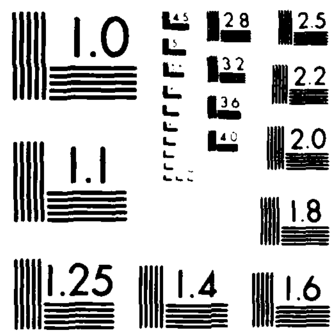
AD-A171 945 DESIGN ACTIVITY IN THE SOFTWARE COST REDUCTION PROJECT 1/1
(U) NAVAL RESEARCH LAB WASHINGTON DC A F NORCIO ET AL.
18 AUG 86 NRL-8974

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART

Naval Research Laboratory

Washington, DC 20375-5000 NRL Report 8974 August 18, 1986

Design Activity in the Software Cost Reduction Project

A. F. NORCIO AND L. J. CHMURA

Computer Science and Systems Branch
Information Technology Division

AD-A171 945

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED

1b RESTRICTIVE MARKINGS:

2a SECURITY CLASSIFICATION AUTHORITY:

2b DECLASSIFICATION/DOWNGRADING SCHEDULE:

3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.

4 PERFORMING ORGANIZATION REPORT NUMBER(S): NRL Report 8974

5 MONITORING ORGANIZATION REPORT NUMBER(S):

6a NAME OF PERFORMING ORGANIZATION: Naval Research Laboratory

6b OFFICE SYMBOL (If applicable): Code 7592

7a NAME OF MONITORING ORGANIZATION:

6c ADDRESS (City, State, and ZIP Code): Washington, DC 20375-5000

7b ADDRESS (City, State, and ZIP Code):

8a NAME OF FUNDING/SPONSORING ORGANIZATION: Surface Naval Weapons Systems Command

8b OFFICE SYMBOL (If applicable): Code 613

9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER:

8c ADDRESS (City, State, and ZIP Code): Washington, DC 20363

10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
63756A		AMC	DN155-191

11 TITLE (Include Security Classification): Design Activity in the Software Cost Reduction Project

12 PERSONAL AUTHOR(S): Norcio, Anthony F. and Chmura, Louis J.

13a TYPE OF REPORT: Interim

13b TIME COVERED: FROM 8/85 TO 2/86

14 DATE OF REPORT (Year, Month, Day): 1986 August 18

15 PAGE COUNT: 23

16 SUPPLEMENTARY NOTATION:

17 COSATI CODES

FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Software technology evaluation; Data collection; Software design activity; Software Cost Reduction (SCR) project

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Since 1978, the goal of the Software Cost Reduction (SCR) project has been to demonstrate the effectiveness of certain software engineering techniques for developing complex software. The application is the redevelopment of the operational flight program for the A-7E aircraft. Also since then, the Software Technology Evaluation (STE) project has been monitoring SCR project activity in order to provide an objective evaluation of the SCR methodology. SCR project activity data are collected from SCR personnel on a weekly basis. Over 55,000 hours of SCR design, code, test, and other activity data have been captured and recorded in a computer data base. Analyses of SCR module design data show that there are parameters that can be used to characterize and predict design progress. One example is the ratio between cumulative design discussing and cumulative design creating activities. This ratio is referred to as the Progress Indicator Ratio (PIR) and seems to be an accurate metric for design completeness. This and other results suggest that discussion activity among software engineers may play a major role in the software design process and may be a leading indicator of design activity progress.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS

21 ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL: Anthony F. Norcio

22b TELEPHONE (Include Area Code): (202) 767-3249

22c OFFICE SYMBOL: Code 7592

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

*U.S. Government Printing Office: 1985—507-047

CONTENTS

DESIGN ACTIVITY IN THE SOFTWARE COST REDUCTION PROJECT

INTRODUCTION

This report presents the results of an investigation of design activities of the software engineers working on the Software Cost Reduction (SCR) project. One purpose of this study is to offer insights into understanding the design process of complex software. A second purpose is to identify parameters that characterize and predict design progress. The data analyses suggest that at least one parameter does characterize and predict design progress under the SCR approach.

The Software Cost Reduction Project

Since 1978, the Naval Research Laboratory in cooperation with the Naval Weapons Center has been redeveloping version 2 of the operational flight program for the A-7E aircraft [1]. Software engineering techniques such as formal requirements specification, information hiding [2], abstract interfaces [3], and cooperating sequential processes [4] are being used. This effort is referred to as the SCR project.

The goals of the SCR project are to (a) demonstrate the feasibility of using selected software engineering techniques in developing complex, real-time software; and (b) provide a model for software design. The claimed advantage of the selected software engineering techniques is that they can facilitate the development of easy-to-change software. Heninger et al. [5] provide complete discussion of the project's software requirements. Britton and Parnas [6] give a detailed description of the module design structure.

The Software Technology Evaluation Project

The goal of the Software Technology Evaluation (STE) project is to evaluate alternative software development technologies.* The approach is to monitor, evaluate, and compare software development technologies used in different software projects. The monitoring and evaluating processes consist of goal-directed data collection and analyses techniques [7]. One of the tasks of the STE project is to provide the basis for an objective evaluation of the methodology used in the SCR project. The two projects are, however, separate research investigations each with its own goals, staff, and funding.

DATA COLLECTION

Since 1978, all SCR project engineers have been required to submit weekly reports on their project activity hours. The activity data are collected on a form called the Weekly Activity Report, the current version of which is presented in Chart 1. The boxes on the form represent different project activities.

A submitted report is usually rather sparse; typically, it has only a few boxes marked with hours spent on project activities during the week. A copy of a completed report form is presented in Chart 2. Once a weekly activity report is given to STE project personnel, it is validated and entered into a computer data base. An instruction sheet explaining how to report weekly activity is provided to each SCR engineer.

Manuscript approved March 5, 1986.

*This work is currently funded by the DoD STARS Program as Measurement Area Task G-06

SCR Project: Weekly Activity Report

Your name ______________ Date Friday_______________

ACTIVITY AREA	ACTIVITY HOURS: Design				Pseudo Code			EC/C/TC Code				Test		
	Creating	Discussing	Peer Reviewing	Formal Reviewing	Creating	Discussing	Peer Reviewing	Creating	Discussing	Peer Reviewing	Programmer Testing	Preparation	Conducting	Reviewing Results
Software Structures														
Module Guide														
Uses Hierarchy														
Software Modules														
Hardware Hiding														
Extended Computer														
Device Interface														
Behavior Hiding														
Function Driver														
Shared Services														
Software Decision														
Application Data Types														
Physical Model														
Filter Behavior														
Data Banker														
System Generation														
Software Utility														

(See reverse side)

Front Side

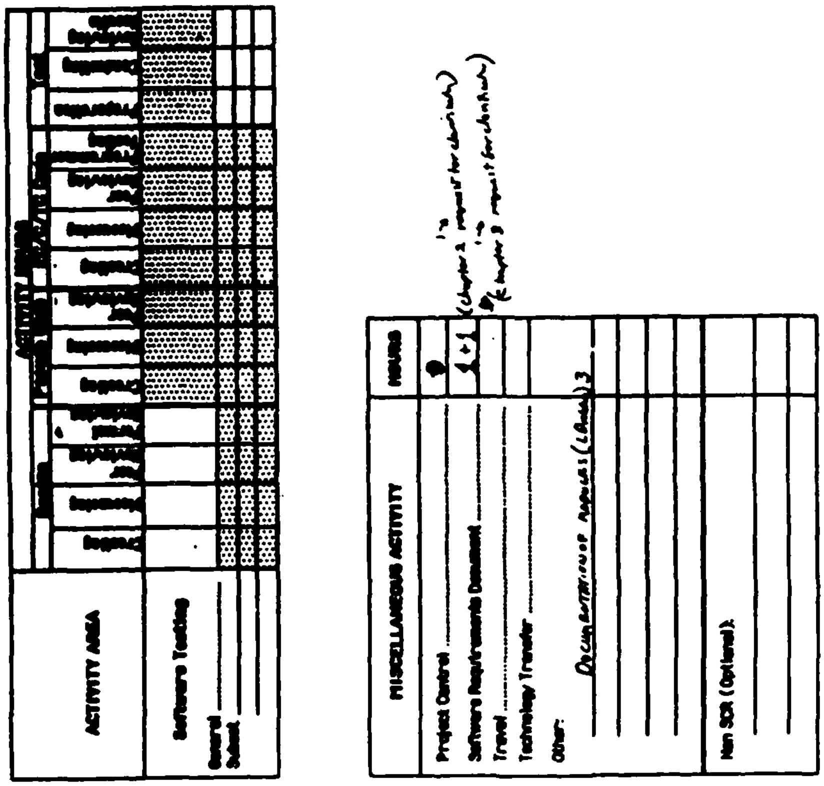
ACTIVITY AREA	ACTIVITY HOURS: Design				Pseudo Code			EC/C/TC Code				Test		
	Creating	Discussing	Peer Reviewing	Formal Reviewing	Creating	Discussing	Peer Reviewing	Creating	Discussing	Peer Reviewing	Programmer Testing	Preparation	Conducting	Reviewing Results
Software Testing														
General														
Subset														

MISCELLANEOUS ACTIVITY	HOURS
Project Control	
Software Requirements Document	
Travel	
Technology Transfer	
Other:	
Non SCR (Optional)	

(8 May 1985)

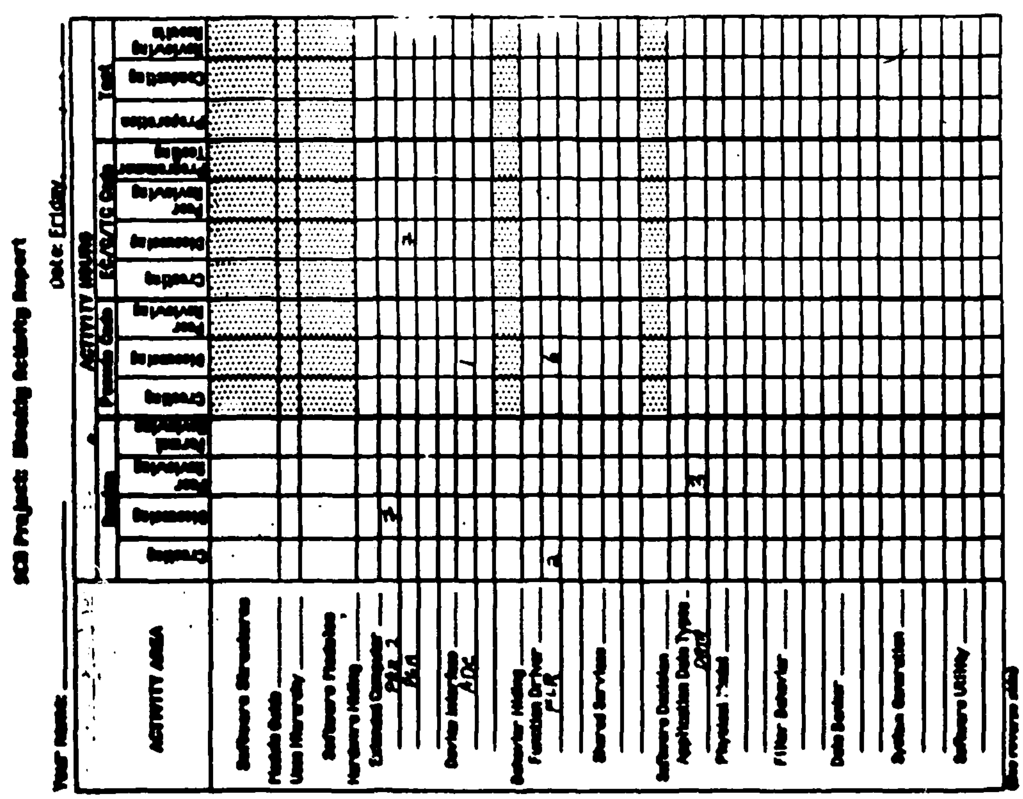
Reverse Side

Chart 1 — Weekly Activity Report form



(8 May 1985)

Reverse Side



Front Side

Chart 2 — Completed Weekly Activity Report form

Module Development Activities

The front page of the Weekly Activity Report form is primarily used to record hours spent on module development activity, where module means information hiding module [2]. As can be seen in Chart 1, SCR development activity hours are captured for each engineer by a specific module within the hierarchy (e.g., Device Interface) and by design, code, and test categories. Space is provided for project personnel to supply the names of the modules below the first two levels.

The primary product of design activity is the development of a module interface specification. A typical interface specification for the Device Interface module [8] is presented in Chart 3. Design activity is reported as hours devoted to design creating, design discussing, design peer reviewing, and design formal reviewing activities. *Design creating* activity is time devoted to thinking about a design including redesigning or documenting. *Design discussing* activity is time devoted to discussing design issues via a computer message or directly with a colleague to assist with the design. *Design peer reviewing* activity is time devoted to reading or commenting on (informally) design documentation produced by another project member in order to assist with the design. *Design formal reviewing* activity is time spent in a formal design review, typically at the Naval Weapons Center, which maintains the current operational flight program.

Coding activity is reported as hours devoted to pseudocode, Extended Computer code,* C code and TC code† activities. Pseudocode activity is further reported as hours devoted to code creating, code discussing, and code peer reviewing activities. EC code, C code, and TC code activities are reported as hours devoted to code creating, code discussing, code peer reviewing, and code programmer testing activities. *Code creating, code discussing,* and *code peer reviewing* activities have definitions similar to their counterparts. *Code programmer testing* activity is time devoted by programmers to computer-based evaluation of their own code to convince themselves of its correctness.

Test activity is reported as hours devoted to test preparation, test conducting, and test reviewing results activities. *Test preparation* activity is time devoted to creating, discussing, and reviewing plans and procedures for computer-based testing of a module prior to formal subset testing. *Test conducting* activity is time devoted to set up and execution of module test procedures on a computer. *Test reviewing results* activity is time devoted to analyzing, discussing, and documenting results of a module test.

Other Activities

The back page of the weekly Activity Report form is used to record hours spent on software testing and miscellaneous activities. *Software testing* activity is reported as hours spent on general issues of computer-based testing and on testing of system subsets. *Miscellaneous* activity is reported as hours spent on activities not included in any of the above definitions.

OVERVIEW OF SCR PROJECT ACTIVITIES

From January 1978 to February 1985, over 55,000 activity hours have been reported. Experiments have been performed to provide reasonable assurance that the reported hours accurately reflect project activity and are appropriately categorized [9]. Figure 1 represents the monthly accumulation of hours expended on all activities. Figure 2 shows the monthly accumulation of hours expended in the top level categories. Software Structures (SS) effort is time spent defining and documenting hierarchical module structure in the A-7E module guide [6]. Software Modules (SM) effort is time devoted primarily to specifying and implementing modules. Software Testing (ST) effort is time spent on validation testing of subsets, and Miscellaneous (MISC) is time spent on all other activities such as travel and project control. Most SCR work so far has concentrated on SM development.

*The Extended mputer is one of the modules of the program. EC code is consists of invocations of programs on this module's interface.

†TC code is the assembly language code for the IBM System 4 PI model TC-2 computer. The operational flight program runs on this machine.

DI.WOG: WEIGHT ON GEAR SENSOR

1. Introduction

The weight on gear device is a sensor that detects whether or not the aircraft is resting on its landing gear. This data can be used to infer whether or not the aircraft is airborne.

2. INTERFACE OVERVIEW

2.1. ACCESS PROGRAM TABLE

Program	*Parameters*	*Description*	*Undesired events*
+G_WEIGHT_ON_GEAR+	p1: boolean; O	!+WOG+!	None

3. LOCAL TYPE DEFINITIONS None.

4. DICTIONARY

!+WOG+! TRUE iff weight on landing gear detected.

5. UNDESIRED EVENT DICTIONARY None.

6. SYSTEM GENERATION PARAMETERS None.

Chart 3 — Sample design specification

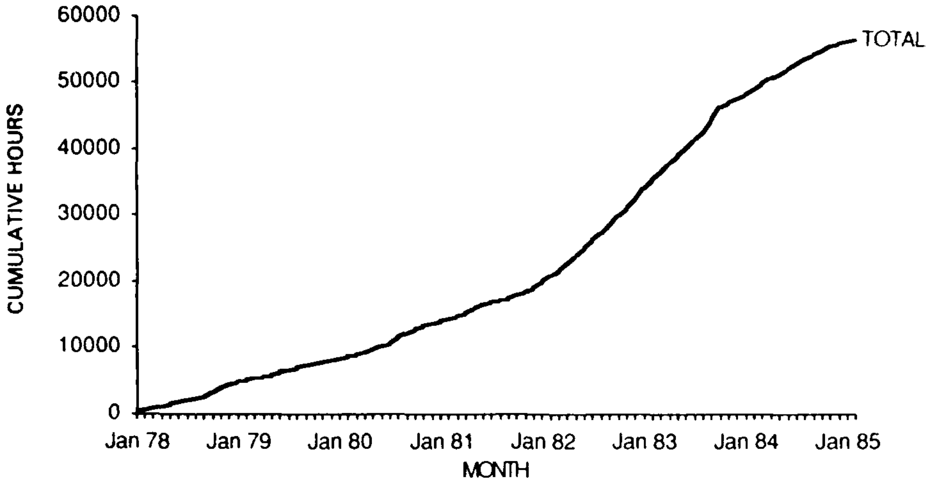


Fig. 1 — SCR activity

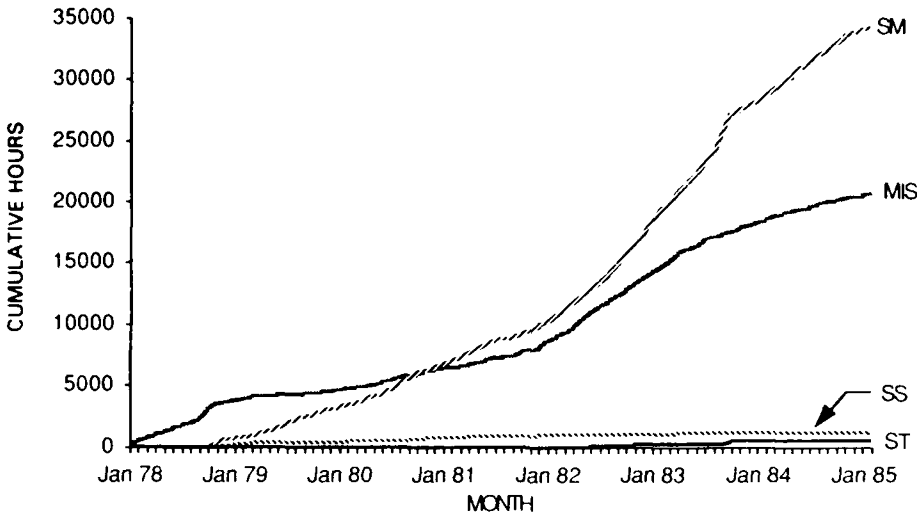


Fig. 2 — SCR area activities

Figure 3 shows the monthly accumulation of hours expended in the Software Modules category on design, code (including pseudocode), and test activities. Over 75% of all reported software module activity is module design including redesign. This is consistent with the emphasis in the SCR methodology on extensive design with the expectation of significant reductions in coding, testing, and maintenance efforts [10].

There are three categories of first-level SCR modules: Hardware Hiding modules, Behavior Hiding modules, and Software Decision modules [6]. These, in turn, include ten categories of second-level modules, listed in Table 1. Each of the second-level modules is organized into several submodules (third-level modules), and some of these are further modularized. The EC module, with seven levels of submodules, has the deepest module structure.

Six of the second-level modules have accumulated more than 1000 hours of activity; these are EC, DI, FD, SS, AT, and PM. The six also have complete module interface specifications that are baseline or nearly baselined. In Figures 4 through 9, the monthly accumulation of hours expended on total activity and on design, code, and test activities is presented for each module.* Only the EC and DI modules have appreciable amounts of coding and testing activities.

*Vertical lines in Figures 4 through 21 represent the dates on which baseline interfaces specifications for the respective modules were released. The absence of these lines on a specific plot indicates that no baseline documents have been released for that module.

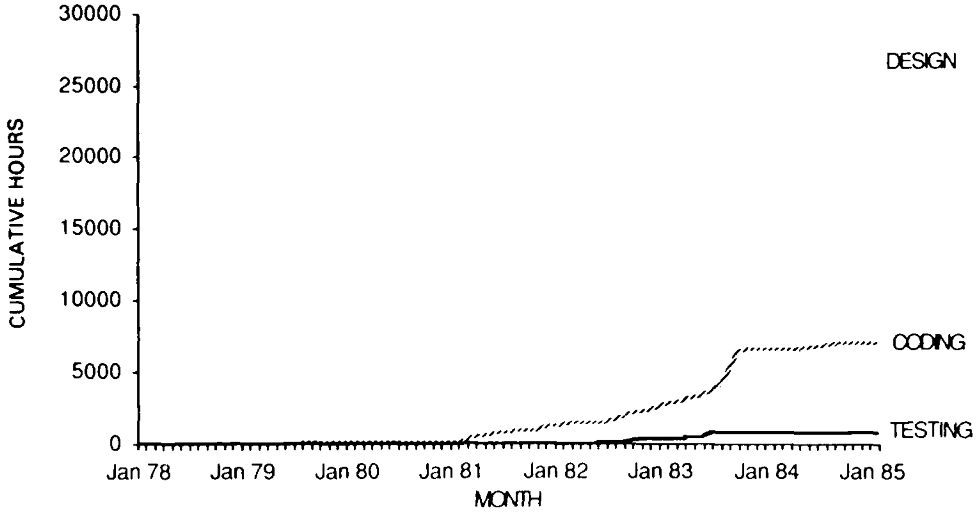


Fig. 3 — Software module activities

Table 1 — Abbreviations and Names of Second-Level Software Modules

Abbreviation	Name
AT	Applications Data Type
DB	Data Banker
DI	Device Interface
EC	Extended Computer
FD	Function Driver
FLT	Filter
PM	Physical Model
SG	System Generation
SS	Shared Services
SU	System Utilities

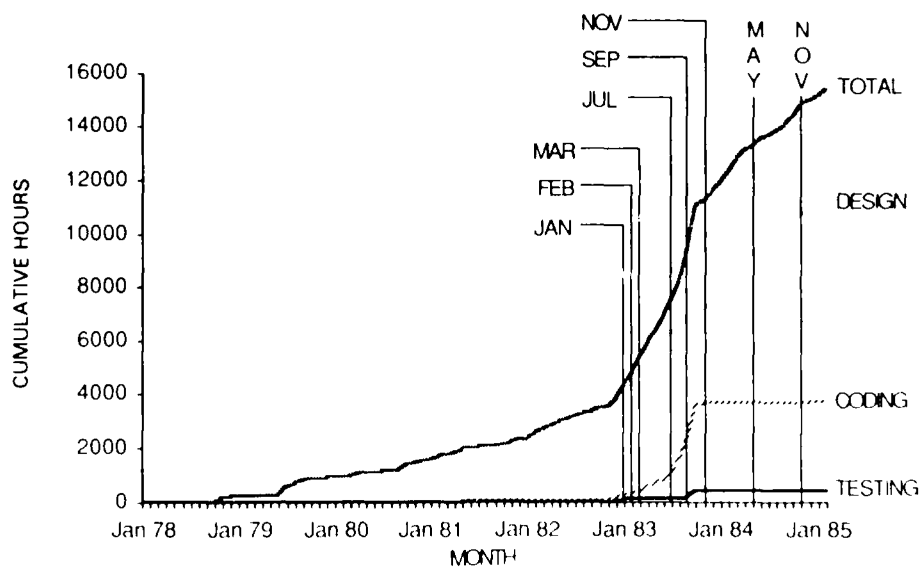


Fig. 4 — Extended computer activities

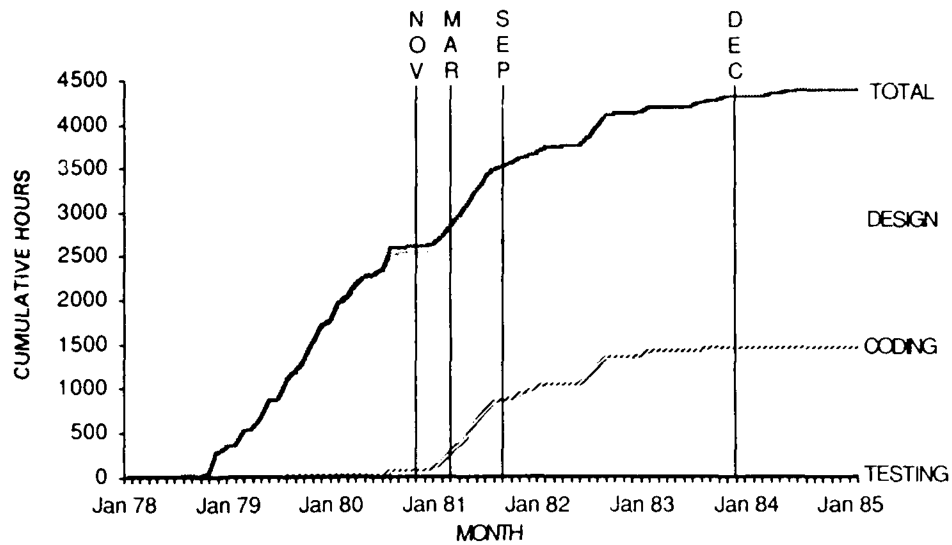


Fig. 5 — Device interface activities

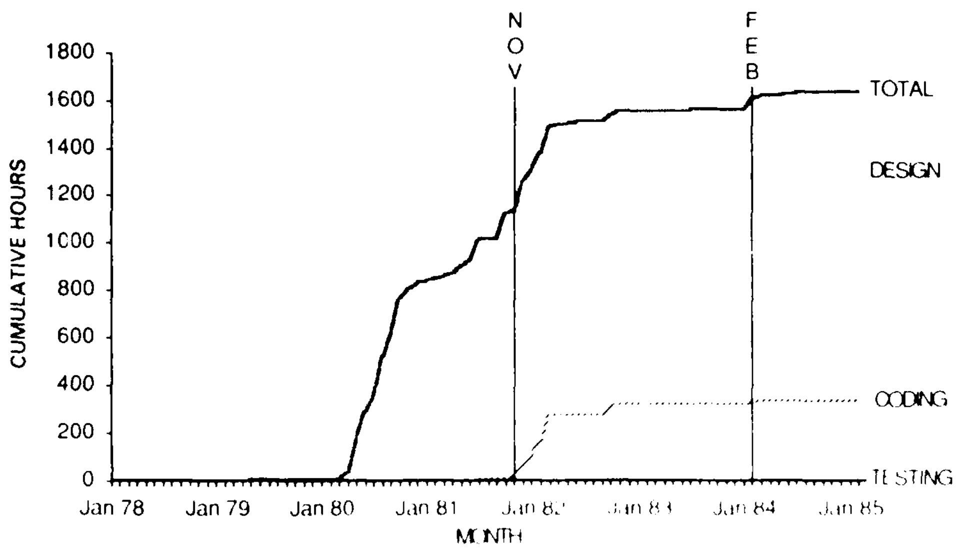


Fig. 6 — Function driver activities

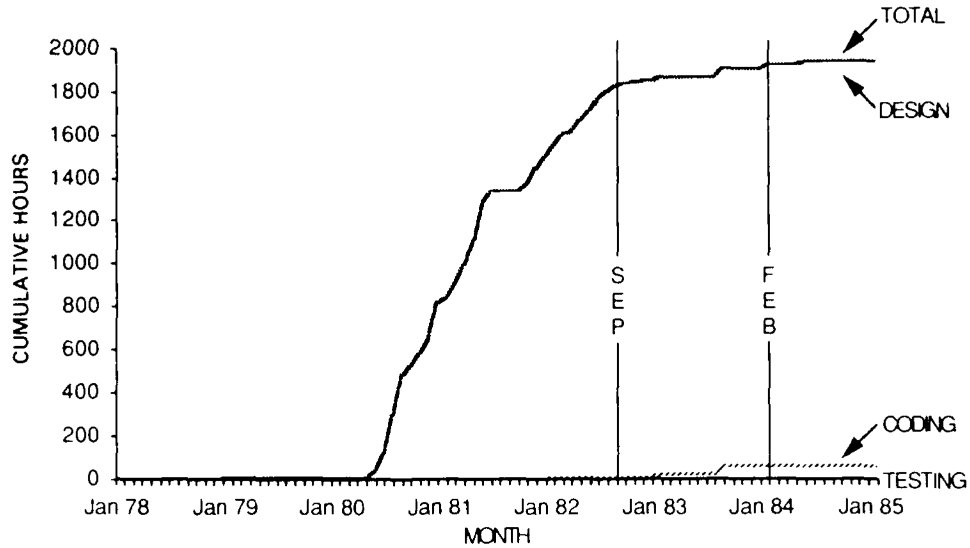


Fig. 7 — Shared services activities

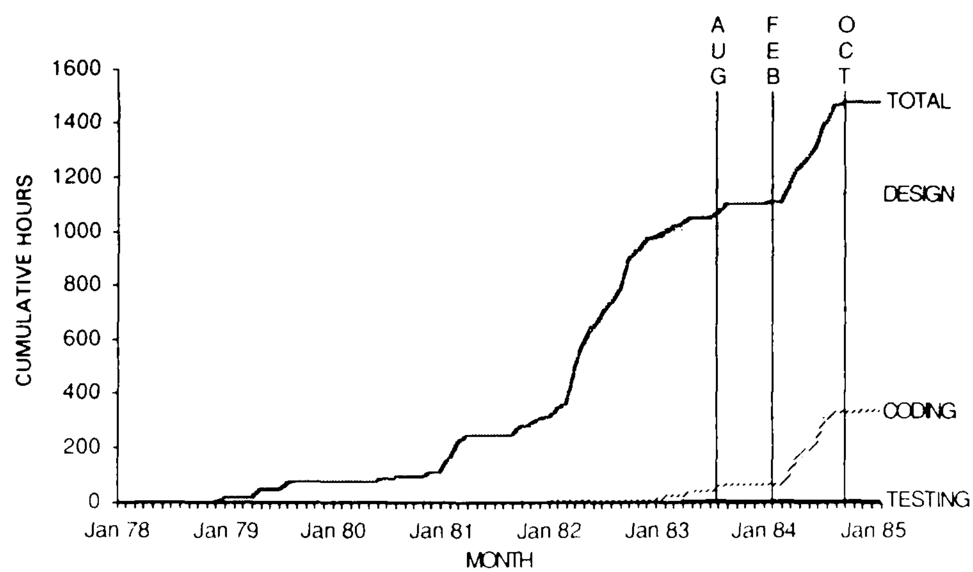


Fig. 8 — Applications data type activities

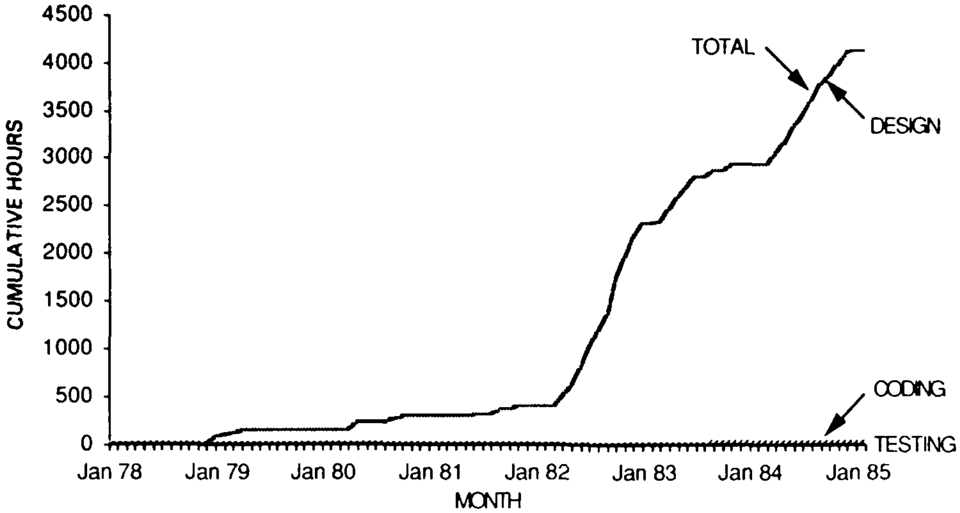


Fig. 9 — Physical model activities

ANALYSES OF MODULE DESIGN DATA

As mentioned above, the SCR project emphasizes careful design which is reflected by the fact that design activity accounts for over 75% of all reported software module activity. One of the purposes of the data analyses was to identify parameters that characterize the design processes and offer predictive capabilities concerning them. Plots were constructed in order to characterize monthly hours expended on the subactivities of module design: design creating (DC), design discussing (DD), design peer reviewing (DR), design formal reviewing (DF), as well as total design (D). Unfortunately, characteristic patterns were not readily apparent.

Subsequently, it was decided to examine the accumulation of hours expended on total design subactivities. This approach is considered appropriate because each data point reflects the history of design activities up to that point in time. Thus, the cumulative total design hours for a module is defined by:

$$\mathrm{CumD}_n = \sum_{i=1}^{n} D_i,$$

where D_i is the monthly total of all design activities on a module for month i. (Note that D_i includes design activity on all submodules of the module.) Because data are available for all the months between January 1978 and February 1985, n has values from 1 to 86. Cumulative design creating hours for a module is defined by:

$$\mathrm{CumDC}_n = \sum_{i=1}^{n} DC_i,$$

where DC_i is the monthly design creating subactivity for a module (including all submodules) for month i. Again, n has values from 1 to 86. Similar definitions apply for CumDD_n and CumDR_n. Figures 10 through 15 show the significant cumulative design activities for each module.

An earlier study [9] highlighted the fact that ratios between activity categories provide valid and potentially useful metrics of SCR project activity. STE Project personnel intuitively suspected that ratios between activity categories could provide descriptive features of the SCR methodology that might be generally applicable to software design. Consequently, six ratio series between CumDC_n, CumDD_n, and CumDR_n were computed. For example, the ratio between cumulative design discussing and cumulative design creating is defined as:

$$(CumDD/CumDC)_n = \frac{CumDD_n}{CumDC_n},$$

where n has values from 1 to 86. The other five ratios, similarly defined, are $(CumDC/CumDD)_n$, $(CumDC/CumDR)_n$, $(CumDD/CumDR)_n$, $(CumDR/CumDC)_n$, and $(CumDR/CumDD)_n$. These ratios were correlated with $CumD_n$ over the 86 reporting months. Pearson correlation coefficients [11] are presented in Table 2. Next, for each module the ratio between monthly DC, DD, and DR were computed (e.g. DC_n/DD_n, DC_n/DR_n, and so on) and correlated with the total monthly Ds and with the monthly CumDs. These coefficients are presented in Tables 3 and 4.

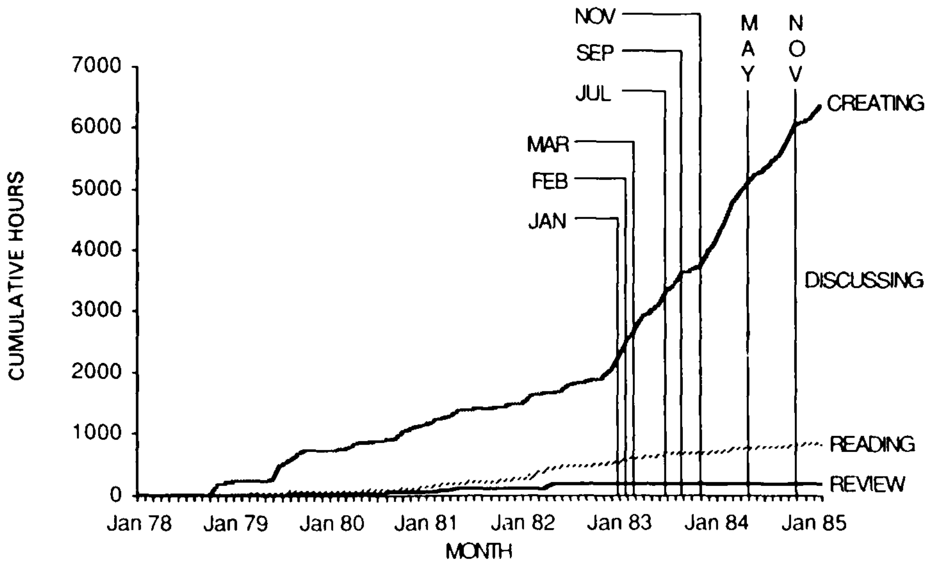


Fig. 10 — Extended computer design activities

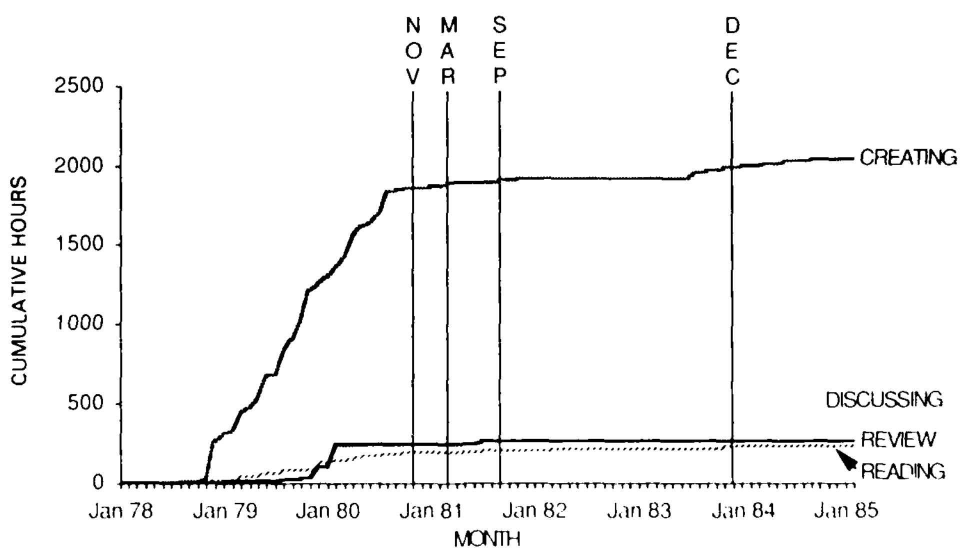


Fig. 11 — Device interface design activities

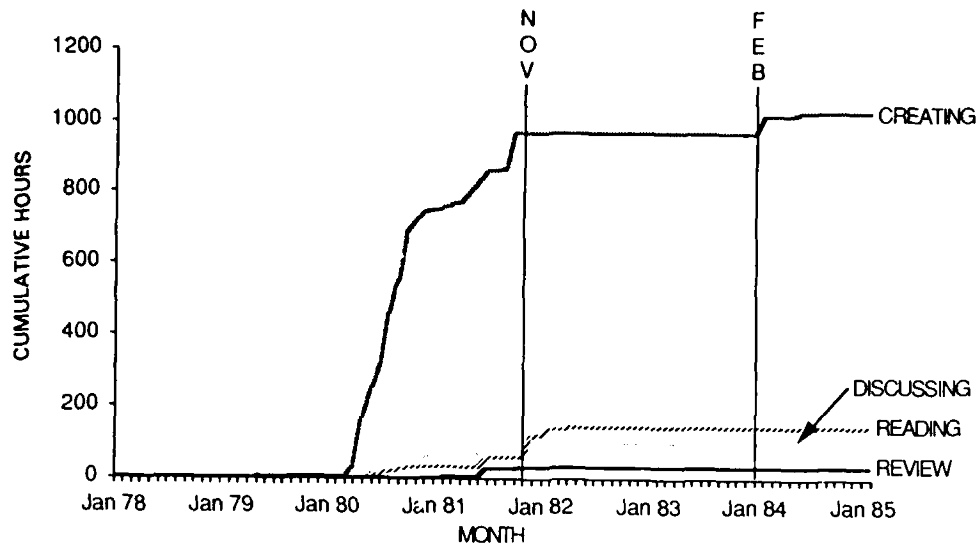


Fig. 12 — Function driver design activities

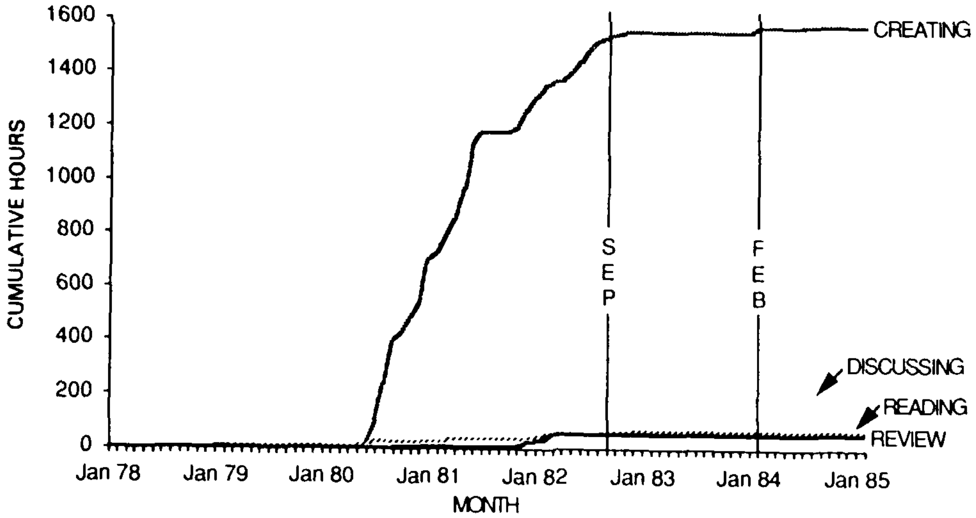


Fig. 13 — Shared services design activities

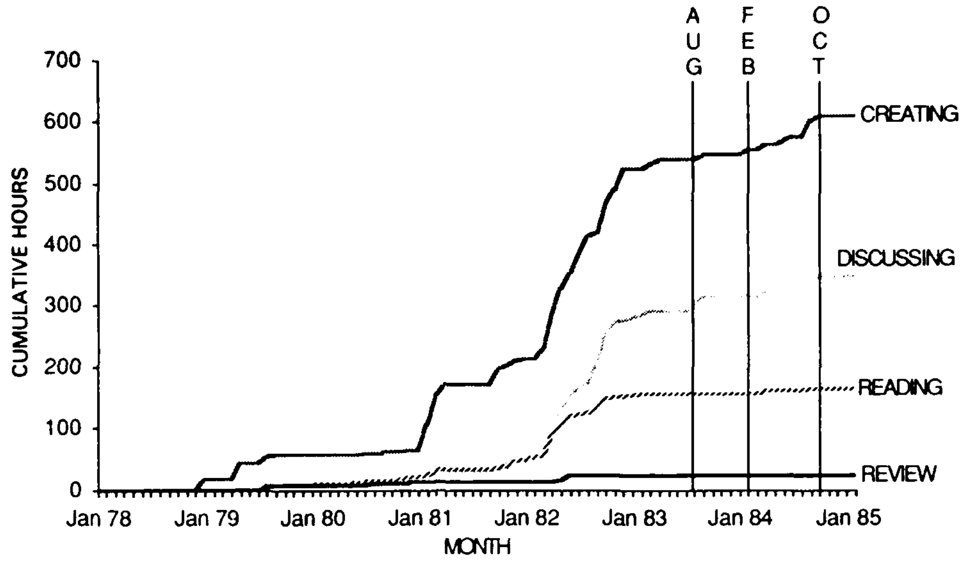


Fig. 14 — Applications data type design activities

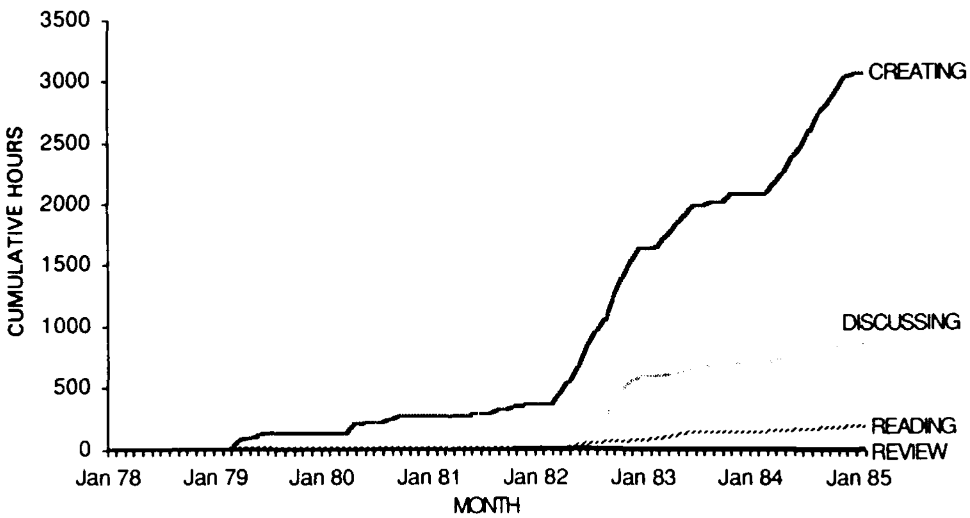


Fig. 15 — Physical model design activities

Table 2 — Pearson Correlation Coefficients Between CumD and Cum. Ratios

Module	$\frac{CumDC}{CumDD}$	$\frac{CumDC}{CumDR}$	$\frac{CumDD}{CumDC}$	$\frac{CumDD}{CumDR}$	$\frac{CumDR}{CumDC}$	$\frac{CumDR}{CumDD}$
AT	−0.4034	0.1245[a]	0.9774	0.9609	0.7483	−0.2006[a]
DI	0.0560[a]	−0.1998[a]	0.6574	0.1002[a]	0.8324	0.8124
EC	−0.3785	−0.3091	0.9492	0.6530	0.5436	0.5436
FD	0.7565	−0.0897[a]	0.9482	−0.0252[a]	0.1001[a]	0.9575
PM	−0.4090	−0.4144	0.9052	0.1429[a]	0.9235	0.3801
SS	0.9181	0.8665	0.5156	0.8408	−0.3675	−0.0568

[a]Not significant at the $p \geq .005$ level.

Table 3 — Pearson Correlation Coefficients Between D and Monthly Ratios

Module	DC/DD	DC/DR	DD/DC	DD/DR	DR/DC	DR/DD
AT	0.3099	0.4031	0.3548	0.3048	0.3664	0.1572[a]
DI	0.3759	0.4829	0.1040[a]	0.2044[a]	0.0227[a]	0.2075[a]
EC	−0.0008[a]	0.3972	0.3401	0.5396	0.0040[a]	−0.0998[a]
FD	0.6470	0.5190	0.1305[a]	0.4813	0.1042[a]	0.3796
PM	0.3083	0.5845	0.3175	0.6581	0.0461[a]	0.3491
SS	0.6509	0.6040	0.0461[a]	0.4309	0.1248[a]	0.2578

[a]Not significant at the $p \geq .005$ level.

Table 4 — Pearson Correlation Coefficients Between CumD and Monthly Ratios

Module	DC/DD	DC/DR	DD/DC	DD/DR	DR/DC	DR/DD
AT	0.0532[a]	0.1667[a]	0.1169[a]	0.0843[a]	−0.0177[a]	−0.1287[a]
DI	−0.1693[a]	−0.1414[a]	0.0344[a]	−0.0879[a]	0.0683[a]	0.0352[a]
EC	−0.0932[a]	0.3933	0.3467	0.5236	−0.0248[a]	−0.0886[a]
FD	−0.0034[a]	0.0603[a]	0.1409[a]	0.0293[a]	0.1032[a]	0.1282[a]
PM	0.4034	0.3568	0.1182[a]	0.2205	0.170[a]	0.2582
SS	0.0547[a]	−0.0447[a]	0.1556[a]	−0.0531[a]	−0.0177[a]	0.0815[a]

[a]Not significant at the $p \geq .005$ level.

RESULTS

An examination of the correlation coefficients reveals that the ratio $(CumDD/CumDC)_n$ correlates consistently well with $CumD_n$, as shown in Table 2.* This relationship is evident from the plots of the ratios. In Fig. 16, the monthly ratio for $(CumDD/CumDC)_n$ is plotted for the EC module. Comparing this with Fig. 10, one can see that design activity surges are characterized by prior or concomitant dramatic increases in this ratio. When this ratio remains constant, it is an indication that design activity has stabilized. Increases in this ratio seem to indicate progress. Consequently, we refer to this as the progress indicator ratio (PIR).

Although the EC module is extremely large and complex, the relationship seems strong. A large jump in design activity follows the large rise of the PIR. However, design activity for this module is not quite stabilized, and the late downward trend in the ratio indicates increasing creating time relative to discussing time.

The same patterns are also present for the DI module. As can be seen in Figs. 12 and 17, the dramatic increase in design activity follows a dramatic increase in the progress indicator ratio. This same relationship holds for the FD, SS, AT, and PM modules. See Figs. 18 through 21.

Coefficients of determination (r^2), as defined in Ref. 11, between $CumD_n$ and $(CumDD/CumDC)_n$ are presented in Table 5 for each module. This ratio seems to explain a high percentage of the variation of $CumD_n$.

The analyses provide supporting evidence that the ratio $(CumDD/CumDC)_n$ is an important measure of design activity progress in developing modules for complex software. When the PIR becomes constant, design activity appears to be at a very low level or even nonexistent. When this ratio increases, design activity increases dramatically. The relationship between this ratio and $CumD_n$ is the strongest of all the possible relationships examined in this study. In at least one module, this ratio can explain over 95% of the variation in $CumD_n$. In the remaining modules, variations in $(CumDD/CumDC)_n$ can explain a surprisingly high degree of the variations in $CumD_n$

*The probability of finding this significant result is not increased because the same analyses were conducted over different data sets.

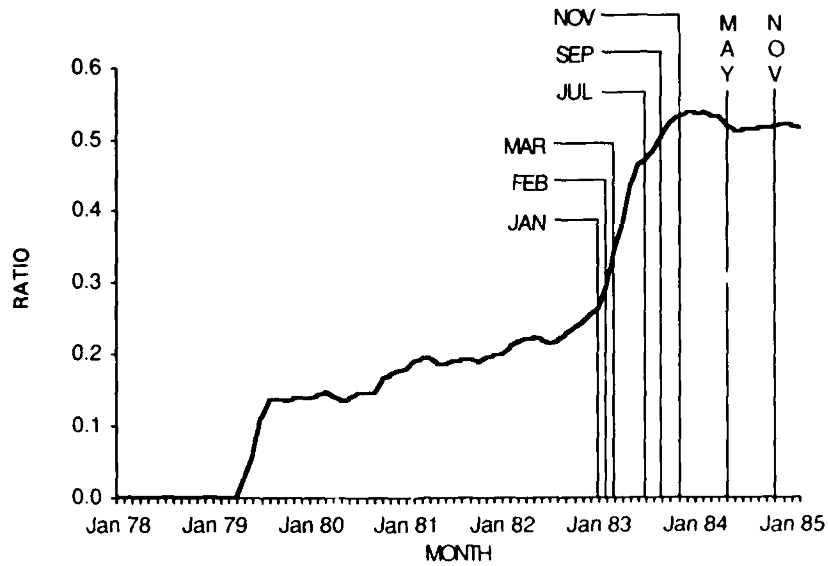


Fig. 16 — Progress indicator ratio for extended computer module

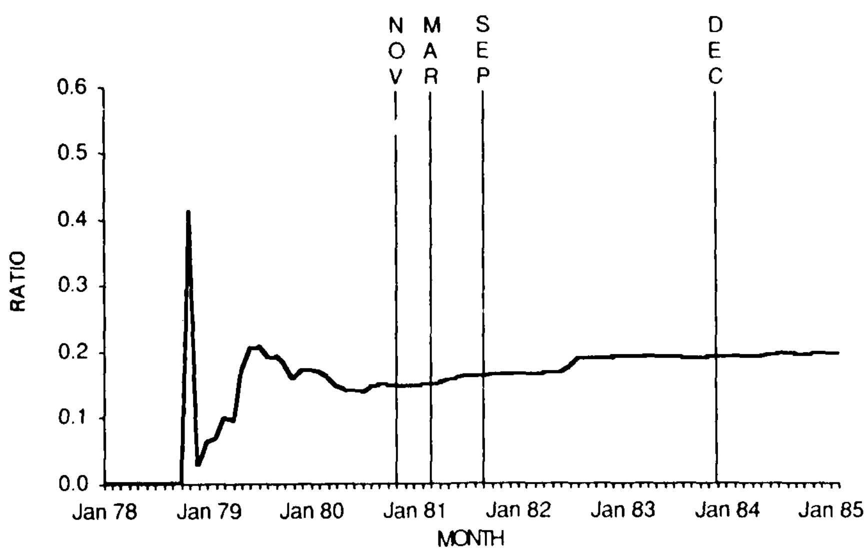


Fig. 17 — Progress indicator ratio for device interface module

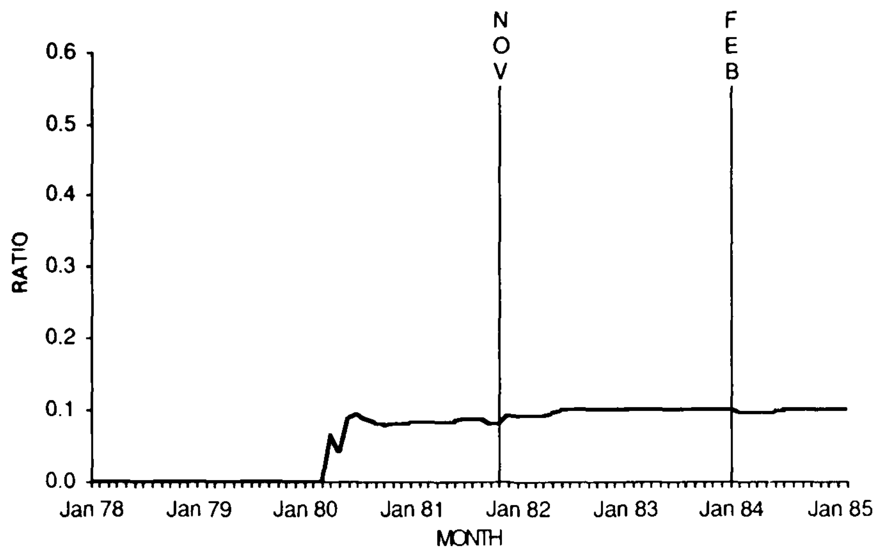


Fig. 18 — Progress indicator ratio for function driver module

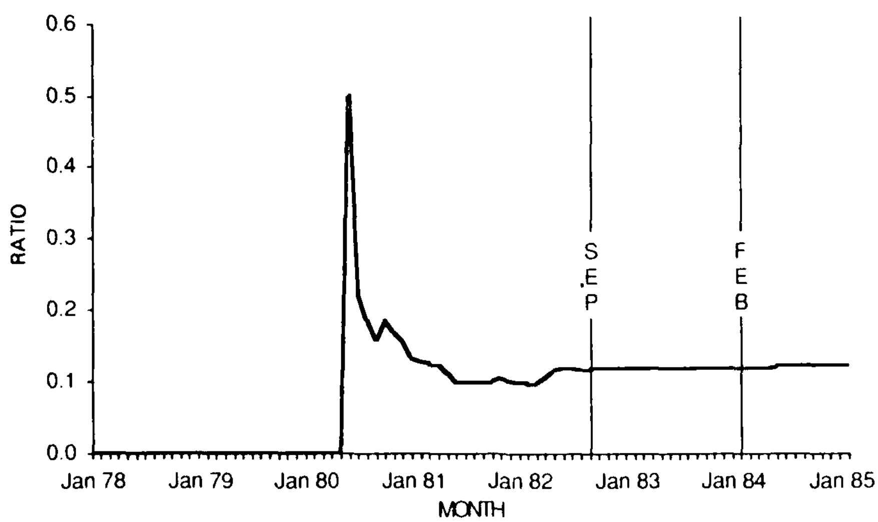


Fig. 19 — Progress indicator ratio for shared services module

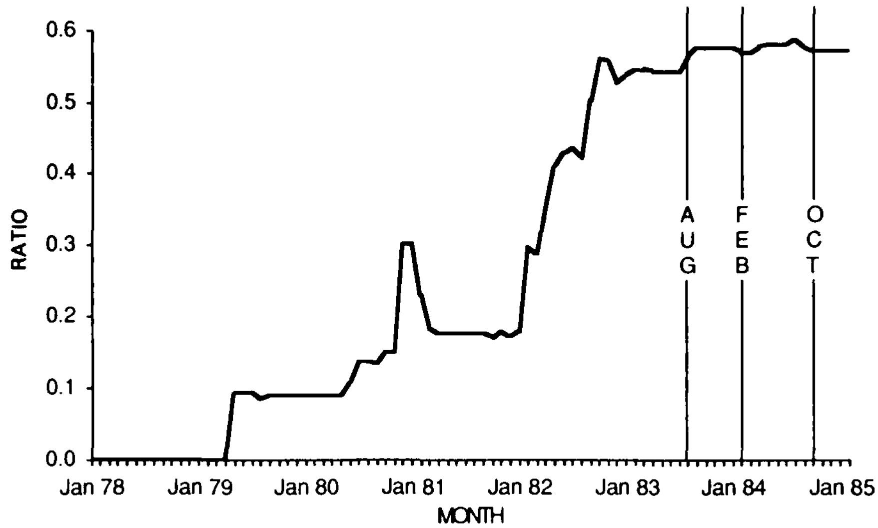


Fig. 20 — Progress indicator ratio for applications data type module

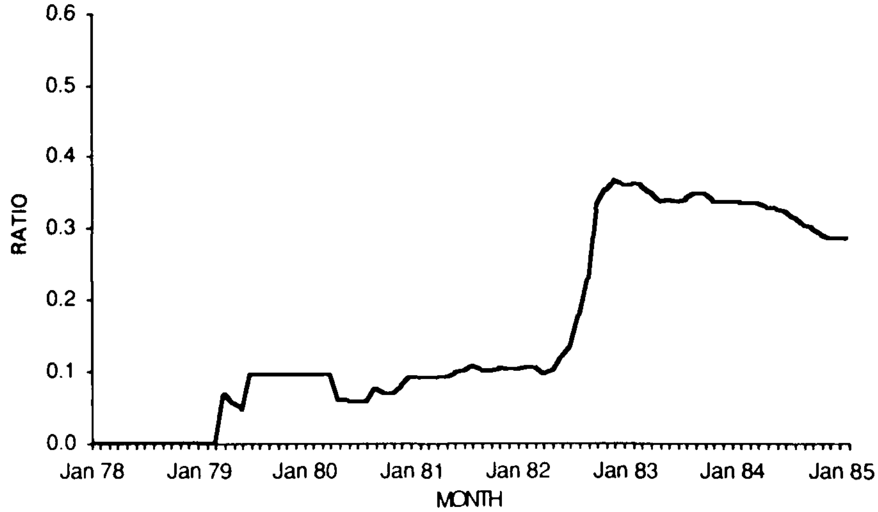


Fig. 21 — Progress indicator ratio for physical model module

Table 5 — Coefficients[a] of Determination (r^2) between CumD and CumDD/CumDC

Module	r^2
AT	0.9552
DI	0.4322
EC	0.9010
FD	0.8992
PM	0.8194
SS	0.2658

[a]All are significant at the $p \geq 005$ level

CONCLUSIONS

A natural conclusion is that discussion between software designers is a critically important factor in the design of information-hiding modules for complex software. When the release dates for specification baseline (e.g., Ref. 8) are examined with the PIR, the PIR seems to be indicating the completeness of the baseline specifications. When a baseline appears before this ratio rises sharply or during a sharp rise, the baseline is probably far from complete. Abstract interface specifications would seem to become reasonably stable only after a sharp rise and settling of this ratio. Plotting this ratio over time may provide for the software manager a meaningful tool with which to track design progress. If the PIR has not surged and stabilized, the design is probably not finished irrespective of personnel claims and published baseline documents.

In addition, the PIR has an attractive property not found in a monthly plot of $CumD_n$. The range of the y-axis is constant over time and over other modules and projects. Therefore, it is possible to compare design progress on one module or project to another by using this ratio. The PIR does, however, have one possible negative property. Because it involves cumulative sums, the accumulation of earlier design hours can dampen the impact of later variations in design activity. The PIR for the EC module indicates, however, that this possible flaw may be more theoretical than practical.

There is no claim that the PIR is a measure of design completeness. There are clearly other reasons why design activity on a specific software module may have stabilized; for example, personnel may have shifted work to another module or they may have been vacation. However, the PIR ratio seems to indicate when work on a piece of software is definitely not finished. If design completion is claimed prior to a rise and settling in this ratio, there is probably more work that needs to be done on that module.

It is necessary that this analysis be replicated on other large scale software development projects to determine whether the PIR behaves similarly in other software development environments using different methodologies. It is intuitively appealing that discussion between project members necessarily enhances the design of software modules. It would also be useful to quantify the relative surges in the PIR. That is, there is practical importance in knowing that a given percentage increase in the PIR is customarily followed by a predictable percentage increase in design activity. This, too, requires replicating these analyses in several different software design environments. Unfortunately, these data are difficult to collect and it is, perhaps, even more difficult to validate their accuracy.

Finally, it is logical to examine coding data for these relationships. It seems reasonable to accept the importance of discussion in the design process. Its importance in the coding and testing processes is not as clear. These data do exist in the SCR data base and plans are under way to examine them.

ACKNOWLEDGMENTS

The authors owe a special debt of gratitude to Dr. Davis Weiss who first suggested that activity ratios could provide useful measures of design activities. In addition, through several readings of this report, he offered many helpful suggestions and comments. Special mention must also be given to Ms. Kathryn Kragh who validated reported SCR activity data, entered them into the computer database, and checked the accuracy of each entry. Without her diligence, the data analyses could have possibly taken years. The authors also thank Mr. Jeff Sabat who prepared many data plots as well as Dr. John O'Hare and Mr. Paul Clements for their technical advice and suggestions.

REFERENCES

1. P.C. Clements, "Software Cost Reduction Through Disciplined Design," *1984 NRL Review*, pp. 79-87 (1985).

2. D.L. Parnas, "On the Criteria to Be Used in Decomposing Systems into Modules," *Commun. ACM* **15**, 1053-1058 (1972).

3. D.L. Parnas, "*Use of Abstr. .t Interfaces in the Development of Software for Embedded Computer Systems*," NRL Report 8047, June 1977.

4. E.W. Dijkstra, "Cooperating Sequential Processes," pp. 43-112 in *Programming Languages*, F. Genuys, ed. (Academic Press, New York, 1968).

5. K.L. Heninger, J.W. Kallander, J.E. Shore, D.L. Parnas, and Staff, "*Software Requirements for the A-7E Aircraft*," NRL Memorandum Report 3876, Nov. 1978.

6. K.H. Britton and D.L. Parnas, "*A-7E Module Guide*," NRL Memorandum Report 4702, Dec. 1981.

7. V.R. Basili and D.M. Weiss, "A Methodology for Collecting Valid Software Engineering Data," *IEEE Trans. Software Eng.* **SE-10**(6), 728-738 (1984).

8. R.A. Parker, K.H. Britton, D.L. Parnas, and J.E. Shore, "*Abstract Interface Specifications for the A-7E Device Interface Module*," NRL Memorandum Report 4385, Nov. 1980.

9. L.J. Chmura and A.F. Norcio, "*Accuracy of Software Development Activity Data: The Software Cost Reduction Project*," NRL Report 8780, Dec. 1983.

10. D.L. Parnas and P.C. Clements, "A Rational Design Process: How and Why to Fake IT," *IEEE Trans. Software Eng.* **SE-12**(2), 251-257 (1986).

11. W.J. Dixon and F.J. Massey, Jr., *Introduction to Statistical Analysis* (McGraw-Hill Book Co., Inc., New York, 1969).

END

10-86

DTIC