# Managing and Predicting the Costs of Real-Time Software

ROGER D. H. WARBURTON

*Abstract*—**The Putnam model can be used to predict and manage software development projects. It is a management tool that takes, as input, easily obtained manpower data and produces cost and schedule estimates. This paper examines data from two real-time software projects and analyzes the applicability of the Putnam model. We propose a variation of the model which more reliably follows the staffing curve of real-time software applications. A critical analysis of the assumptions is presented and the parameters are reinterpreted so that they reflect the environment of embedded applications. Two projects are analyzed from actual data. It is shown how management decisions are reflected in the model. Even erratic and incomplete data can yield valuable conclusions. It is also shown that the model is appropriate to software developed with modern practices. We show how valuable management information can be obtained by laying out the data in a systematic manner.**

*Index Terms*—**Cost prediction, Putnam model, Rayleigh curve, real-time software, software management.**

## I. Introduction

THE Putnam model is a software management tool that can be used to predict costs and delivery schedules.[1] The two most important management parameters are indeed the projected cost of the job and the delivery data. The Putnam model uses easily obtained manpower data, presenting it in a manner that is easily understood. Furthermore, the two interesting management parameters are clearly represented. The effects of delays and cost overruns are easily visualized, and their effects quantified.[2]

Manuscript received September 10, 1981; revised January 5, 1983. This work was supported in part by Raytheon Submarine Signal Division Research and Development Funding.

The author is with JAYCOR, Middletown, RI 02840.

[1]For a complete description of the Putnam model see, for example, L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," *IEEE Trans. Software Eng.*, vol. SE-4, no. 4, July 1978.

[2]For a review of software cost modeling and estimation techniques, the reader is referred to B. W. Boehm, *Software Engineering Economics*. New York: Prentice-Hall, 1981.

The Putnam model uses the manpower data in different ways, to obtain different types of predictions. Historical data on completed projects provides a calibration database for an organization. These data can be used to generate likely cost and schedule estimates on future projects. For example, during a proposal bid effort, the Putnam model can be used in a risk analysis to assess the increased costs of a shorter schedule.

The second use of the Putnam model is in predicting costs and schedules while a job is progressing. The model provides realistic projections that can be compared with actual milestones.

In this paper we discuss the application of the model to real-time software applications. The standard model is found to be deficient in certain areas, but a reinterpretation of some of the parameters has allowed us to use it to describe the real-time software development process.

### A. The Putnam Model and Its Assumptions

The first step in using the Putnam model is to plot the number of people working on a project as a function of time. Fig. 1 is an example of such a plot. The curve shows that the people are assigned to the project, starting with relatively few during software design. The manpower reaches a peak and falls off, the decrease in manpower during testing is slower than the earlier build up.

The first assumption of the Putnam model is that all software projects follow this type of curve, and that the curve is characterized by a Rayleigh distribution. The Putnam model then proceeds as follows: the entire manpower curve is supposed to be Rayleigh shaped, with several subcycles. Early requirements work is not included in the manpower curve (represented in Fig. 1 as a dashed line).

The first major subcycle is the detail design and code cycle, which is also supposed to follow a Rayleigh curve. The manpower is supposed to peak around the time of the first system

0098-5589/83/0900-0562$01.00 

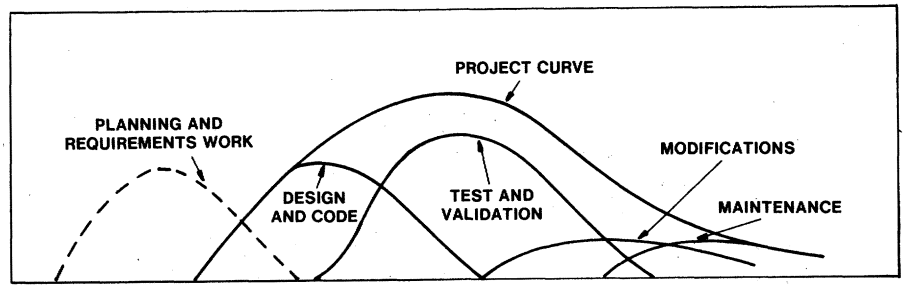



Fig. 1. Manpower curve subcycles. This figure describes the subcycles that are assumed to make up the software life cycle. Planning and requirements work is not usually included in the Putnam model.

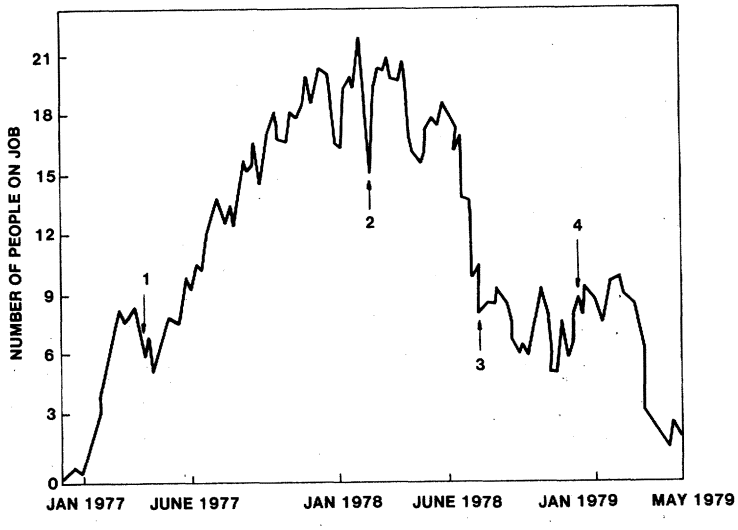



Fig. 2. This figure shows the manpower data for a real-time software project. Even without knowing the Putnam model, predictions can easily be made from this type of curve. Significant fluctuations reflect management problems. (1) Customer funding problems. (2) Winter holidays and snowstorms. (3) Psychological trap of running out of money. (4) Subsequent build-up to correct previous problem.

delivery, which occurs at approximately 40 percent of total budget. This is often followed by a long series of modifications and upgrades which constitute 60 percent of the total project expenditure. This peaking of manpower at the prototype delivery stage is a weak assumption.

In data processing applications, it is often true that the life of a program proceeds in this way. For example, the author observed a bank statement program when it first became operational—it was usable and yet far from its final form. The bank statement formats changed from month to month. The statement format grew in complexity and sophistication for about six months, and was then stable for several months. Since then it has undergone several upgrades as banking conditions have changed.

This is to be contrasted with a tactical embedded program in which the software has to meet strict specifications or delivery will not be accepted. Delivery thus marks the end of a project. By the time a tactical system reaches delivery, the work is complete and there will only be a very small maintenance staff remaining. However, in an embedded software development project, we still observe the manpower to increase, peak, and tail off during testing. The challenge was to modify (or more accurately, to reinterpret) the Putnam model so that it was suitable for modeling the real-time software development process.

## II. A Real-Time Embedded Computer Application

Fig. 2 shows the manpower curve for a project which delivered a trainer/simulator for a sonar and fire control system. The manpower included here is for the entire project from requirements specification through design, code, test, integration, and delivery. Delivery constituted a complete working system with a MTTF of more than seven days.

Fig. 2 is to be contrasted with Fig. 1. In Fig. 1, the manpower peak is at the first delivery, while in Fig. 2, the manpower peak occurs at the time of the completion of the software design.

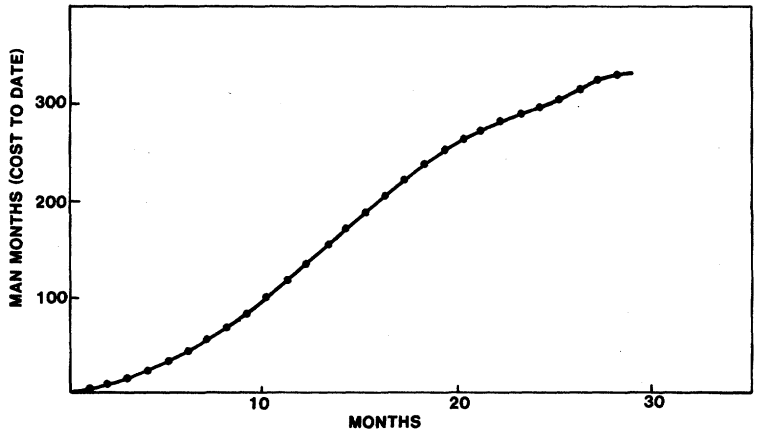



Fig. 3. Cumulative manpower data. This kind of plot is often preferred by accounting departments. It is smoother than Fig. 2, but less useful. It is much harder to predict the turnover and endpoints from this curve than it is to predict from Fig. 2.

The curve remains Rayleigh shaped, however. We therefore kept the mathematics of the Putnam model, and set about reinterpreting the meaning of the parameters.

### *A. Management Aspects of the Curve*

Just plotting the data is useful, providing an overall view of the project. There is, of course, noise in the data, but all significant fluctuations from smoothness reflect management problems or decisions. For example, in Fig. 2, point (1) is the first obvious irregularity. At this time the customer stretched funding by several months. The customer was thwarting the inclination of the project manager to build up the staff. Point (2) is interesting. At the peak of production one would not expect such dramatic variations in personnel. The fluctuations were actually due to a combination of snowstorms and the usual winter holidays.

At point (3) an interesting psychological trap develops—aggravated by typical accounting methods. At this stage of the project's development, there are 18–20 engineers producing code at a prodigious rate. Half the money is spent, and with all these people on board, the money will run out very soon. The combination of voluminous code and apparent running out of money led management to cut back the manpower. However, they did it too quickly.

The crisis is not even apparent when one plots the cumulative cost as in Fig. 3. The cumulative plot smooths out all of the interesting features. It is very difficult to predict anything from this type of plot. A more controlled decline in manpower would have avoided the subsequent mini-build-up which is shown in Fig. 2 at point (4). In this project, all manpower demands were filled on the basis of management requests, not Rayleigh curve estimation.

A major success of applying the Rayleigh curve technique comes as a result of identifying the curve peak. This can be done quite accurately. Knowing that the peak occurs after about 40 percent of the schedule gives the first good prediction of the project completion date. It is more difficult to project a completion date from the cumulative data of Fig. 3. The cumulative plot smooths out the changes toward the end of the project. Also, the point at which the cumulative curve flattens out is difficult to predict.

The area under the curve represents the total number of man-years expended. When multiplied by the average cost per man-year, the area under the curve represents the cost of the job. By manipulating the time of the peak and the size of the peak, different costs are obtained.

If there is scatter in the manpower data, the peak may be hard to identify. However, in that case, one can obtain a lower bound on costs by assuming the smallest peak at the earliest time, consistent with the data. This produces the minimum cost.

Cumulative plotting of manpower data is not so useful. Management can easily fool themselves into thinking that the curve will flatten at any time, whereas, the instantaneous plot of Fig. 2 quickly shows a more correct view. The plotting of instantaneous manpower versus time, instead of cumulative money spent, is probably the single most valuable tool of this entire methodology. It is simple and promotes straightforward communication.

### *B. Mathematical Model*

The Rayleigh distribution seems to fit a wide class of software projects, both small and large. The model also applies to business, real-time and embedded software. The Rayleigh curve for manpower has the form

$$m(t) = \frac{2Kt^2}{t_d^2} \exp\left[-\frac{t^2}{2t_d^2}\right]$$

where $m(t)$ is the manpower as a function of time $t$. $K$ is the total area under the curve and represents the total number of man-months for the project; $t_d$ is the time to the manpower peak. The logarithmic form is

$$\log(m/t^2) = -\frac{t^2}{2t_d^2} + \log(2Kt_d^{-2}).$$

Curves are hard to fit precisely—one prefers to fit to straight lines. This is done in Fig. 4 which plots $\log(m/t^2)$ against $t^2$

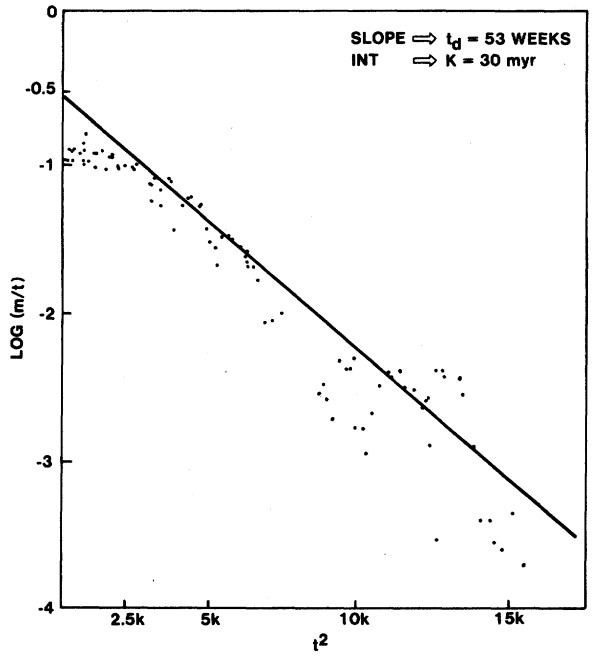



Fig. 4. This figure shows the data from Fig. 2 plotted in logarithmic coordinates. From the slope and intercept the values of $K$, the total manpower, and the time of the manpower peak are determined. The slope is consistent with a value of $t_d$ = 53 weeks and the intercept with a value of $K$ = 30 man-years. Both of these values are in agreement with the actual project values.

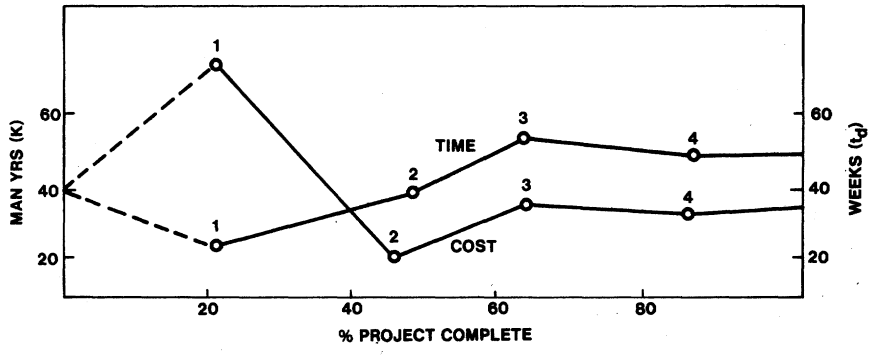



Fig. 5. Cost and schedule predictions. By using the data of Fig. 4 as it becomes available, one can make incremental predictions of the completion date and the final cost. This figure shows four predictions of both the cost and the schedule. Once the manpower has peaked, the predictions become very reliable and converge to the actual values.

for the data of Fig. 2. The slope and intercept predict the values $K$ = 30 man-years and $t_d$ = 53 weeks. The peak of Fig. 2 is consistent with a value of $t_d$ = 52 weeks.

The fit to the data is reasonably good, although the early data has considerable scatter. Also, fluctuations in this region are magnified by the choice of variables: the $(m/t^2)$ in the $y$ direction and the $t^2$ in the $x$ direction. Both tend to expand the plot for low $t$ values and compress it for high $t$ values.

The value of using the model in this way is in predicting project completion dates, while the project is in progress. The scatter in the early data makes predictions in the early phases somewhat erratic. Fig. 5 shows the predictions for completion date and total budget that one would have obtained at various stages of the project. Notice that once the manpower peak is identified, the projections converge to the actual values, and become quite reliable.

One of the benefits of using the Putnam method is clearly demonstrated here. It is very easy to pick out the manpower peak for a project and then deduce the completion date. In a cumulative plot, however, it is very difficult to determine the finishing date by projecting the flattening out of the manpower curve. In fact, without using mathematics, it is much easier to make projections from Fig. 2 than Fig. 3. This is intuitively slightly strange since Fig. 3 is smoother. Fig. 3 smoothes away the interesting effects.

One could try to determine the completion data by discovering where the Rayleigh curve falls to zero. However, it is much easier to project from the peak because of considerable noise

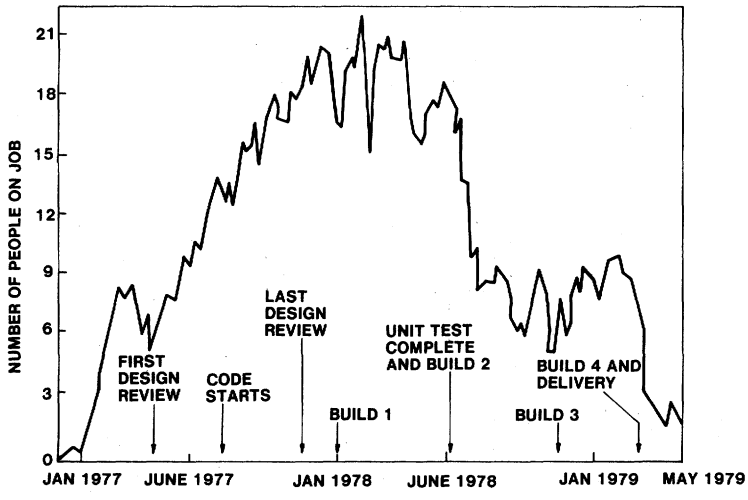



Fig. 6. The milestones for a typical software project are shown overlaid on the manpower curve. Knowing the place in time where these should occur, one can assess the correctness of the schedule.

in the tail due to follow-on contracts and maintenance. Knowing that the peak occurs after 40 percent of the project, the delivery date can be found quite accurately.

One caution about the logarithmic formulation should be mentioned. There is usually considerable scatter in early data, due to both systematic and random effects. The usual random effects such as staff availability, Thanksgiving, and sick leave are magnified because of the choice of variables, and the few people on the project. With a little management thought, the systematic effects can usually be discovered. Some examples of the kinds of things to look for are variations due to customer or management funding practices and inexact start dates.

### C. *Milestones–Avoiding the Pitfall*

A pitfall in using the Putnam model is that a manager may follow the curve and ignore the actual project status. There is a temptation to plan a project and staff it according to a Rayleigh curve. One can imagine a staff build-up, reaching the peak, and staffing down to zero only to find that the project is only 80 percent complete! The protection against this trap is to match milestones with the curve.

Fig. 6 shows the milestones for the project of Fig. 2. These are appropriate to a software project developed in the modern style. That is, top down design with reviews, structured code with walks through, unit testing, builds, and strict configuration management. Not all parts of the design were developed together. Critical modules and major components were designed first. This explains the milestones "first" and "last" design reviews. Coding was begun only after approval at a formal design review. However, some modules were in design while others were being coded. With 18-20 engineers and overlapping phases, strict configuration management and formal review procedures were essential.

Comparison of projected and actual milestones has to be done with intelligence. An example will illustrate the pitfalls. The completion of the program design is usually a major milestone. Unfortunately, there is practically no way to guarantee a complete, self-consistent design.

A poor, incomplete design specification published on-time to meet a milestone only pushes problems off to a later date. Design work will be performed during the coding phase. Unfortunately, when the size of the design task is underestimated, the code phase will also be underestimated. However, the appearance is that the coding phase milestones are slipping. The Rayleigh curve shows the magnification of the coding phase manpower. A design which is incomplete by 10 percent can cause the project to be late by 25 percent.

Hence, publication of a design specification just to meet a milestone may put off the recognition of trouble until coding milestones are not met. Instead of recognizing the problem early, when it can be easily attacked, it has been postponed until it can do considerably more damage. It is far better to recognize that the design is incomplete and either reschedule or adjust the project. At design time, it is still early enough to present management with options. The Rayleigh curve can be used to estimate the growth of the manpower requirements and project the budget problems. At design time, there is time to affect the implementation given the type of information available from the model.

## III. A Second Case Study

Projects experiencing difficulties are seldom discussed in the literature. Authors prefer to talk about their successes. However, there are lessons to be learned from problem projects. In fact, such painful lessons may be of considerable value to the community.

In this section, it is shown how the model can be used as a prediction tool. Fig. 7 shows the manpower data collected for another project. The data are plotted up to the point where some concern was being expressed that the project was not going as well as expected. The smooth curve is a Rayleigh curve deduced from the data. It was not used by management as a staffing curve. Staffing was done through the usual combination of availability and demand.

The actual staff build-up was reasonable. However, at the point at which the data stop, the project manager was still

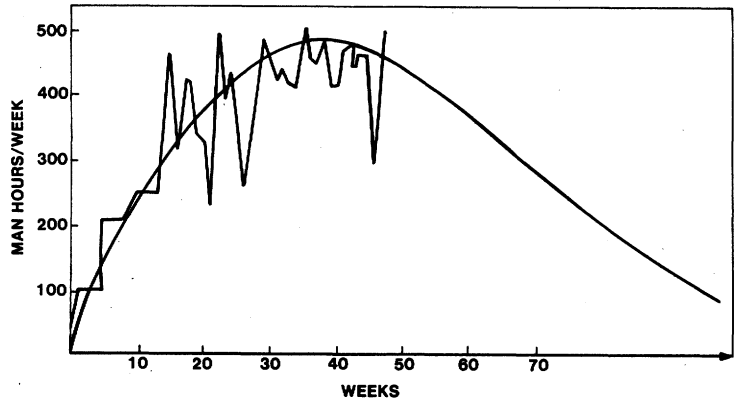



Fig. 7. Manpower data for a second real-time software project. The data are shown up to the point where concern was being expressed that the project was not on schedule. Predictions of final cost and schedule from this data correctly showed the slippage and increase in the delivered amount of code.

asking for staff. The first indications of trouble were on the horizon,[3] and an analysis of the project status was conducted.

The first step in the analysis was to obtain the manpower data. After examining the data, it was not clear if the manpower had peaked or not. The following approach was taken. We assumed that the manpower peak was at the time when the data stopped. If the peak were assumed to be any later, the resulting project cost would be higher. Also, by examining the milestones completed, we saw that the project had completed its design reviews but had not generated a build. Therefore, everything suggested that the project should have been around the manpower peak. The minimum cost to complete was found by assuming the manpower data had peaked at the latest value. The cost, schedule, and estimated source line count were projected.

The job was originally estimated to be a two-year task with approximately 10K source lines. The most optimistic predictions of the Putnam model showed a 6-9 month slip and a total of 20-25K lines. No one had checked the source line count since the original bid. (One group leader was found to be rapidly approaching 10K lines.) These "theoretical" projections were ignored until a few months before the originally scheduled delivery, when the actual status could not be denied. At this point, a reorganization occurred and a new cost to complete was generated.

The eventual costs and completion dates were consistent with the predictions made using the Putnam model. The predictions were quite accurate enough to show the difference between the actual status and the hoped-for status.

When it is determined that a software project is going to be late, the temptation is to put extra people on the job to finish it quicker. Using the Putnam model, one can interpret that situation as follows: putting extra people on the job makes the peak higher and the job will therefore cost more. This suggests that the size of the job was probably underestimated, and that the actual job will follow a Rayleigh distribution with a higher peak. The job will not finish sooner, but later. Thus we see more evidence for Brooke's Law—adding people to a late job makes it later. The expectation, of course, is that adding people will shorten the time to completion. This is only true if the predicted and actual status are the same and the people are added early enough to change to a new Rayleigh profile. The usual situation is that the job is underestimated, making the peak later and higher.

Current software technology efforts are directed towards improving software design tools. It is recognized that the technical completeness and consistency of the eventual product depends critically on having a good design. The Putnam model provides a management tool analogous to design tools.

Management leverage, like technical leverage, is greatest early on. The Putnam model, if used during the design phase, can provide management benefits analogous to those found by applying design tools. One should expect of management exactly what one expects of a design, i.e., self-consistency and completeness, only applied to milestones, staffing, and costs.

There are several lessons to be learned from this difficult experience. The first is that Rayleigh curve estimating can help a manager assess the progress of a project. However, the application of the model should be backed up by data and experience on the project. The model can be used to obtain clues about the status of the job. Management insight must then be used to track down and resolve the problems and inconsistencies.

In the above case, the source line count was so high that it triggered an investigation. The trend there gave credence to the schedule and cost slippages. Other project data and general detective work showed the real status of the job.

Fig. 7 contains information on the responsiveness of management. The data for this project suggest that the spending was carefully adjusted so as to be "on target." The under and overshoots represent management adjusting to pressures. The characteristic time scale for these adjustments is 4-6 weeks. This time scale is not a characteristic of the project, it probably reflects monthly accounting reviews and the adjustment of the personnel to the dollars scheduled in the reviews. The cost

[3] At this point, staff grumbling was getting out of hand. There is a gray area between "this milestone is difficult" and "this milestone is impossible." Management needs concrete evidence to recognize the transition from the first to the second.

was being adjusted independently of the technical progress of the project.

## IV. Model Extensions

We have examined the cost and schedule predictions associated with the manpower curve. We now show how the model is extended to include other parameters useful to a manager.

Putnam defines a quantity called the "difficulty" as follows:

$$\text{difficulty } D = K/t_d^2 \tag{2}$$

where $K$ is again the total manpower (cost) and $t_d$ is the time at which the manpower peaks. As an example, consider the project shown in Fig. 2. There, $K$ = 30 man-years and $t_d$ = one year. The difficulty of that project is therefore 30. This value for the "difficulty" turns out to be quite representative of medium size real-time software projects. We now define a "productivity" function $P$ as follows:

$$P = \frac{\text{total number of source lines}}{\text{total number of man-months}}. \tag{3}$$

This productivity function is an average value over the whole project. Using data for a considerable number of projects,[4] Putnam deduced the following realtionship between the productivity $P$ and the difficulty $D$:

$$P = \text{constant } D^{-2/3}. \tag{4}$$

After substituting for the difficuly function, one obtains an expression for the source lines, as follows:

$$S = c_t K^{1/3} t_d^{4/3} \tag{5}$$

where $c_t$ is a constant. This formula allows one to estimate source lines from the manpower and schedule. Or, in the more usual case, given a schedule and some estimate for the source line count, one can estimate the cost.

The constant $c_t$ is called the "technology constant," because it seems to have increased as software projects have moved from an assembly language, batch oriented approach to high-level languages and on-line systems.

For the data in Fig. 2, the following values can be calculated:

| | |
|---|---|
| difficulty = 30 | a typical value for a new real-time system |
| technology constant = 8000 | a typical value for a project with design reviews, on-line development, and structured code. |

The treatment here is different from the original Putnam model. We assumed that the above equations apply over the entire life cycle, whereas Putnam applies them to the detail design and code subcycle only. Furthermore, Putnam *assumes* that the detail design and code subcycle peaks at a value of $t_d^{1/2}/3$. There is no rationale for this choice, although it is mathematically convenient. We avoid this whole discussion by applying (2) to (5) to the whole life cycle.

[4]See, for example, "Software data collection and analysis at RADC," Rome Air Development Center, Rome, NY, 1978.

## V. Management Aspects of the Difficulty and Technology Constants

The claim has been made that the constant $c_t$ is quantized as

$$c_t = 6000, 8000, 10000.$$

This assumption seems unrealistic. However, a range of parameters can be associated with project types. We have gathered data that provides us with a range of difficulty values. As part of a proposal, a manpower estimate is calculated. One can usually characterize the difficulty from experience and historical data. This leads to a range of values for the development time. Because the difficulty is a steep function of time, this analysis is extremely valuable. A risk analysis is performed by varying the project parameters, and shows for a given project how the risks are distributed among cost, schedule, or code estimates.

Whereas we find the difficulty function is predictable, the technology constant is more variable and not so reliable. However, the best method of use for these relations is via tradeoffs. Often, one or another of the parameters is inconsistent. One then has the task of adjusting the others until a consensus is reached. The value of this process is not just in estimating values for the project. By varying parameters, one obtains an increased understanding and knowledge about the job at hand.

The process of adjusting and playing with the parameters usually highlights the risk areas. Very often one hears a statement such as "that instruction count looks low." Using the above techniques, one can begin to quantify such statements and decide whether one's intuition is correctly based.

## VI. Future Directions

A major uncertainty associated with Rayleigh curve estimation is that the formula is empirical—it has no physical or statistical background. A formal derivation of the Rayleigh curve with a prediction for the values of the parameters would lead to a better understanding of the model.

There are several aspects of software methodology that one would like to see addressed in the Putnam model. For example, what is the effect on manpower curves of the present trend toward more requirements and design analysis? Will the curve remain Rayleigh shaped? If it does, software could cost less.

It might be suggested that the Rayleigh curve technique was formulated using data from the 1970's, before top-down design and structured coding were commonplace. Will the technique continue to be applicable as methods evolve? The answer to this question seems to be "yes" since the project of Fig. 1 (an excellent fit to a Rayleigh curve) was completed with such practices as multiple review cycles, structured code, walks through, and module development folders. It appears, therefore, that Rayleigh modeling is an appropriate tool to apply to modern software development.

## VII. Conclusions

The Putnam model provides an effective management tool. We have applied the model slightly differently from the standard approach, accounting for the different staffing profile that we apply to real-time software development.

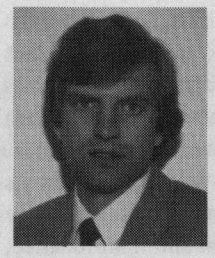
One of the valuable aspects of the Rayleigh curve is the de-

termination of the manpower peak. The ratio of manpower peak to completion date is quite precise and is the best method of determining the delivery data. Cumulative manpower plots are particularly unsuited for any type of predictions, changing most slowly at the time when one desires the most information. We have shown how valuable management information can be obtained by laying out the data in a systematic manner.

The manpower data by itself should not be used to drive the project. The data are used for clues that point to the real status and problems. The key to success in this aspect of modeling is the establishment of reliable and identifiable milestones and an understanding of their relationship to the overall project.

### Acknowledgment

W. J. Wesner of Raytheon managed the project described in Fig. 1. His contributions to the analysis of the data are gratefully acknowledged. The author would also like to thank the referee for constructive suggestions.



**Roger D. H. Warburton** was born in Cardiff, Wales. He received a degree in physics from Sussex University, Sussex, England, and the Ph.D. degree in physics from the University of Pennsylvania, Philadelphia, where he studied under a Thouron Scholarship.

After working at Raytheon, Portsmouth, RI, for several years, he became Manager of Software Technology, specializing in software cost estimation, CAD tools for signal processing, and factory test languages. He is currently Manager of Operations for JAYCOR, Middletown, RI.

# Analyzing Software Safety

NANCY G. LEVESON AND PETER R. HARVEY

*Abstract*—**With the increased use of software controls in critical real-time applications, a new dimension has been introduced into software reliability—the "cost" of errors. The problems of safety have become critical as these applications have increasingly included areas where the consequences of failure are serious and may involve grave dangers to human life and property. This paper defines software safety and describes a technique called software fault tree analysis which can be used to analyze a design as to its safety. The technique has been applied to a program which controls the flight and telemetry for a University of California spacecraft. A critical failure scenario was detected by the technique which had not been revealed during substantial testing of the program. Parts of this analysis are presented as an example of the use of the technique and the results are discussed.**

*Index Terms*—**Fail-safe software, fault tree, real-time software, safety verification, software reliability, software safety, software validation, system safety.**

Manuscript received July 30, 1982; revised January 20, 1983. This work was supported in part by Contract 7-656146-T-DS with Hughes Aircraft Company and by a joint grant of the University of California MICRO Project and Hughes Aircraft Company. The project is receiving additional support from the System Development Corporation.

The authors are with the Department of Information and Computer Science, University of California, Irvine, CA 92717.

## Introduction

IN RECENT YEARS, advances in computer technology have gone hand in hand with the introduction of computer usage in new application areas. The problems of safety have become critical as these applications have increasingly included areas where the consequences of failure are serious and may involve grave dangers to human life and property. Computers currently control reactions in nuclear power plants, track airplane positions in air traffic control systems, monitor patients in intensive care units of hospitals, deal with the complexities of space flight in aerospace programs, and control military and defense systems.

There is growing concern about errors in these systems. Much of the focus in software research has been on techniques to eliminate errors prior to the operational use of the software system [4]. For the most part, software errors have been regarded as a temporary problem which will disappear as soon as adequate methodologies for program development and validation can be devised. But progress in developing these methodologies has been slow, and error-free software may not be a

0098-5589/83/0900-0569$01.00 