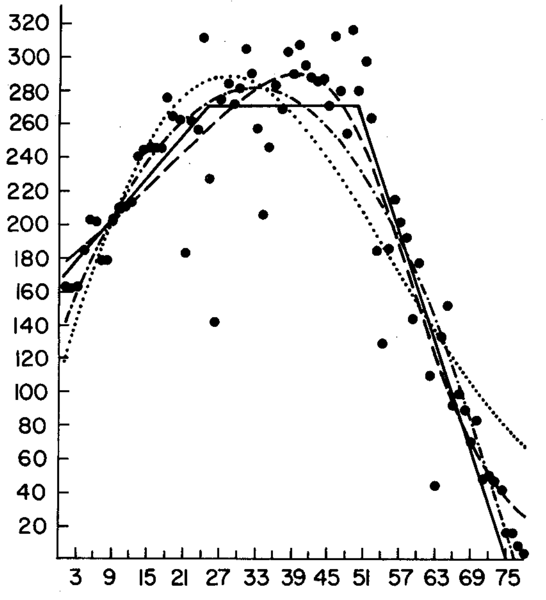
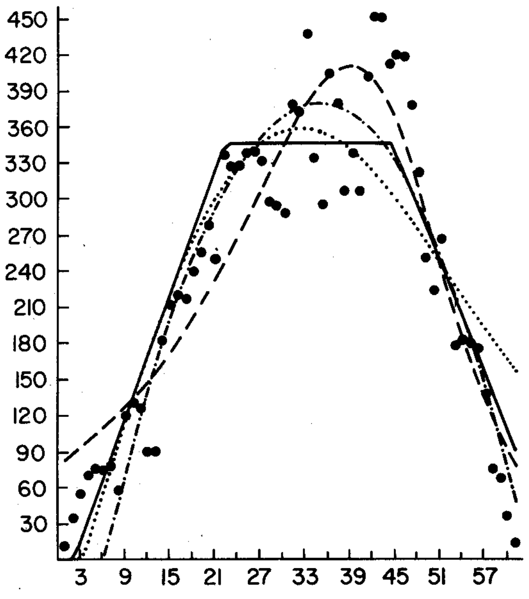


Figure 2. Three-parameter fit of man-hours to weeks for project 2. For key see Figure 1.



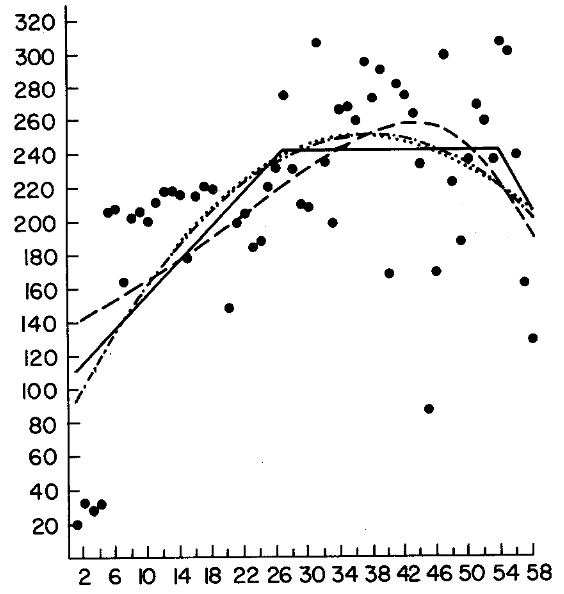


Figure 3. Three-parameter fit of man-hours to weeks for project 3. For key see Figure 1.

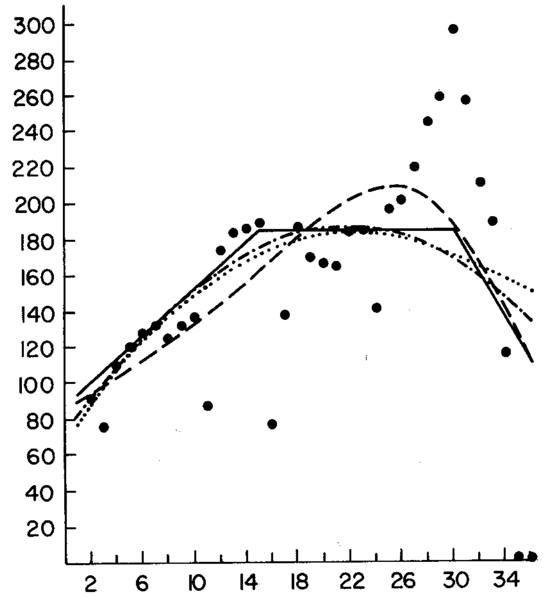


Figure 5. Three-parameter fit of man-hours to weeks for project 5. For key see Figure 1.

algorithm given both the algorithm and the effort data. The rule of thumb for staffing that the contractor tries to follow is this:

1. At the start of the project assign from one-half to three-quarters of full staffing (because of a lack of early funding and problems in finding available people).
2. At the end of the design phase, plus-or-minus a month, build to full staffing.
3. During the coding phase maintain full staffing.
4. During the testing phase, (a) if on schedule, decrease manning as appropriate; (b) if behind, work overtime; and (c) if there are late changes to the user requirements, increase manning by an additional one-third.

Figure 4. Three-parameter fit of man-hours to weeks for project 4. For key see Figure 1.

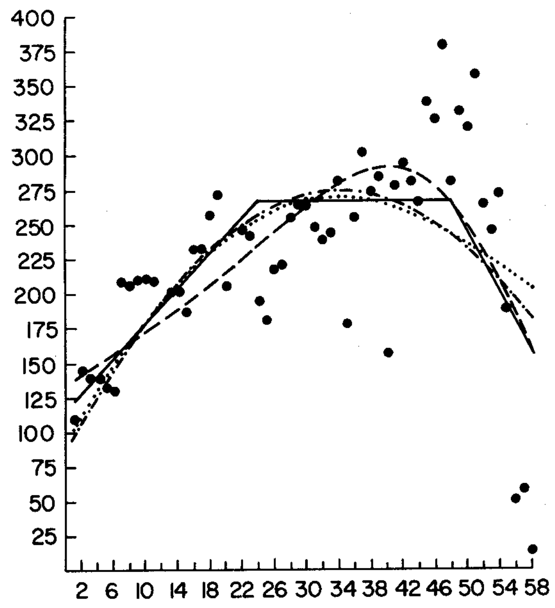
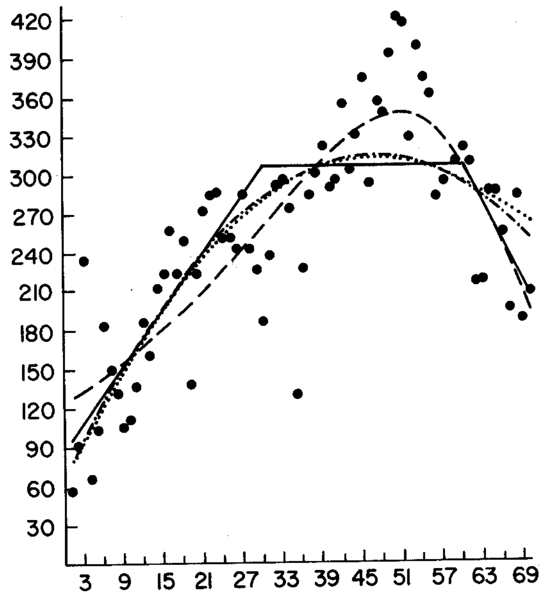


Figure 6. Three-parameter fit of man-hours to weeks for project 6. For key see Figure 1.



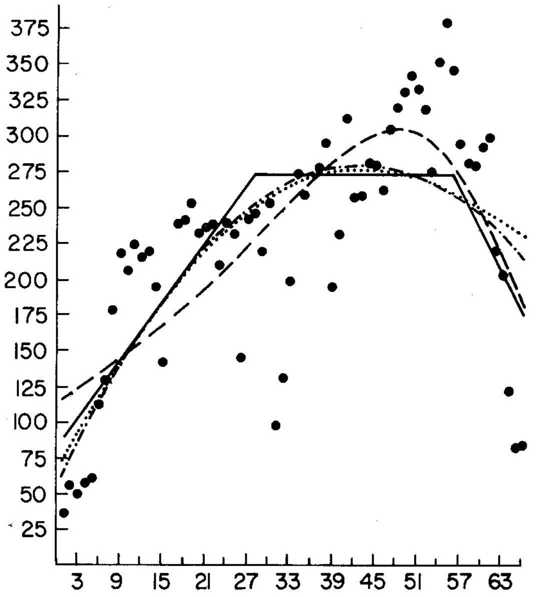


Figure 7. Three-parameter fit of man-hours to weeks for project 7. For key see Figure 1.

These guidelines convey the impression that management has considerable flexibility in terms of staffing to handle problems when they arise. The reason new people can be brought in at the end and contribute almost immediately is the similarity between the projects. Often a new system is a modification or enhancement of an old system, as seen in the percentage of existing code that gets reused, so little time is wasted in becoming familiar with a new system.

Next we wanted to check whether the contractor's rule of thumb was being used in practice. Since the algorithm is expressed as a step function, we needed to calculate averages for the phases concerned. In particular, we chose an 8-week period from the middle of each phase, which we thought could be representative in the sense that expenditures for those weeks took on roughly median values for the phase as a whole. We gave added weight to periods in which expenditures were more or less stable, whether the period fell in the middle or not. The averages computed from these periods are approximate. By selecting a different period the numbers can be changed by as much as 25 man-hours. Table 5 shows the numbers for five projects. If we assume that the numbers for the coding phase represent full staffing, they correspond fairly well to the algorithm. The average percentage of weekly design expenditures was 67% that of full staffing taken across all projects, a number midway within the range quoted in the algorithm for design, and various projects seemed to exhibit behavior that fit well into the three options for step 4. Project 1 decreased the level of staffing, projects 2 and 3 remained at the same level, and projects 4 and 5 increased staffing by one-third during the testing phase. The conclusion, then, is that we cannot reject the assumption that the contractor's algorithm is being used as a rough guideline by the managers. However, if we plotted the contractors algorithm as we did the other curves, the SE would be no better.

Table 5. Verifying the Contractor's Algorithm

Project	Design[a]	Code[a]	Test[a]	Design/code	Test/code
1	197	270	220	0.73	0.81
2	94	364	360	0.26	0.99
3	202	244	253	0.83	1.04
4	205	245	326	0.84	1.33
5	114	170	224	0.67	1.32
Average:				0.67	1.10

[a]Averages for 8-week periods (in man-hours per week).

REALITY AND THE PARR PARAMETERS

One of the benefits we mentioned in connection with theoretical effort distribution models was the ability to predict important milestones in the development schedule. However, before turning to the prediction problem, we wanted to be sure that the curves we fit to the effort data resulted in dates that were close to the actual milestones. This was another way of validating the model. In particular, we looked at the time period from the official start of the project through acceptance testing (t_a), or roughly the duration of the development activity. Solving the Parr equation for t_a yields

$$t_a = -\frac{1}{\alpha\gamma}\frac{\ln(K'/K'_a)^\gamma - 1}{1000},$$

where K'_a is the cumulative effort up to acceptance testing. (In the SEL environment we have estimated $K_a = 0.88K$. To convert these parameters into their Parr equivalents we added K_0 to each, so $K'_a = 0.88\,(K' - K_0) + K_0$.) The derivation of the equation for t_a is given in Appendix C, and the results of the comparison between the real values of t_a and the values taken from the curve are presented in Table 6. Except for project 1, the predicted values are within 5% of the true values. The bad estimate can be explained (at least in part) by

Table 6

	Project			
	1	2	3	4
t_a (estimated)	57.7	52.1	61.9	55.8
t_a (actual)	47.8	54.5	60.8	53.4

our not beginning to collect data for project 1 until well into the design phase.

Now that we had some confidence that the Parr curve fit the data about as well as any other and that the milestones calculated from the parameters were fairly accurate, we turned to the difficult problem of estimating the curve's parameters. K' is the total development effort, and a model like that of Walston and Felix [7] or Boehm [8] could be used to obtain an estimate. γ describes the degree to which formal (structured) methodologies are part of the development process. This parameter like Putnam's "technology constant," can be calibrated based on the techniques in use for a given environment. The remaining parameters A and α determine K_0, the amount of effort expended before the official start of the project; α converts the time variable onto a unitless scale, and A shifts the curve horizontally. Both depend on the duration of the project, the unit of measure for the time variable, and what part of the curve (how much of the exponential tails) is to be used to fit the data. α was introduced in the derivation of the Parr curve as a proportionality constant relating the rate of expending effort to the amount of work to be done at a given moment. If it took one person one time unit to solve each problem in the development effort, then α would be 1. It can be shown that when A is large (for our environment A was on the order of 1000) α is approximately equal to the y intercept of the distribution curve, $y'(0)$, divided by K_0 (see Appendix D). Our energies were therefore directed to finding a way to estimate these two quantities.

During the early stages of the project when the effort estimation and distribution models are needed, some initial effort data are available (Table 7). We attempted to use the data to estimate the y intercept of the distribution curve. Table 8 compares the y intercept with the first data point, the average of the first five data points, and the average of the first ten data points. For projects 1 and 4 the initial effort point is a close approximation to the y intercept, and after 5 weeks the estimate is even better. However, the averages of the initial effort for the other two projects do not begin to approximate the y intercept until after the tenth week.

This approach seemed to fail because of the nature of our data. Most projects have trouble finding enough people at the start, and many of the people who are assigned to the project begin by working part time, so even when the new effort allows a good deal of parallel activity at its inception, the problem of short staffing often squanders the opportunity for a fast start. As a result, the optimal manpower rates as reflected by the Parr curve are not met by the projects with early staffing problems. Using the initial effort data for a project is thus not an acceptable method of estimating $y'(0)$.

We also tried to estimate K_0, combining those activities that help define the problem before the official start of the project. Such activities as feasibility studies, requirements analysis, the use of existing design and code, and the general experience of the contractor with the application area partition the problem so that more people can work in parallel at an early stage in the development. For a rough estimate we chose the percentage of existing code because it was easy to get the data. Table 7 shows that comparison to K_0. Much of the variation in K_0 is not explained by this factor alone. To improve the comparison other factors (such as those mentioned above) will have to be incorporated into the estimate.

Thus we were unsuccessful in using the Parr curve as a predictor of such milestones as completion date since we were unable to associate the equation parameters with any data that would be easy to estimate at the onset of the project.

Table 7

	Project			
	1	2	3	4
Old code (%)	24.2	20.5	11.0	34.5
K_0/K'	40.6	8.9	33.5	28.0

Table 8

	Project			
	1	2	3	4
y intercept	179	79	138	136
Effort for first week	163	11	20	110
Average for first 5 weeks	175	49	64	133
Average for first 10 weeks	185	71	130	163

CONCLUSION

Dynamic distribution models offer an estimation tool for critical software quantities such as project duration, as well as a set of assumptions to enhance our understanding of problem solving behavior. To provide some assurance that these assumptions are valid for a given environment, we proposed fitting the effort distribution curve to actual data. In previous studies the Rayleigh curve proved to be a good method for estimating project duration, but for small- to medium-scale projects it did not fit the data. Thus there is some doubt whether the model can be used to monitor the expenditures of effort for an environment. This paper analyzed the applicability of an alternative curve developed by Parr. In a comparison of four curves with an equal number of pa-

Table 9. Parr Curve Fit for *A* Constrained (Project 1)

	<20	<25	<40	<100	<200	<1000
$100\alpha\gamma$	7.2	7.6	7.9	9.1	10.1	12.4
$f(\alpha\gamma)$	35.5	53.8	59.3	134.4	265.0	1296.1
SE	1382	1338	1218	1091	1019	939

Table 10. Parr Curve Fit for *A* Constrained (Project 4)

	<20	<50	<100	<500	<1000	<3000
$100\alpha\gamma$	5.9	8.1	9.1	11.7	12.9	14.9
$f(\alpha\gamma)$	14.8	69.1	133.4	831.7	1910.6	7643.4
SE	4123	3425	3341	3145	3060	2928

rameters, the Parr curve produced the best fits. However, the results and the data tend to contradict rather than support the theory on which the curve is based. The data imply that management has the ability to change staffing almost arbitrarily to meet the short-term needs of the project. The fluctuations in the data imply that a natural shape for the effort distribution may not exist for projects of this size.

The Parr model must do more than fit effort data. Although a fitted curve produced an accurate prediction of project duration, the crucial question is whether we can discover a way to estimate the Parr parameters themselves. Our efforts have not been fruitful up to this point, but other options of study remain. For now the Parr curve has limited usefulness as an effort distribution and resource estimation tool.

APPENDIX A. Eliminating a Parameter from the Parr Curve by a Power Relationship

The Parr curve is flexible enough to allow A to change by several orders of magnitude and still retain a reasonable fit. As A is increased, the product of α and γ increased in a similar way so as to suggest the following power relationship:

$$A = f(\alpha\gamma) = 2^{100\alpha\gamma - 2}.$$

This function was deduced by noticing the change in the parameters for various fits where the value of A was constrained to be less than some bound. In the cases of projects 1 and 4, the bounds were consistently set too low, so that by increasing the bound the fit improved. The results of contrasting A are shown in Tables 9 and 10.

APPENDIX B. Three-Parameter Curves

Three curves were compared with the Parr curve in the three-parameter test. The variable t is time measured in weeks.

For the Rayleigh (Putnam) curve,

$$y'(t) = 2Ka(t + t_s)e^{-a(t+t_s)^2},$$

where

y' is the effort in man-hours expended per week,

a a shaping parameter related to the time when the curve reaches a maximum,

K the total effort for the project in man-hours, and

t_s a horizontal shift factor to remove the origin constraint.

For the parabola,

$$\begin{aligned} y'(t) &= at^2 + bt + c && \text{if } y'(t) > 0,\\ &= 0 && \text{if } y'(t) \leq 0. \end{aligned}$$

The parameters a, b, and c do not have any special meaning from an estimating point of view.

For the trapezoid,

$$\begin{aligned} y'(t) &= [3(H - y_s)/T]t + y_s && \text{if } 0 \leq t < T/3,\\ &= H && \text{if } T/3 \leq t < 2T/3,\\ &= 3H - (3H/T)t && \text{if } 2T/3 \leq t < T,\\ &= 0 && \text{if } t > T, \end{aligned}$$

where

y' is the effort in man-hours expended each week,

H the maximum manning for the project in man-hours,

T an arbitrary time period in weeks, and

y_s the manning at project start.

The measure for goodness of fit used in comparing the curves (see Table 4) was

$$\mathrm{SE} = \sum_{t=1}^{N} \frac{[f(t) - x(t)]^2}{N},$$

where

N is the number of data points (the project duration in weeks),

$f(t)$ the value of the curve at t, and

$x(t)$ the actual effort in man-hours expended during week t.

APPENDIX C. Verifying the Time to Acceptance Testing as Predicted by the Parr Curve

The integral of the Parr curve is an equation for the cumulative effort expended up to time t. We solve this equation for the time t_a, the time to acceptance testing.

$$Y'(t) = K'(1 + Ae^{-\alpha\gamma t})^{-1/\gamma}$$

Substitute $Y'(t_a) = K_a'$ and $A = 1000$:

$$\begin{aligned} K_a'/K' &= (1 + 1000e^{-\alpha\gamma t_a})^{-1/\gamma},\\ 1000e^{-\alpha\gamma t_a} &= (K'/K_a')^\gamma - 1,\\ -\alpha\gamma t_a &= \frac{\ln[(K'/K_a')^\gamma - 1]}{1000},\\ t_a &= -\frac{1}{\alpha\gamma}\,\frac{\ln (K'/K_a')^\gamma - 1}{1000}. \end{aligned}$$

Table 6 compares the estimate of t_a using a fitted three-parameter Parr curve with the actual data (the values for t_a are in weeks).

APPENDIX D. Searching for a Physical Interpretation of the Parameter α

α can be expressed in terms of $y'(0)$ and K_0. The three-parameter Parr curve evaluated at $t = 0$ is

$$y'(0) = 1000\alpha K'/1001^{1+1/\gamma}.$$

Using the cumulative distribution curve, the initial effort K_0 is

$$K_0 = Y'(0) = K'1001^{-1/\gamma}.$$

Solving for K' and substituting back into the first equation leaves $1000\alpha K_0/1001$. α is approximately equal to $y'(0)/K_0$.

Table 6 shows a comparison of the percentage of existing code that is reused in the new system to the percentage of initial effort as computed by the Parr curve. Table 7 compares the effort data at the beginning of the projects with the y intercept of the three-parameter Parr curve. Three measures for the effort data are used: the first data point, the average of the first five data points, and the average of the first ten. All the numbers are in units of man-hours per week.

REFERENCES

1. F. Brooks, *The Mythical Man-Month; Essays on Software Engineering,* Addison-Wesley, Reading, Massachusetts, 1975
2. L. Putnam, A General Empirical Solution to the Macro Software Sizing and Estimating Problem, *IEEE Trans. Software Eng.* 4 (4) (1978).
3. V. Basili and M. Zelkowitz, Analyzing Medium-Scale Software Development, *Proc. 3rd Int. Conf. Software Eng.*, Atlanta, Georgia, May 1978
4. T. Mapp, Applicability of the Rayleigh Curve to the SEL Environment, unpublished, University of Maryland, 1978
5. F. Parr, An Alternative to the Rayleigh Curve Model for Software Development Effort, *IEEE Trans. Software Eng.* (May 1980).
6. R. Wolverton, The Cost of Developing Large Scale Software, *IEEE Trans. Comput.* 23 (6) (1974).
7. C. Walston and C. Felix, A Method of Programming Measurement and Estimation, *IBM Syst. J.* 16 (1) (1977).
8. B. Boehm, *Software Economics*, Prentice Hall, Englewood Cliffs, N.J. 1981.