COMPUTER SOFTWARE SCHEDULE ESTIMATION: AN APPRAISAL

Daniel V. Ferens
Air Force Institute of Technology (AFIT/LSY)
Wright-Patterson Air Force Base, OH

- ABSTRACT

This paper discusses the area of software schedule estimation; including the results of recent research conducted at Air Force Institute of Technology to assess which factors influence software schedules, and to determine which commercially-available software cost models contain algorithms which are most suitable for software schedule estimation. The suitability of scheduling methodologies such as the Program Evaluation and Review Technique for software scheduling will also be discussed. Finally, an assessment of the current status of software schedule estimation will be presented, along with recommendations for further work in this area.

- INTRODUCTION

Schedule delays can be very costly to software development projects. For systems such as defense weapon systems, software schedule delays can be especially harmful since the entire system schedule can be adversely affected. However, these delays seem to have been the norm for software development efforts in the past. Perhaps a principal cause of this condition is that software project managers have been unable to, or have not known how to estimate software schedules during the early phases of a program.

Fortunately, there are several tools and techniques that are available for estimating software schedules. Most popular parametric cost models contain scheduling algorithms; and traditional scheduling techniques such as Gantt charts and the program evaluation and review technique (PERT) have been used for software scheduling. However, all of the available techniques do have significant limitations which a manager must be cognizant.

A sampling of available models and techniques will now be discussed, including an assessment of the capabilities and limitations of these techniques. The current techniques have been divided into two categories for clarity: cost model scheduling algorithms and traditional scheduling techniques. Future directions in software schedule estimation will also be discussed.

- COST MODEL SCHEDULING ALGORITHMS

Most detailed software parametric cost models contain scheduling algorithms. Algorithms for six of the better-known software cost models will now be discussed. For all of the software cost models except the Constructive Cost Model (COCOMO), portions of the scheduling algorithms may be proprietary; therefore, not all information is available or can not be presented. The six models are discussed in approximate chronological order of development.

PRICE-S: This parametric cost model was probably the first large-scale parametric cost model to be marketed, and is still very popular today. The internal equations of the PRICE-S model have not been available in the past; however, Dr. Robert Park of PRICE Systems, the marketing agency for PRICE-S, recently published a paper [1] describing many of the model's equations in detail, including selected scheduling equations. According to Dr. Park, PRICE-S estimates schedule based primarily on the model inputs of size, application, complexity, and component reuse. Other inputs are used to make adjustments to the schedule as needed. PRICE-S uses Beta curves for three stages of software development; design, code, and test; to estimate total development months and to define personnel profiles over the development cycle. The PRICE-S model also outputs expected times for traditional software development milestones, such as software specification review (SSR), preliminary design review (PDR), critical design review (CDR), and functional configuration audit (FCA) which usually occurs at the end of development. The
model computes an "ideal" schedule based on optimal use of resources, and will, at user request, compare the model's predicted schedule with the user's schedule and assess penalties due to compression or extension compared with the schedule computed by the PRICE-S model.

SLIM: The Software Life Cycle Model (SLIM) was developed by Quantitative Software Management and is based on the work of Larry Putnam [2]. The primary estimating algorithm used in SLIM is the Rayleigh-Norden curve, illustrated in Figure 1, which relates development time to effort, program size, and a technology factor; all of which are inputs to SLIM. Other SLIM inputs which affect schedule include software type, language, and use of modern development practices. The primary schedule output for SLIM is a "minimum time" solution which illustrates the least time in which the software program can be developed. The user can then specify a longer schedule and assess the cost or effort savings associated with schedule expansion. (Apparently, SLIM assumes that the schedule can be extended indefinitely with a corresponding decrease in cost or effort required.) SLIM can also provide manloading profiles, times for selected milestones, and a risk analysis of schedule completion.

COCOMO: This model was developed by Dr. Barry Boehm of TRW Corporation, and is published in his popular book, Software Engineering Economics [3]. Figure 2 shows the effort and schedule equations for the Basic COCOMO, as described in Dr. Boehm's book; where "KDSI" is thousands of delivered source instructions, "MM" is man-months of effort, and "TDEV" is development time in months. The primary input to COCOMO is size in KDSI. However, in the more advanced intermediate and detailed versions of COCOMO, other factors are used to compute manmonths and, consequently, affect the schedule. Some of these inputs include personnel qualifications, reliability requirements, use of modern design practices and tools, and complexity. COCOMO uses a modified version of the Rayleigh-Norden curve in computing software schedule. The primary schedule output is development time; however, the model can also provide manloading profiles based on the modified Rayleigh-Norden curve, and times for selected milestones such as SSR and PDR.

SYSTEM-3: This model was developed by Computer Economics, Inc., and is based on the work of Dr. Randall Jensen [4]. Like SLIM, the SYSTEM-3 model uses the Rayleigh-Norden curve to estimate software schedule, and outputs a "minimum time" solution. However, unlike SLIM, this model also outputs a "minimum effort" schedule which is usually longer than the minimum time schedule, but does not extend indefinitely. Like PRICE-S, SYSTEM-3 assumes that, beyond the minimum effort time, resources are being used inefficiently and, consequently, more effort will be required as the schedule is expanded. The primary model inputs which affect schedule include size, development capability factors, complexity, and environmental parameters. The model's schedule outputs include; in addition to minimum and optimal development times; the times of milestones such as PDR and CDR, manloading profiles, and schedule risk analyses.

SPQR/20: The Software Productivity, Quality, and Reliability model (SPQR/20) was developed by Software Productivity Research and is based on the work of Capers Jones [5]. This model is unique in that it does not require size as an input if program function points are known. It also uses about twenty other inputs including project type and class, environmental factors, complexity, and code descriptors such as language used. SPQR/20 outputs development schedule in months, and provides times for milestones such as SSR and CDR.

SOFTCOST-R: This model was developed
by Reifer Consultants, Inc.; and is based on the work of Dr. Robert Tausworthe at the Jet Propulsion Laboratory [6]. Inputs to SOFTCOST-R which affect scheduling include program size and factors in the categories of management, staffing, complexity, and environment. Although the scheduling algorithms are not explained in depth in the SOFTCOST-R manual [6], the model apparently uses some variant of the Rayleigh-Norden curve. The model outputs include a development schedule in months, manloading profiles, and resource allocation summaries. A notable scheduling feature of SOFTCOST-R is that it provides information for constructing Gantt and PERT charts, which will now be described in more detail.

TRADITIONAL SCHEDULING TECHNIQUES

The scheduling techniques shown in Figure 3; Gantt charts, milestone charts, and PERT diagrams; have been in use for many years for a wide variety of scheduling applications. According to Rosenau and Lewin [7] and other authors, these techniques have been tried for software schedule estimation with varying results. A brief description of each technique is now presented.

Gantt Charts: These charts, also called bar charts, have been in use since World War I. Gantt charts are used to compare estimated and actual times for project activities, as shown by the clear and shaded activity areas in Figure 3a. At a given time, the percentage completed for an activity is assessed and compared with the projected percentage completed for that activity, and an estimate of completion time is plotted. The primary advantage of Gantt charts is that they are easy to construct and understand. However, they do not illustrate the interrelationships or interdependencies among project activities which may affect the completion times of individual activities. Furthermore, percentage completions of activities are often misleading or difficult to assess.

Milestone Charts: These charts, illustrated in Figure 3b, denote key events and expected times between events. As discussed previously, several cost models can output information which can be used to construct milestone charts for single software efforts. Like Gantt charts, milestone charts are easy to understand. They also provide high level project visibility since they emphasize significant events or completed products during a project. However, like Gantt charts, they do not illustrate the interrelationships and interdependencies among project activities.

PERT Charts: PERT charts became very popular during the 1950s as a planning tool for research and development endeavors. Figure 3c provides an example of a PERT diagram, where circles illustrate events (milestones), arrows between circles illustrate activities, and numbers along arrows represent expected times to complete activities. PERT diagrams do show the interrelationships and interdependencies among activities and events; thus overcoming a major weakness of Gantt and milestone charts. Also, according to Kerzner [8], PERT diagrams can sometimes be constructed directly from a combination of related Gantt and milestone charts, incorporating the advantages of both of those techniques. Although PERT has received acclaim for its usefulness as a project management tool, it is not without limitations. As illustrated by Alford [9], PERT diagrams are often difficult to construct. Also, an obsession with the PERT process itself can sometimes obscure its role as a tool for project management planning and decision making.

AN ASSESSMENT OF SCHEDULING TECHNIQUES

Unfortunately, as of this writing, few comprehensive studies have been accom-
plished to determine the expected validity or accuracy of software scheduling techniques, especially for the cost model scheduling algorithms. Therefore, accuracy claims made by model developers are difficult to substantiate. For example, Dr. Barry Boehm claims that the COCOMO scheduling equations shown in Figure 2 are accurate to "within 20%, 56% of the time" [3]. However, a study performed by Blalock [10] showed that COCOMO was the least accurate of five models studied; it had a greater than 50% error in estimating schedule. The other four models; PRICE-S, SYSTEM-3, SPQR/20, and SOFTCOST-R, were accurate to within 20% of actual schedule. In defense of Dr. Boehm, Blalock’s study contained a very limited data base, and no general conclusions could be made regarding the models studied. Blalock’s research would have to be expanded to a larger database and, perhaps, to a wider variety of programs before any conclusions could be made regarding the models studied.

Regarding traditional estimating techniques, all three appear to be of limited use for software schedule estimation. Gantt charts are very limited because percentage completion is difficult to measure for software and, as Rosenau and Lewin [7] assert, they seem to be more useful in tracking what has already occurred than in planning and estimating. Milestone charts are produced by most cost models, but the times have not been validated. PERT may show more promise than the other techniques, but it also has its weaknesses. Metzger [11] states that PERT has had only spotty success in programming projects, and often requires an enormous amount of effort such that the diagrams sometimes cover entire walls. A limitation common to all traditional scheduling techniques is that it may be difficult to estimate activity times early in a program when the required detailed planning information is frequently unavailable.

CONCLUSIONS

A software manager does have a number of models and techniques available for software schedule estimation. However, all of the currently available models and techniques have several limitations. Furthermore, very few studies have been done to establish the validity, accuracy, or even the usefulness of these models and techniques. A software manager, therefore, must use any scheduling model or technique very carefully.

It is anticipated that software schedule estimation will become even more challenging in the future. New software development techniques such as rapid prototyping are becoming more prevalent; however, most existing cost models assume the traditional "waterfall" approach to software development. To be useful for new development approaches, the models will probably require significant modifications. Fortunately one model, SOFTCOST-R, allows the user to choose among three development techniques in schedule estimation: traditional (waterfall) development, incremental development (prototyping), and object oriented development frequently used with the Ada language [6]. Also, COCOMO and SOFTCOST-R have recently introduced versions for the Ada language, which may facilitate schedule estimation for Ada programs. However, the effectiveness of these new models is still uncertain.

Overall, it appears that much more research is needed in software schedule estimation, especially in validating existing models and techniques. As such, the current status of software schedule estimation may be summarized in the title of a song by the Carpenters in 1970: "We've Only Just Begun."

REFERENCES


