Analyzing Medium-scale Software Development

Victor R. Basili and Marvin V. Zelkowitz

Department of Computer Science University of Maryland
College Park, Maryland 20742

ABSTRACT

The collection and analysis of data from programming projects is necessary for the appropriate evaluation of software engineering methodologies. Towards this end, the Software Engineering Laboratory was organized between the University of Maryland and NASA Goddard Space Flight Center. This paper describes the structure of the Laboratory and provides some data on project evaluation from some of the early projects that have been monitored. The analysis relates to resource forecasting using a model of the project life cycle based upon the Rayleigh equation and to error rates applying ideas developed by Belady and Lehman.

GOALS OF LABORATORY

A great deal of time and money has been and will continue to be spent in developing software. Much effort has gone into the generation of various software development methodologies that are meant to improve both the process and the product [Myers, Baker, Wolverton]. Unfortunately, it has not always been clear what the underlying principles involved in the software development process are and what effect the methodologies have; it is not always clear what constitutes a better product. Thus progress in finding techniques that produce better, cheaper software depends on developing new deeper understandings of good software and the software development process. At the same time we must continue to produce software.

In order to investigate these issues, the Software Engineering Laboratory was established,in August, 1976, at NASA Goddard Space Flight Center in cooperation with the University of Maryland to promote such understandings [Basili & Zelkowitz]. The goals of the Laboratory are to analyze the software development process and the software produced in order to understand the development process, the software product, the effects of various "improvements" on the process and to develop quantitative measures that correlate well with intuitive notions of good software.

The goals of the Laboratory can be broken down into three major tasks:

1. Provide a reporting mechanism for monitoring current project progress. This goal is to provide management with up-to-date data on current project development. Better reporting procedures can pinpoint problems as they develop and help eliminate their spread and growth.

2. Collect data at as fine a level as possible that can be used to determine how the software is being developed, extend results that have been reported in the literature about very large software developments and their characteristics to medium sized projects (5 to 10 man-years), help discover what parameters can be validly isolated, expose the parameters that appear to be causing trouble, and discover appropriate milestones and techniques that show success under certain conditions.

3. By comparing data collected from several NASA projects, compare the effects of various technologies and other parameters upon system development and performance.

LABORATORY OPERATION

Projects for the Systems Development Section at NASA typically are produced by an outside contractor under supervision by NASA employees. Most products are in the 5

This research was sponsored in part by grant NSG-5123 from NASA Goddard Space Flight Center, Greenbelt, Maryland to the University of Maryland.

to 10 man-year range in size, and are generally large batch programs for an IBM 360 system. The programs are almost always written in FORTRAN.

To evaluate programming methodologies, a mechanism was established to collect data on each such project. The initial goal was to collect as much relevant data as possible with as little impact on the projects and software development practices as possible. It is believed that although there has been some impact and interference, it has been minimal. As we gain knowledge as to what data to collect, we hope to shorten the manual input from the project personnel, and to automate some of the tasks.

Similar to other reporting projects of this type, the principal data gathering mechanism is a set of seven reporting forms that are filled out by project personnel at various times in the development life cycle of a project [Walston & Felix]. Some of these are filled out only once or twice, while others are filled out regularly. The seven forms that are currently in use include:

1. General Project Summary. This form is filled out or updated at each project milestone and defines the scope of the problem, how much has been completed, estimates for the remainder of the project, and what techniques are being used. It is a top level structure of the overall organization and is filled out by the project manager.

2. Component Summary. This form is filled out during the design phase and describes the structure of each component (e. g. subroutine, COMMON block, etc.)

3. Programmer Analyst Survey. This form is filled out once by each programmer in order to provide a general background of project personnel.

4. Resource Summary. This form is filled out weekly by the project manager and gives manpower and other resources charged to the project during the week.

5. Component Status Report. This is the major accounting form that lists, for each programmer, what activities were performed on each component for the week. This is the basic form that lists what happened and when.

6. Computer Program Run Analysis. This form contains an entry each time the computer is used. It beiefly describes what the computer is used for (e. g. compile, test, etc.) and what happened (e. g. error messages).

7. Change Report Form. This form is completed for each change made to the system. The reason for and a description of the change are given. If the change is made to correct an error, the method of detection, effects on other parts of the system, time to correct and type of error are noted on the form.

The data that is collected is entered into the INGRES PDP 11 data base system [Held]. This process is somewhat tedious due to the care needed to insure data validity. Almost all of the errors not detected by hand checking of the coded input is detected by the input program.

All projects that are currently being monitored can be broken down into three broad classifications:

1. The screening experiments are the projects that simply have the requirement to submit reporting forms. They provide a base line from which further comparisons can be made, and upon which the monitoring methodology can be tested.

2. The semi-controlled experiments are a set of relatively similar large scale developments. While they are different projects, they are sufficiently similar in size and scope so that comparisons can be made across these projects. In this case, specific techniques are sometimes required to be used in order to measure their effectiveness. These projects are the standard spacecraft software developed by the Systems Development Section at NASA.

3. The controlled experiments are a set of projects that are developed using different methodologies. These developments are the most closely monitored and controlled of the three classifications so that the effects of methodology upon these projects can more easily be measured than in the semi-controlled experiments.

For each project, a set of factors that effect software development are extracted by the forms. Some of the factors that are of interest include:

1. People factors (size and expertise of development team, team organization)

2. Problem factors (type of problem to solve, magnitude of problem, format of specifications, constraints placed upon solution)

3. Process factors (specification, design and programming languages, techniques such as code reading, walkthroughs, top down design and structured programming)

4. Product factors (reliability, size of system, efficiency, structure of control)

5. Resource factors (target and development computer system, development time, budget)

6. Tools (Libraries, compilers, testing tools, maintenance tools)

Some of these factors can be controlled while others are inflexible. Such items as development computer system, budget, format of input specifications and type of problem to solve are mostly fixed and change very slowly year by year. On the other hand, factors like structured programming, design techniques and team organization are much more under the control of the laboratory and can be varied across different projects.

For each semi-controlled or controlled project, a set of these factors is predetermined. For example, a project may use a librarian, code reading, walkthroughs, a PDL and structured programming. The other factors that affect development will become apparent through the information obtained on the general project summary. In order to enforce these methodologies on project personnel, a training period, consisting from a two hour lecture on filling out forms up to a week's classroom training, is being utilized. Every effort is being made to use methodologies that are compatible with a project manager's basic beliefs so that no friction develops between what the manager wants to do and what he must do.

Much of the early effort in the Laboratory was expended in the organization of the operation and generation of data collection and validation procedures and forms. We have reached a point where sufficient data has been obtained to permit us to evaluate our operational procedures and to analyze data with respect to goals one and two in the introduction. In the following two sections, early evaluation of the collected data is presented. The major emphasis in these first evaluations is on reporting progress and reliability of the developing system.

PROGRESS FORECASTING

One important aspect of project control is the accurate prediction of future costs and schedules. A model of project progress has been developed and with it estimates on project costs can be predicted.

The Rayleigh curve has been found to closely resemble the life cycle costs on large scale software projects [Norden, Putnam]. At present, we are assuming that this is true for medium scale projects as well, and are developing reporting procedures based upon this function. As data becomes available, we will be better able to test the underlying hypothesis and refine it further.

The Rayleigh curve yielding current resource expenditures (y) at time (t) is given by the equation:

$$y = 2 K a t \exp(-a t^2)$$

where the constant K is the total estimated project cost, and the constant a is equal to 1/(Td**2) where Td is the time when development expenditures reach a maximum. In our environment K and a are measures of hours of effort, and t is given in weeks.

Estimates on Initial Data

For each project in the NASA environment, the requirements phase yields estimates of the total resources and development time needed for completion. This data is obtained by the Laboratory via the General Project Summary form. From this data, a Rayleigh curve for this project can be computed.

From the General Project Summary, the following three parameters are relevant to this analysis:

1) Ka, total estimated resources needed to complete the project through acceptance testing (in hours).

2) Yd, the maximum resources needed per week to complete the project (in hours).

3) Ta, the number of weeks until acceptance testing.

Since the Rayleigh curve has only two parameters (K and a), the above system is over specified and one of the above

variables can be determined from the other two. Since NASA budgets are generally fixed a year in advance, there is usually little that can be done with total resources available (K). Also, since the contractor assigns a fixed number of individuals to work on the project, the maximum resources Yd (at least for several months) is also relatively fixed. Therefore, the completion date (Ta) will vary depending upon K and Yd.

As stated above, Ka is the total estimated resources needed to develop and test the system through the acceptance testing stage. By analyzing previous NASA projects, this figure Ka is about 88% of total expenditures K. The remaining 12% goes towards last minute changes. The seemingly low figure of only 12% to cover everything other than design, coding, and testing can be explained by the following two facts local to our NASA environment:

1) the initial requirements and specifications phases are handled by different groups from the development section, and thus this data does not appear, and

2) shortly after acceptance testing, a third group undertakes the maintenance operation, and so the full maintenance costs also are not included in the estimates.

For this reason it should be clear that we have no actual data to match the Rayleigh curve in the early stage (requirements) and late stage (maintenance). However, the major central portion of the curve should be a reliable estimate of the development costs, and it is here that we hope to prove consistency between the data collected on these medium scale projects and the large scale projects in the literature. Besides, on the large scale projects, the Rayleigh curve also acts as an accurate predictor of the design, coding, and testing stages both combined and individually [Putnam]. (In the future we expect to obtain some data on the long term maintenance phase. A Maintenance Reporting Form has been developed, and the maintenance section has agreed to fill out this form and report back the data. Due to the lifetimes of these spacecraft related software systems, the data will not be available for about another year.)

Thus given the estimate of project costs Ka in hours, the total resources needed is given by:

$$Ka = .88\ K$$

or

$$K = Ka/.88$$

The raw data for personnel resource estimates are not directly usable in our analyses since they include individuals of varying functions and salaries and therefore varying costs. The following normalization algorithm has been applied to the resource data in computing Ka: Each programmer hour is given a weight of 1, an hour of management time costs 1.5 while a support hour (secretary, typing, librarian, etc.) costs .5. This is a reasonable approximation to the true costs at NASA.

Then given constant a, the date of acceptance testing Ta can be computed as follows. The integral form of the Rayleigh curve is given by:

$$E = K\ (1 - \exp(-a\ t^2))$$

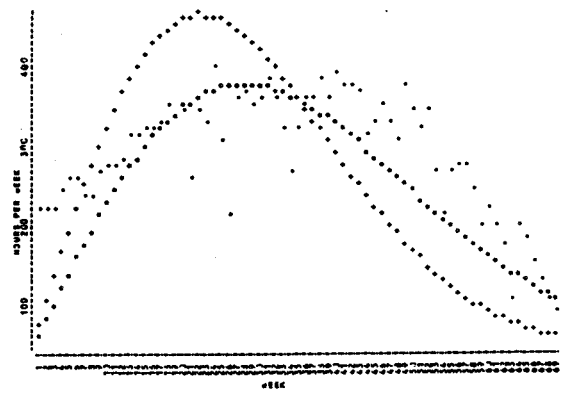
where E is the total expenditures until time t. From the previous discussion, we know that at acceptance testing, E is .88K. Therefore,

$$.88K = K\ (1 - \exp(-at^2))$$

Solving for t yields:

$$t = \mathrm{sqrt}(\ -\ln(.12)/a\ )$$

Putnam [Putnam2] states that for development efforts only, acceptance testing (Ta) is related to the time of peak effort (Tp) by the relation:

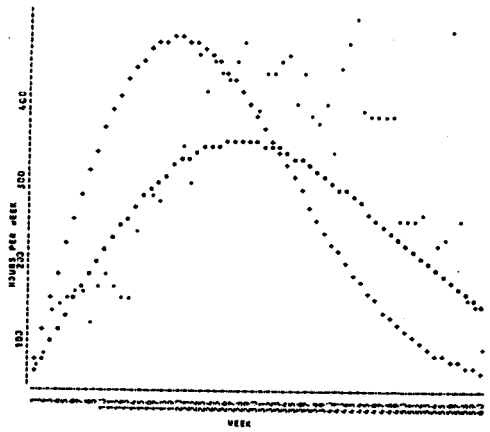


* - Estimating curve with Yd (maximum resources) fixed

\+ - Estimating curve with Ta (Completion date) fixed

. - Actual data

Figure 1. Project A - Estimated resource expenditures curve



* - Estimating curve with Yd (maximum resources) fixed

\+ - Estimating curve with Ta (completion date) fixe

. - Actual data

Figure 2. Project B - Estimated resource expenditures curve

$$Tp = \frac{Ta}{sqrt(6)}$$

or

$$Ta = Tp * sqrt(6)$$

From our own smaller projects, we found that this gives answers consistently higher by about 8 to 10 weeks, therefore we are using our own .88K rule to determine acceptance testing. Why our projects do not agree with the empirical evidence of large scale projects in this area is now under study.

Taking the given value of K, two different Rayleigh curve estimates were plotted for each of two different projects (referred to as projects A and B) by adjusting the constant a. For one estimating curve it was assumed that the estimate for maximum resources per week Yd was accurate and that the acceptance testing date Ta could vary, while in the other case the assumed acceptance testing date Ta was fixed and the constant a could be adjusted to determine maximum weekly expenditures Yd needed to meet the target date. These plots for the two different projects are shown as figures 1 and 2.

The curve limiting maximum weekly expenditures might be considered the more valuable of the two since it more closely approximates project development during the early stages of the project. In both projects A and B, the maximum resource estimate Yd was predicted to be insufficient for completing acceptance testing by the initially estimated completion date Ta. In project A the Rayleigh curve prediction for acceptance testing was 58 weeks instead of the proposed 46 weeks. The actual date was 62 weeks - yielding only a 7% error (Figure 3). The prediction for project B showed similar results.

	PROJECT A	PROJECT B
A INITIAL ESTIMATES FROM GENERAL PROJECT SUMMARY		
Ka; Resources needed (hours)	14,213	12,997
Ta; Time to completion (weeks)	46	41
Yd; Maximum resources/week (hrs)	350	320
B COMPLETION ESTIMATES USING RAYLEIGH CURVE		
K; Resources needed (hours)	16,151	14,770
Estimated Yd with Ta fixed (hrs)	440	456
Estimated Ta with Yd fixed (hrs)	58	58
C ACTUAL PROJECT DATA		
K; Resources needed (hrs)	17,742	16,543
Yd; Maximum resources (hrs)	371	462
Ta; Completion time (weeks)	62	54
Ta; estimated using actual values of K and Yd (weeks)	60	43

Figure 3. Estimating Ta and Yd from General Project Summary data.

As it turned out, both projects used approximately 1600 hours more than initially estimated (10% for A and 12% for B), and maximum weekly resources did not agree exactly with initial estimates. If these corrected figures for Ka and Yd are used in the analysis, then Ta, the date for acceptance testing, is 60 weeks instead of the actual 62 weeks for project A - an error of only 3% (Figure 3).

Note however that the corrected figures for project B yield a Ta of 44 weeks instead of the actual 54. This discrepancy is due in part to the extreme variance in actual development hours allocated to the project each week, especially towards the latter period (See figure 2). If an average maximum value of 425 hours per week is substituted for the absolute maximum, the projected completion date becomes 49 weeks, yielding an error of only 5 weeks.

It is clear from the analysis of this last data, that due to the size of the project and the effect small perturbations have on the prediction of results, that there is definitely a difference in the analysis of projects of the size being studied by the Laboratory and the large scale efforts reported in the literature. To demonstrate this point even further, consider the actual data in the curve in Figure 1. The significant drop in development activities during the weeks 21, 26 and 34 can be attributed to Thanksgiving, Christmas and Washington's Birthday, all holidays for the contractor. Thus our data is quite sensitive to holidays, employee illness, and project personnel changes.

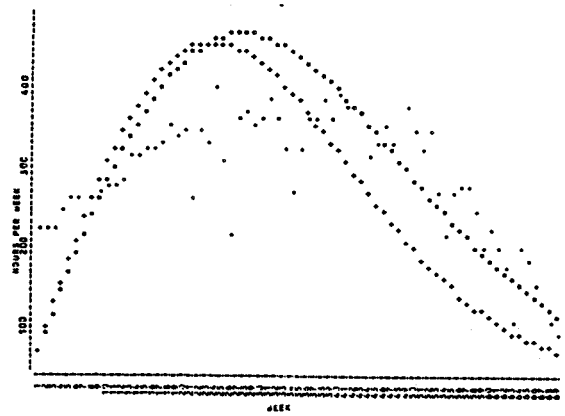
Predicting Progress

In order to test the predictability of the model, curve fitting techniques to the actual data were used. The Rayleigh curve can be rewritten as:

$$\ln\left(\frac{y}{t}\right) = \ln c - a*t^2$$

where

$$K = \frac{c}{2*a}$$



* - Least squares fit for all data points

\+ - Least squares fit using only points up to initial date of acceptance testing

. - Actual data

Figure 4. Project A - Least squares fit for resource data

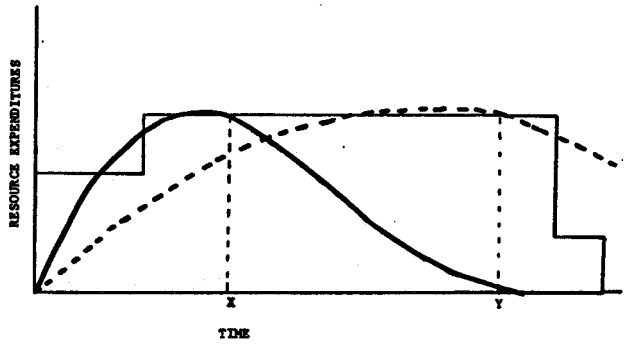
This equation can be used to derive the equation y=f(t) for the collected data (yi/ti, ti) using least squares techniques.

From this solution, figure 4 was plotted for project A. The * represents a best fit using all of the collected data points while the curve plotted with + represents a best fit based upon points up to the original point assumed to be acceptance testing (46 weeks for project A) to check the model's ability to predict completion.

Figure 5 summarizes the results. These are not very good, and Figure 6 is a possible explanation. On projects this small, the resource curve is mostly a step function. Thus assuming a Rayleigh curve estimate at point x results in an earlier. sharper decline while an estimate at y results in too little a decline. Starting with Norden's original assumptions that led to the Rayleigh curve as a predictor for large scale developments, current research is investigating variations to the basic curve so that it is "flatter" in its mid-range, and better approximates projects of this size.

	PROJECT A	PROJECT B
LEAST SQUARES FIT THROUGH ALL POINTS		
K, in hours	20,087	17,964
Ta, in weeks	57	61
LEAST SQUARES FIT USING POINTS UP TO ESTIMATED ACCEPTANCE TESTING DATE		
K, in hours	16,827	25,714
Ta, in weeks	49	61
ACTUAL PROJECT DATA		
K, in hours	17,742	16,543
Ta, in weeks	62	54

Figure 5. Estimating K and Ta using least squares fit.



——— Best fit based upon points up to time X

— — — Best fit based upon points up to time Y

Figure 6. Rayleigh curve estimation on medium scale projects

Forecasting of Components

As part of the reporting procedure, the Component Status Report gives manpower data on each component of the system, and the Component Summary gives the necessary size and time estimates. Therefore equations can be developed for each component in the system. Thus we are able to estimate whether any piece of the system is on schedule or has slipped.

At the present time, summary data can be printed on expenditures for each component in a project. In figure 7, CM is a subsystem of the project, and the other listed components are a sample of the components of CM. The above algorithm is now being investigated to see whether all components should be checked and some indication (such as a * next to the name) made if a component seems to be slipping from its estimated schedule. In the future, more accurate predictions of K from Ka will be investigated. How well the basic Rayleigh curve fits this data is also being studied. In addition, we would like to collect data from the analysis and maintenance sections at NASA to include the requirements, specifications and maintenance phases in the lifetime of each project.

COMPONENT	HOURS ON EACH ACTIVITY DESIGN	CODE	TEST	TOTAL	DATE LAST REFERENCED	ESTIMATED HOURS	ESTIMATED COMPLETION
CM			79	79	9/16/77		
CMARRO		12	9	21	7/ 8/77	15	7/18/77
CMARRP		6	3	9	5/18/77	14	6/30/77
CMASP		7	1	8	2/18/77	5	5/ 1/77
CMCMP		8	10	18	2/11/77	15	6/30/77
CMDRIV		2	3	5	3/11/77	10	6/19/77
CMDTCT	1	10	11	22	4/ 1/77	5	4/15/77

Figure 7. Resource data by components (Data collection on this project began after design phase completed, so little design time is shown.)

Putnam lists only two parameters affecting overall system development: total manpower needs and maximum manpower. What effects do other programming techniques have (if any) on the shape of this curve? For example, proponents of many methodologies, such as structured programming, predict a slower rise in the curve using the proposed techniques.

OTHER INVESTIGATIONS

Besides project forecasting, several other areas are under investigation. Some of these are briefly described in the following paragraphs.

Overhead

Overhead is often an elusive item to pin down. In our projects three aspects of development have been identified: programmer effort, project management, and support items (typing, librarians, clerical, etc.). In one project programmers accounted for about 80% of total expenditures with the support activities taking about one third of the remaining resources. In addition, only about 60% of all programmer time was accountable to explicit components of the system. The remaining time includes activities like meetings, traveling, attending training sessions, and other activities not directly accountable. As others have shown, this figure must be included in computing effective workloads in hours per week.

Error Analysis

One early investigation using the collected change reports, was to test the hypothesis of Belady and Lehman [1976]. By studying several large systems, they determined that for each release of a given system, the per cent of modules altered since the previous release was constant over time ("handling rate"). Since our own data was mostly data collected during integration testing, the extension of their results were tested in our own environment. In addition, besides the handling rate, we also wanted to investigate the report rate, or the rate at which changes were reported over time on the developing system.

Figure 8(a) shows this early evaluation, which clearly does not represent a constant handling rate. The maximum rate of handling modules occurs in the middle of the testing period.

One result which was surprising, however, is is the report rate of figure 8(b). This represents the number of change reports submitted each week. This figure did remain constant for almost the entire development time.

In order to test this second result further, data from a second project was plotted. It too had handling rates and report rates similar to the above project. This phenomenon will be studied in greater detail in the future..

SUMMARY

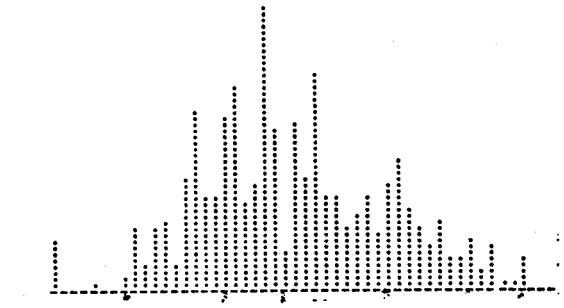
The major contribution of the Laboratory to the field of software engineering is the ability to collect the kind of detailed data currently unavailable, and collect it for a class of projects (medium scale) that has not yet been well analyzed. The finer level of monitoring and data collection can yield better analysis and understanding of the details of the development process and product. The medium scale size of the projects permit us to study more projects although it is clear that good data collection techniques are more important here than in larger projects because mistakes can have a much stronger impact. The large number of projects being compared also permit various software development parameters and techniques to be analyzed and compared with quantitative assessments by correlating data across several projects.

The current status of projects in the Laboratory have permitted us to begin reporting back to management the status of projects and to begin analyzing individual aspects of projects, checking their relationships to large scale project results found in the literature. The model of resource utilization via the Rayleigh curve is an important idea that is being investigated. Error rates and their causes are also under study. Since the Laboratory only started to collect data in December of 1976, and since most projects take from 12 to 18 months to complete, the first few projects are only now being completed; however, within the next 4 to 6 months, about four more

projects will be ready for analysis. This will allow for more careful comparisons with the data already collected.

ACKNOWLEDGEMENTS

We would like to acknowledge the contributions and cooperation of Mr. Frank McGarry, head of the Systems Development Section of NASA Goddard Space Flight Center. He has been instrumental in organizing the Laboratory and in interfacing with the contractor in order to see that the data is collected reliably and timely. We would also like to thank Computer Sciences Corporation for their patience during form development and their contributions to the organization and operation of the Laboratory.



(a) Handling Rate by week

(b) Report rate by week

Figure 8. Handling and report rate of project A.

REFERENCES

[Baker] Baker F. T., Structured programming in a production programming environment, International Conference on Reliable Software, Los Angeles, April, 1975 (SIGPLAN Notices 10, No. 6, 172-185).

[Basili & Zelkowitz] Basili V. and M. Zelkowitz, The Software Engineering Laboratory: Objectives, Proceedings of the Fifteenth Annual ACM Computer Personnel Research Conference, Washington D. C., August, 1977.

[Belady & Lehman] Belady L. A. and M. M. Lehman, A model of large program development, IBM Systems Journal 15, No. 3, 1976, 225-252.

[Held] Held G., M. Stonebraker, E. Wong, INGRES - a relational data base system, National Computer Conference, 1975, 409-416.

[Myers] Myers G., Software Reliability through composite design, Mason Charter, New York, 1975.

[Norden] Norden P., Use tools for project management, Management of Production, M. K. Starr (ed), Penguin Books, Baltimore, Md., 1970, 71-101.

[Putnam] Putnam L., A macro-estimating methodology for software development, IEEE Computer Society Compcon, Washington, D. C., September, 1976, 138-143.

[Putnam2] Putnam L., Private communication.

[Walston & Felix] Walston C. E. and C. P. Felix, A method of program measurement and estimation, IBM Systems Journal 16, No. 1, 1977, 54-73.

[Wolverton] Wolverton R. W., The cost of developing large scale software, IEEE Transactions on Computers 23, No. 6, June, 1974, 615-636.